Development of a Chinese computer-assisted video instruction system

Jia-Rong Jerome Wen
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Communication Technology and New Media Commons, Engineering Education Commons, and the Instructional Media Design Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/9251

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations. It is the result of an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17” x 23” black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Development of a Chinese computer-assisted video instruction system

Wen, Jia-Rong Jerome, Ph.D.

Iowa State University, 1989
Development of a Chinese computer-assisted video instruction system

by

Jia-Rong Jerome Wen

A Dissertation Submitted to the Graduate Faculty in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Education and Technology

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1989
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER I. INTRODUCTION</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>Need for the Study</td>
<td>8</td>
</tr>
<tr>
<td>Problem of the Study</td>
<td>10</td>
</tr>
<tr>
<td>Purpose of the Study</td>
<td>10</td>
</tr>
<tr>
<td>System Structure</td>
<td>11</td>
</tr>
<tr>
<td>Procedure of the Study</td>
<td>14</td>
</tr>
<tr>
<td>Limitation of the Study</td>
<td>17</td>
</tr>
<tr>
<td>Definitions of Terms</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER II. LITERATURE REVIEW</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief History</td>
<td>20</td>
</tr>
<tr>
<td>Interactive Video Levels</td>
<td>21</td>
</tr>
<tr>
<td>Effectiveness of Interactive Video</td>
<td>23</td>
</tr>
<tr>
<td>Video Discs vs. Video Tapes</td>
<td>25</td>
</tr>
<tr>
<td>Interactive Video Equipment</td>
<td>28</td>
</tr>
<tr>
<td>Authoring Tools</td>
<td>31</td>
</tr>
<tr>
<td>Digitals' Authoring System</td>
<td>36</td>
</tr>
<tr>
<td>Summary</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER III. SYSTEM DEVELOPMENT</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Organization</td>
<td>40</td>
</tr>
<tr>
<td>Voice Processor</td>
<td>43</td>
</tr>
<tr>
<td>Voice Editor</td>
<td>52</td>
</tr>
<tr>
<td>VCR/Slide Controller (Interface)</td>
<td>55</td>
</tr>
<tr>
<td>VCR/Slide Test Program</td>
<td>63</td>
</tr>
<tr>
<td>Courseware Graphics Editor</td>
<td>67</td>
</tr>
<tr>
<td>Test Editor</td>
<td>72</td>
</tr>
<tr>
<td>Integrating the Authoring System</td>
<td>74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER IV. FINDING AND DISCUSSIONS</th>
<th>Page</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CHAPTER V. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary of the Research</td>
<td>84</td>
</tr>
<tr>
<td>Conclusions</td>
<td>88</td>
</tr>
<tr>
<td>Recommendations for Further Studies</td>
<td>90</td>
</tr>
</tbody>
</table>

| BIBLIOGRAPHY                                      | 93   |

| ACKNOWLEDGEMENT                                   | 99   |
APPENDIX ................................................................. 100
Appendix A. Voice Processor VP-1000 ......................... 101
Appendix B. Chinese Courseware Graphics Editor (CCGE) 
Users' Guide ......................................................... 105
Appendix C. Voice Editor BASIC Program List (I) ............ 135
Appendix D. Voice Editor BASIC Program List (II) .......... 137
Appendix E. Voice Process Assembly Routines ................. 140
Appendix F. VCR & Slide Projector Test Program ............... 143
Appendix G. Program List Of The Chinese Courseware 
Graphics Editor (CCGE Program) ......................... 146
Appendix H. Lesson Presentation Program List 
(CCAS Program) .................................................. 178
Appendix I. Voice Playback Program List ....................... 185
Appendix J. VCR and Slide Control Routines ................. 187
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Outcome Variable Used in Evaluation of Effectiveness</td>
<td>24</td>
</tr>
<tr>
<td>Table 2</td>
<td>Report Benefits on Achievement, Training Time, User Attitude, and Performance</td>
<td>24</td>
</tr>
<tr>
<td>Table 3</td>
<td>Computer Functions Necessary for Writing CBI</td>
<td>35</td>
</tr>
<tr>
<td>Table 4</td>
<td>List of IC Description of Voice Processor</td>
<td>47</td>
</tr>
<tr>
<td>Table 5</td>
<td>Function Table of SN74LS139</td>
<td>48</td>
</tr>
<tr>
<td>Table 6</td>
<td>List of IC Description of VCR/Slider Controller</td>
<td>55</td>
</tr>
<tr>
<td>Table 7</td>
<td>Function Table of Port A0 to A3</td>
<td>59</td>
</tr>
<tr>
<td>Table 8</td>
<td>Function Table of Port A4 to A7</td>
<td>59</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 1</td>
<td>Basic Hardware Configuration</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Basic System Configuration</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Graphic Support for CAL</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Interactive Video Hardware Configurations</td>
<td>32</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Relationships of Graphics Instructions</td>
<td>37</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Overall Structure of the CIVS System</td>
<td>42</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Block Diagram of Voice Processor</td>
<td>45</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Voice Processor Circuitry</td>
<td>46</td>
</tr>
<tr>
<td>Figure 9</td>
<td>IBM I/O Configuration</td>
<td>51</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Voice Memory Location of Voice Data</td>
<td>53</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Menu of Voice Editor</td>
<td>54</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Block Diagram of the Voice/VCR/Slide Control Interface</td>
<td>56</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Circuitry of the VCR/Slide Controller</td>
<td>57</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Bit-control Function of the VCR/Slide Controller</td>
<td>61</td>
</tr>
<tr>
<td>Figure 15</td>
<td>The Video Signal Cable the TV-Mouse</td>
<td>62</td>
</tr>
<tr>
<td>Figure 16</td>
<td>The Circuitry of the Slide Projector Emulator (SPE)</td>
<td>64</td>
</tr>
<tr>
<td>Figure 17</td>
<td>VCR/Slide Function Selection Menu</td>
<td>66</td>
</tr>
<tr>
<td>Figure 18</td>
<td>VCR/Slide Function Testing Menu</td>
<td>66</td>
</tr>
<tr>
<td>Figure 19</td>
<td>CCGE Editing Screen</td>
<td>69</td>
</tr>
</tbody>
</table>
Figure 20. Relationships Between CCGE, PASCAL, and ET Graphic Codes .................................................. 70
Figure 21. Character Sets Editing Screen ................................................. 71
Figure 22. Query Input Screen ............................................................... 73
Figure 23. Main Menu of CIVS Authoring System ................................. 74
Figure 24. Screen of Script Building ...................................................... 76
Figure 25. Flowchart of Courseware Production ................................. 77
CHAPTER I. INTRODUCTION

People are facing a new technological revolution, which has been called "the fourth revolution," and which will change the way people learn. The computer now offers a myriad of options for teaching and learning. Deliveries in education are moving toward a future in which computers will comprise the dominant delivery system in education. Lessons can be designed that incorporate text, graphics, sound, and even animation. Although currently Computer Assisted Instruction (CAI) faces some difficulty in many parts of the world, a future with computers as the dominant educational delivery system seems inevitable.

CAI research began in the United States early in the late 1950s. A number of large-scale experimental projects have been conducted since then, and their results have implications for the future use of CAI as a classroom tool. For the past ten years, CAI has been attracting the attention of governmental agencies and the computer industry in Taiwan, R.O.C. As indicated by their previous studies, the CAI researchers in Taiwan have confronted considerable problems.

Background

The educational system in Taiwan is somewhat similar to that in the United States, and the structure and content of science programs
are nearly the same in both countries. Many concepts of the social sciences, which have been developed by western countries, were introduced and adapted well in Taiwan. However, the same cannot be said of computer assisted instruction.

The CAI concept was first introduced to the island by Tamkang University in 1975. For a number of years now, with the support of governmental foundations, CAI has become a popular topic on this small island, and many CAI related studies have been conducted (Wu, 1987).

The most significant handicap for CAI research in Taiwan is that there is no computer adapted to the Chinese language. The computer, which was developed by the west, was designed for "spelling" languages, not for a symbolic language such as Chinese. In order to facilitate communication between users and computers, the computer factories in Taiwan have had to revise the computer architecture for Chinese input/output. By separating Chinese words into several root characters, Chinese words can thus be input to the computer. Unfortunately, the complexity of Chinese input has so far restricted CAI research and implementation.

Another difficulty for CAI development in Taiwan is that authoring aids for developing the material, and thus the production of CAI courseware, are limited. A computer system with an authoring program that would enable teachers to write educational software without special computer programming skills seems necessary.

Moreover, without vocal feedback or explanation, students would have a hard time in reading the Chinese words on the computer screen,
since the Chinese words are complex and the characters are condensed inside the words.

Unlike other media, the computer demands the user's active participation, and for this reason, as well as because of its potential for individualized instruction, the computer has distinctive advantages over the other media as a tool of instruction (Chambers, 1974).

Conventional CAI involves a CAI system that uses a general computer set in which the computer is equipped with a monitor, keyboard, and auxiliary memory devices. A principal benefit of CAI as an instructional delivery system is its capability to adapt instruction to the needs of individual learners. It also permits the learner to proceed at his or her own pace. Reinforcement of learning in such a situation is immediate and systematized, according to the theory of instruction: such instant feedback should result in more effective learning. The weakness of the current conventional CAI system is that it cannot provide accurate colors, graphics, and voices, while these factors are strongly needed for science instruction.

Despite the tremendous benefits associated with CAI, there are many situations in which CAI is not instructionally adequate; for regardless of the formidable capabilities of microcomputers, they are often inadequate due to the unique requirements of different curricula. For example, computer-generated graphics are often
incapable of accurately depicting intricate visually-oriented learning sequences.

Traditional audio-visual aids, such as film, radio, and television, enjoyed tremendous success in providing fidelity images. These advantages can be retained by combining a computer with a video tape recorder and a slide projector. The educational potential for use of such a system in the classroom is enormous (White, 1987).

The Computer Assisted Video Instruction (CAVI) system combines the CAI system with audio-visual devices, such as video discs, which provide the ability to merge video and audio with computer graphics. The best features of each medium can be used at the same time. Video instruction combined with a vocal explanation provides clear and detailed concepts that far exceed the limited capabilities of the conventional computer system (Alton, 1986). One kind of video system that is similar to the CAVI is called an interactive video.

Interactive video extends the feasibility of both computer and television. Although the computer is a multi-purpose machine, individual programs are designed for specific applications (Bayard-White, 1985). An interactive video module may be used not only for different audiences but for different purposes: i.e., a catalogue of information, a learning program, etc.

The CAVI system differs from the interactive video in that it is specially designed for instructional purposes. A sophisticated CAI system, such as the CAVI system, provides authoring functions to help teachers develop CAI materials. CAVI systems provide new capabilities
for representing and controlling instructional strategies and capabilities, which were unavailable to designers of conventional video tapes. The new capabilities include random access to video and audio displays and the coordination of video and audio displays with computer text and graphics. This new teaching method is usually suitable for implementing technical instruction in Taiwan.

Interactive video has the advantage of both of these technologies and offers in a single medium a blend of visual and textual information and computer-assisted learning techniques. Bayard-White (1985) enumerate four possible uses for interactive video in schools: a teaching tool, a small-group learning medium, an individual learning aid, and a resource for data access. Despite its versatility, there is a great deal of uncertainty surrounding the question of what young children can learn directly from it.

Video recorders are numerous in nearly all secondary schools and are increasing in number in primary schools. Therefore, interactive television is a development that warrants consideration. Teachers can use the machine to preview, stop/start, freeze-frame, play-back, play-forward, etc.

It is also possible for teachers to record their own material, as small video cameras are not too expensive to acquire. Teachers could plan and create modules, then record the material and edit as necessary.

The optical video disc is already in existence and is in use in
industry and higher education. It is perhaps a more flexible system than the magnetic video tape and provides greater control for the teacher and students. The system is computer-controlled and enables the computer to instruct a search for a given frame number or to play from one frame to another. Laurillard (1983) maintains that, for teachers, one of the main advantages of the interactive video disc is its capability to act as a vast resource of pictorial data. A video disc can store twenty minutes of motion film and still have room for a slide bank of twenty-five thousand single pictures and can access and display any slide within a few seconds. As few as one video disc per curriculum subject would enormously enhance a teacher's classroom resources.

But educators must be aware that the development of such video disc materials takes time and that it is expensive. Presently, images must be taken into the video tape first, then the video images are transferred from the video tape to the video disc, with the use of special equipment. The production of video discs, which are considered to be an ideal digitized picture presenting system, is rather complex and expensive. The weakness of the video disc system is that teachers cannot record their own pictures onto the disc (Pauline & Hannafin, 1987). Choat et al. (1987) believes that in the foreseeable future, there is no likelihood of video discs becoming commonplace in schools for young children. Meanwhile, the video recorder must be regarded as the most powerful tool of change.
As mentioned earlier, the computer has been designed with an alphabet-based keyboard, a fact which makes it difficult to use Chinese characters to communicate with the computer. Teachers and students find themselves facing a barricade when they attempt to put Chinese characters onto the keyboard or to display Chinese characters on the monitor. This also causes some communication problems in course content. The computerized multimedia system will help in displaying course content clearly when Chinese courseware is used.

In addition, recent innovations in integrated speech products permit a computer to speak. It makes it possible for the computer to provide an immediate audio feedback integrated with computer assisted instruction. This is a very important factor for successful learning (Bosco, 1986). The new technology should be incorporated into the classroom as a supplement to other teaching methods.

If a program has a voice component, the student can listen to the audio portion and simultaneously see images develop on the screen (Braden, 1986). This is especially important for educational programs in the sciences, where visual imagery, diagrams, and models are so necessary. Wessel (1986) has predicted that the talking computer has a future in the classroom.

It seems worthwhile to develop a system that links a computer with a video cassette recorder, slide, and voice processor as multimedia interactive system. In order to help teachers in R.O.C. to use this system, special consideration for the Chinese operating system that allows the computer to display Chinese characters is
necessary. An authoring program is also required, since we expect teachers without a strong programming background to create lessons. In this study, the researcher developed an authoring program which will help such teachers to design courseware.

Need for the Study

The motivation for this work arose from a desire to provide a CAI tool for teachers in China. By designing a multi-media CAI system, it is hoped that teachers will be able to make their own Chinese CAI courseware without difficulty. For a long time, teachers in Taiwan, the Republic of China (R.O.C.), have had difficulties in designing interactive courseware. One of the reasons for these difficulties is that the computer can not easily handle Chinese characters. Chinese CAI Courseware does not allow students to answer in Chinese. The only way that students can reply to the questions displayed by the computer is to make meaningless key-strokes. Multiple choice is the way which is most frequently applied by teachers to ask students questions. Since multiple choice examinations elicit answers like 1, 2, 3 or a, b, c CAI courseware makers can avoid dealing with Chinese characters. However, limiting the use of Chinese characters also reduces the interactive capability of Chinese CAI courseware.

Many studies in Taiwan have indicated that some major problems still confront the teacher who attempts to design courseware. These problems for the courseware design are listed below:
(1) Difficulty in displaying high-fidelity pictures
(2) Difficulty in designing an animation screen
(3) Difficulty in providing interesting music
(4) Difficulty in succinctly explaining courseware content

The application of an interactive video system would solve most of these problems (Elliotto, 1984). Currently in Taiwan, there is limited or no interactive system which has been used in computer assisted instruction. On the other hand, this educational technology has been implemented in Western schools for a period of time, and the CAVI system is a new hope for implementing CAI in Taiwan. Although the CAVI system does not solve the Chinese input problems, it provides a variety of output channels for increasing the modes of learning perception. It is hoped that this study will improve the ability to create Chinese courseware to enhance interaction between students and computers. Additionally, it is the intent of this study to establish a module for using multimedia CAI in Chinese-speaking countries.

In summary, the major difficulties for CAI in Taiwan are the following:

(1) The Chinese language is based on a complex system of characters, rather than on the simpler concept of a 26-discrete alphabet symbols (as is English). Consequently, it is more difficult to provide an interactive program of Computer Assisted Instruction by using the conventional
computer equipment for Chinese use.

(2) Ideally, a CAVI system would incorporate the use of video-disc (laser-disc) technology and thus offer rapid, random access to video images. Currently, the application of video-disc technology in CAI in Taiwan is severely limited, however, as a result of the expensive and sophisticated equipment needed to record video images on discs.

(3) The computer cannot speak. In developing a program of computer-assisted instruction for students, providing voice feedback is important, especially in Chinese speaking countries.

(4) Because of the absence of authoring aids, or an authoring program, teachers in China have had difficulty in making their CAI courseware.

Problem of the Study

The problem of this study is to investigate and develop a Chinese interactive video instructional system with the use of a micro-computer.

Purpose of the Study

The purpose of this study is:

(1) to develop a Chinese CAVI system that would be useful to
teachers in creating Chinese multi-media CAI courseware,
(2) to explore a model of integrating traditional audio/visual aids with a computer in order to create an interactive teaching system in developing Chinese CAI courseware,
(3) to design a voice processing interface that would provide students with instant voice feedback in the above-mentioned CAI system,
(4) to design an authoring program in this interactive CAI system that would help teachers to edit Chinese courseware.

System Structure

Two components, a computer interface and an authoring program, were designed as the major components of this research. An interactive video system interfaces a computer with a video cassette recorder (VCR), a slide projector, and a voice processor. The sequence of presenting the video lesson is under control of the computer program. The computer is thus used to teach, query, remediate, or otherwise support the video lesson (Floyd, 1982). When visually-oriented presentations are converted into interactive video/audio lessons, the unique capabilities of a computer, such as individualization and controlled pacing, can be realized, in addition to the instructional value of the sights and sounds of the lesson. The basic hardware configuration of the system is illustrated in Figure 1.
A. Hardware Components

The diagram, Figure 1, includes four main hardware components:

(1) COMPUTER (IBM XT/AT or IBM compatibles):
   . At least 640K RAM
   . Dual 360K Disk Drive
   . Keyboard
   . Enhance Graphic Adapter
   . Monochrome Graphic Adapter
   . Interactive Video Control Interface
   . EGA monitor

(2) VOICE PROCESSOR:
   . Dynamic Microphone
   . Voice Processor Interface

(3) VIDEO:
   . Video Cassette Recorder
   . Video Camera

(4) SLIDE:
   . Slide Projector
   . Photo Camera
Figure 2. Basic System Configuration
In order to produce interactive lessons, a high-level authoring program (or system) is used to control the lesson. The authoring program utilizes a simple format that enables the designer to create lessons without having a strong programming background. The authoring program consists of four subprograms: the test editor, graphics editor, voice editor, and VCR/Slide control program.

B. Software

This system includes three major software packages.

1). MS DOS (Microsoft Disk Operating System)

2). Chinese Operating System

3). CAVI Authoring Program (coded in Pascal 3.0):
   a. Test Editor
   b. Graphic Editor
   c. Voice Editor
   d. VCR/Slide Control Program

Procedure of the Study

The following outline was established in order to accomplish the objectives of the study of the development of a Chinese CAVI System.

1. Research and Information Collecting -- The first step in this system development project included a review of literature, an evaluation of CAVI systems, and a system observation.

2. Planning -- Based upon the research findings, a detailed proposal was prepared; including both pedagogical and
methodological details. The proposal defined the general scope of the project. The systems for both hardware and software were defined in this planning stage.

3. Voice Processor Design -- This was the first stage of the design process. According to the functions defined, the circuit of the voice processor was designed. Components were also purchased and assembled, and then the interface was tested. In the mean time, in order to make sure that the voice interface worked as expected, it was necessary to design the voice control software early in the project.

4. Voice Cassette Recorder and Slide Projecter Controller Design -- The interface functions were defined, and then the circuit design was conducted. Next the VCR/Slide controller was assembled and tested. The program for the VCR/Slide controller was developed at this stage, and thus proper functioning of the interface could be ensured.

5. Design of the Chinese Courseware Graphics Editor (CCGE) -- This stage was a most important stage in the design process. In fact, the detailed design of the modules was perhaps the most critical to the overall quality of the product. The system analysis, the flowchart design, and the screen arrangement of the CGE program, were carefully determined. Separate experts were consulted for their opinion of the course. A CGE users' guide was written as a supplementary document for the possible
users.

6. Design of the Test Editor — Again based on the functions
defined in the proposal for the test program, both a test editor
and a test retrieval program were analyzed and coded.

7. Development of Subroutines — Subroutines were revised from
each subprogram in this stage. These routines have to be
consistent in format in order to be called and executed by
the main authoring program.

8. Integration of the Program — This stage involved the coding of
the main program. Several pieces of subroutines were integrated
into a main program. Now, for the first time, the program was
actually running. The program was then inspected by the project
researcher and run many times to see if it was doing what it was
supposed to do.

9. Internal Revision of the Program — Based on an internal review,
a revision was conducted. These two steps were repeated many
times before there was a version considered satisfactory.

10. Peer Review — After the internal revision, the system was also
reviewed by expert teachers, including professors on the
research committee.

11. Final Revision — Based on the information collected at peer
review stages. The final production was once more revised.

12. Final Report — The researcher wrote a report of the research
results based on the investigation and the development of the
interactive video system.
Limitations of the Study

This study was conducted under the following limitations:

(1) Because of financial and time constraints and because the major target consumers of this study will be the teachers in Taiwan, selecting a sample in the U.S. was not possible, therefore, there was no field testing. It is recommended that field tests should be conducted in Taiwan, R.O.C., to collect information that can be used to improve the system developed in this study.

(2) Using this interactive CAI system, some devices that are connected to the computer have to be reconnected with wires and cables. It means that the VCR and the slide projector need to be controlled with some additional connectors in their circuits. This technical requirement may prevent this system from being feasible or readily succeeding in some schools.

(3) One of the most serious posed problems with the use of authoring systems is that of creating frame-oriented tutorials packages. Program branching is not as flexible as one would like. The authoring system developed in this study was not able to eliminate this shortcoming. In order to create innovative simulations or intelligent tutorial packages on the system developed by this study, the teachers
will need to use Turbo Pascal and make good use of those Pascal subroutines supported by the system.

Definitions of Terms

(1) Interactive Video — Any video program in which the sequence and selection of messages is determined by the user's response to the material. Viewer participation in an interactive video program is a critical factor. Viewer participation or involvement may take a number of different forms, such as answering questions, manipulating a control during simulation, or simply choosing which segment of material to view from a menu. As a result, within the same program, a number of alternatives or paths are available to different users, and the route followed by individuals may vary significantly (Floyd, 1982).

(2) Authoring System — An authoring system enables the message designer to enter all of the sequences and events in a step-by-step procedure, according to a lesson plan. This is done by using a series of prompts, or questions, to convert a lesson into a interactive program, complete with computer-generated tests, video segments, questions, and branching paths. The author's instructions are translated into machine-language commands so that the appropriate segment or
information follows each response. As a result, nonprogrammers can enter lessons into a system without knowing computer language or syntax.

(3) Courseware -- A computerized course, typically including text, workbook, and software, for a specific computer system. Sometimes people refer to such courseware as educational software (Wagers, 1984).

(4) Frame -- One in a series of visual displays on the monitor of a computer system.

(5) Editor -- A program that allows the author to build individual "items," which can be pieces of text, forms, or questions that will subsequently be presented to the students. The term "item" is used instead of the more common term "screen" for this reason: the word "screen" denotes a static display occupying the display of the terminal (Heck & Grimm, 1984).

(6) Interface Cards -- An interface card is a circuit board that fits into a slot inside the microcomputer and plugs into the peripheral equipment that gives the microcomputer control of the equipments' functions.
CHAPTER II. LITERATURE REVIEW

The review of literature has been organized as follows: first, a brief history of video technology and computers in education; second, an investigation and discussion of the early stages of video discs and video cassette recorders in education; third, the nature of interactive video; fourth, the evidence for effectiveness; fifth, the design tools of interactive courseware; and sixth, the system and structure of the Digital authoring system. The concept of courseware graphics editor on Digital computer is also presented.

Brief History

Video technology traces its roots to the 1920s in the U.S. After the birth and growth of commercial television in the 1940s, a number of educators began to explore the possibilities of television in education. Today, slowly but surely, video technology has found its way into schools. The great majority of schools today make use of video cassette recorders (VCRs), and an increasing number of institutions are beginning to explore the possibilities of video discs.

Video disc players were introduced to the market in 1978 (Daynes, 1984). At that time, the video disc players were cheaper than VCRs, but the public was skeptical because video disc players could not
record as could VCRs. Although optical laser systems suffered in the home market, they have undergone a rebirth of sorts as a technology of tremendous educational potential (Lehman, 1986).

The idea of linking computers and video arose in the 1970s. In 1975, the National Science Foundation (NSF) sponsored a conference of leading computer and communications experts and identified video disc technology as the most promising near-term technology for education (Molnar, 1979). Today, these are a variety of interactive video systems, both video tape and video disc-based, there are available for educational use.

**Interactive Video Levels**

Bunderson (1983); Priestman (1984), and Yampolsky (1983), have defined the common features that will serve as an operational definition for interactive video. According to these sources, interactive video is the use of a video delivery system designed in such a way as to respond to choices made by the individual user. These choices may be spontaneous on the part of the user or they may be prompted by the system.

Interactive Video is the integration of video and computers. This technology combines sound, motion and color video with the logic, branching, and testing capabilities of a computer/microprocessor, in order to present information and instructional material.

OmniCom Associates (1983) indicate that every interactive video
system has four basic components:

(1) Microprocessor/Computer
(2) Video tape/Video disc player
(3) Monitor/Receiver
(4) Interface (The component that makes the system interactive.)

Interactive video programs are often classified according to the levels of interactivity. Essentially, the levels are determined by the hardware involved and the way in which the interactivity occurs. There seems to be no agreed upon the definition of interactive levels. Experts in the field have their own concepts (OmniCom Associates, 1983; Lehman, 1986; Salpetor, 1986; Smith, 1987; Rhodes & Azbell; 1985). Their common features are described in terms of levels.

Miller (1987) defines the levels of interactive video:

A level-one system consists of a video disc player, a remote control to branch to predefined chapter stops. In a level-two system, the player has an internal microprocessor which can be programmed to perform a certain playback sequence and respond to review choices during the program. In a level-three system, the video disc is controlled by an external computer. The computer can be a micro or a mainframe. The important point to remember is that the computer itself contains the program which responds to user input and causes the player to perform accordingly.

It is within a level-three configuration that all benefits of computer-assisted instruction are joined with the benefits of television. These two media working together provide a learning tool more powerful than either separate component.
Effectiveness of Interactive Video

Interactive video offers new opportunities for science instruction. Many different types of projects involving video discs and video tapes in education are reportedly underway all over the country in the U.S.

Traditional forms of computer-based instruction -- drill and practice, tutorial, and simulation -- can be enhanced with interactive video. Instead of being restricted to the computer's often limited graphics and sound, the sound and clear visuals of the video disc (or the video tape) can be made an integral part of instruction.

Three major sectors use and study interactive video: the military and government, private industry, and education. Daynes (1984) states that the military is one of the leaders in interactive video applications.

Can interactive video ever play a significant role in education? Much research has been conducted in the investigation of this question. By analyzing data based on 28 reports, Bosco (1986) found that "the consequences of interactive video will depend on the design and use of specific programs." To illustrate Bosco developed a table containing a summary of the basic aspects of the evaluations. Table 1 lists the outcome variables used in the studies. The table indicated that achievement, user attitude, training time, and performance were the typical variables used to measure outcomes.
Table 1. Outcome Variables Used in Evaluation of Effectiveness

<table>
<thead>
<tr>
<th>Outcome Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 Achievement</td>
</tr>
<tr>
<td>16 User attitude</td>
</tr>
<tr>
<td>11 Training time</td>
</tr>
<tr>
<td>6 Performance</td>
</tr>
<tr>
<td>4 Instructor attitude</td>
</tr>
<tr>
<td>5 Cost</td>
</tr>
<tr>
<td>3 Retention</td>
</tr>
<tr>
<td>2 User-estimated confidence</td>
</tr>
<tr>
<td>1 Number of users repeating module</td>
</tr>
<tr>
<td>1 Time on performance test</td>
</tr>
<tr>
<td>1 Self-assessment of school learning</td>
</tr>
<tr>
<td>1 Learning productivity</td>
</tr>
<tr>
<td>1 User's perceived speed of skill acquisition</td>
</tr>
<tr>
<td>1 Instructor's perceived transfer of learning</td>
</tr>
<tr>
<td>1 Instructor's perception of transfer of learning</td>
</tr>
<tr>
<td>1 Program implementation</td>
</tr>
</tbody>
</table>

In Table 2, Bosco listed the findings for the four most common dependent variables: training time, achievement, performance, and user attitude, when the reported benefits involving the use of a statistical test were examined, benefits were most prevalent on user attitude and training time variables. They were less prevalent on the achievement variable.

Table 2. Report Benefits on Achievement, Training Time, User Attitude, and Performance

<table>
<thead>
<tr>
<th>Benefit Reported</th>
<th>No Benefit Reported</th>
<th>Mixed Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stat Test (Used) (NO)</td>
<td>Stat Test (Used) (NO)</td>
</tr>
<tr>
<td>Achievement</td>
<td>7 4</td>
<td>5 0</td>
</tr>
<tr>
<td>Training Time</td>
<td>5 4</td>
<td>2 0</td>
</tr>
<tr>
<td>User Attitude</td>
<td>5 4</td>
<td>2 0</td>
</tr>
<tr>
<td>Performance</td>
<td>1 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>
One rationale for interactive video is to accommodate differences among learners, and the large standard deviations for training time in Bosco's study indicate that differences exist.

Although some have questioned the interpretation of research results (Clark, 1985), there is a great deal of evidence that computer-assisted interactive video in general is both effective and efficient (Hannafin, Phillips, & Tripp, 1986; Ho, 1986; Dalton, 1986a; Dalton, 1986b; Hannafin, 1985; Smith, 1987). Despite its short history, specific evidence for the effectiveness and efficiency of interactive video is growing; there often seems to be more differences between studies than similarities, which makes it difficult to form conclusions.

**Video Discs vs. Video Tapes**

The last decade has seen a tremendous increase in the use of video materials in curriculum development. At universities all over the country, case-study video tapes are becoming the preferred mode of teaching the complexities of decision-making and behavior in different subjects. The following is a summary of the advantages of video tapes educational technology:

1. Video tape can also offer accurate frame access, limited still frame capacity, and slow motion on some well modified video hardware.
2. Video tape offers extended lesson play time
(Newell, Sims & Myers, 1983).

(3) Video tapes are typically less expensive to provide than video discs for limited distribution. Since it can be edited directly (Allen, 1986).

Unfortunately, video tapes have the following major drawbacks for use as educational aids (Gerstein & Sasnett, 1987):

(1) It is difficult and time-consuming to access any part of the material at will, necessitating long and inaccurate searches.

(2) The presentation is 'cast in stone' and cannot be altered.

(3) Image quality is doomed to deterioration due to contact between the physical media and its playback device, making it difficult to maintain archival quality over time.

(4) The coordination of supplementary documents with the visual content is clumsy and expensive.

(5) It is impossible to provide a table of contents due to the lack of built-in addressing mechanisms.

The combination of computer-controlled video discs and high-quality graphic displays is capable of solving the problems inherent in video tape educational materials. Video disc technology offers unique positive features (Hannafin & Phillips, 1987):

(1) The video disc is read without physical contact, it is exceptionally durable.

(2) Video disc technology permits rapid access to various segments of a lesson.
Video disc technology offers excellent display quality and slow motion display.

Each frame on a video disc is implicitly identified with a unique frame number, allowing precise and rapid "frame accurate" location of lesson segments.

Nevertheless, video discs are not without limitations. Presently, most videodiscs are a "fixed" medium that can be neither recorded nor edited using standard video production facilities. In 1977, McGraw-Hill funded an interactive video disc on biology which was developed by WICAT, Inc. (World Institute for Computer Assisted Teaching). This video-disc project is generally recognized as the first interactive disc developed for individualized learning and is considered by some to have been the most significant breakthrough in instructional technology since the invention of the printing press (Reigeluth & Garfield, 1984). To many, videodisc remains superior due to inherent features such as the random versus linear access capability (Hoffos, 1983). But, Videodisc's high cost -- especially for lessons of limited distribution -- is offset by its advantages, for many educators.

In order to present a balanced picture of both delivery systems, the major advantages are summarized and compared below.

The major advantages of video discs are:

(1) Rapid access to any frame.

(2) Unlimited still frame capability, with no wear on discs.
Video tape systems have the following advantages:

(1) Record capability.
(2) Can be duplicated locally.

However, video discs do several things that video tapes can not. Floyd (1982) illustrated several key points that would help to answer the questions at the program planning stage.

. If there are to be 10 - 100 copies of the program, tape may be better.
. If the program is needed within 7 days, tape may be better.
. If the program will be played for many times, over an extended period, discs may be better.
. If a linear format is desired, tape may be better.
. If groups will view the program, tape may be better; if an individual, disc is superior.

Video discs can store one still frame in 1/30th a second.

Clearly, even though the video discs, are in many ways superior to video tapes, video tapes will nevertheless be more appropriate in some cases.

Interactive Video Equipment

Interactive video equipment is undergoing dramatic change and innovation as manufacturers compete to develop hardware that meets the
demands of the marketplace. As a result of these rapid changes, the hardware will continue to evolve until a system is introduced that alters the complexion of the industry.

Conventional interactive video has four basic components — microcomputer, monitor, video player, and an interface, and any of the interactive video resources can be utilized within a computer-assisted learning (CAL) environment of the type depicted schematically in Figure 3 (Barker, 1986).

Figure 3. Graphic Support for CAL
The system illustrated in this figure consists of three units: an image production system; an image storage; and a picture-display facility. The images that are produced by the image-production unit can be generated in real-time (that is, created and used as they are needed), or they can be fabricated and then committed to storage for later use. In the latter case, there is a diverse range of image storage possibilities: video tape, photographs, slides, laser disc, or compact disc, and so on.

Of equal importance to image storage and production is the concept of image display. To that end, a variety of techniques may be employed. These range from the use of conventional audio-visual teaching aids and project systems to the use of high-resolution color graphics systems.

Generally speaking, hardware developments for interactive video are running at a pace similar to those of computing hardware. Figure 4 shows four hardware configurations that have become readily available over the past two years (Butcher, 1986). The differences in the four configurations lie in the ways in which the video signals from the microcomputer and the video disc player are or are not mixed.

Configuration A makes no attempt to mix the video signal. This computer is used simply to control the functions of the video disc player.

Configuration B attempts to improve on configuration A by reducing the number of screens in the system to one. Both
microcomputer and video disc player signals may be displayed on the screen, but only one signal may be displayed at one time. The switching of the video signals is undertaken through computer control.

Configuration C is provided by the Philips Laservision range of video disc players. The computer can both control the functions of the video disc player, and provide teletext (text) and graphics for superimposition on the video picture.

Configuration D enables the mixing of microcomputer and video disc player video signals so that they may be displayed together on the same screen.

Authoring Tools

Courseware production requires the use of some form of control language. Currently, there are three approaches to courseware production. These involve the use of conventional programming languages such as BASIC or PASCAL, authoring language such as PILOT (Hannafin, 1984) or DAL (Reed & Porter, 1984), or authoring systems such as Digital Authoring System (Reed & Porter, 1984) or PLATO.

Avner, Smith and Tenczar (1984) indicated that a survey made by the National Center for Educational Statistics showed that the Apple II was the most-used computer in classrooms. Despite the availability of special Computer-Based Instruction (CBI) languages for the Apple, publishers and users reported that most material was being written in BASIC. Though BASIC provides reasonable access to the capabilities of
Figure 4. Interactive Video Hardware Configurations
(Butcher, 1986)
the Apple and allows rapid test-revise-test cycles, it is not designed for CBI. Smith suggested augmenting BASIC with a set of commands that would add CBI capabilities presently lacking in BASIC and in available tool software. The many programmers familiar with BASIC would thus be able to use CBI techniques easily without having to learn a new language and without giving up the flexibility provided by a general-purpose computer language.

Authoring languages have been custom-designed to make the tasks of designing CAI -- and interactive video -- more readily executable than by using complex sequences of computer programming language code (Burke, 1982). Conceptually, authoring languages treat the multiple program statements of programming language statements as singularly defined macros, making good CAI programming less complex than through the programming language code. One of the most popular interactive video authoring languages presently available for microcomputer use is Apple Super PILOT.

Authoring systems are those which provide embedded CAI logic, requiring little or no facility with programming, or in many cases, instructional design logic. Typically, a designer or subject matter expert inputs information, usually under maximally cued and prompted conditions, into a "friendly" nontechnical system. The system, in turn, organizes and formats inputted information, displays the next set of logical CAI options, and continues until completion of the lesson (Burke, 1982).
The principal difference between an authoring system, an
authoring language, and a programming language is that an authoring
system does not require knowledge of a computer language, but the
others do. PASCAL, for example, is a common programming language.
Unlike PASCAL, DAL is a computer language in which the commands have
been developed especially for writing educational applications. DAL
is an authoring language. PASCAL and DAL can be used to author
computer-aided interactive video courseware, but they both require
programming skills on the part of the user.

Authoring systems generally require little or no programming
skill. They are "menu-driven," in general. That is, the user may
determine what he or she wants to do from a number of options that
appear on the screen. If the user needs to enter the specific
commands, he/she is prompted at the appropriate place in the control
program. This enables the user to focus on design and development
rather than on programming the computer (Rhodes & Azbell, 1986).

Long and Squire (1984) had indicated the capabilities of new
Courseware, an authoring system for APPLE computer. The six functions
identified by Long can be considered as important features that an
authoring system should offer.

(1) Text - Users can change text size, font, and
color. They can select a combination of different
kinds of foreground and background colors.

(2) Graphics - Users can access and modify
any bit-mapped graphics font. They can place
these graphics at any location on the screen,
and can copy, move, or negate them. Users can
also fill any rectangular portion of the screen with any of a number of patterns.

(3) Special Functions - Users can embed displays, which can be very effective for creating a sense of movement. They can embed a "wait for keypress" to allow the student to control the pace of the presentation.

(4) Answer Processing - User can create questions using the same functions as described above, with the cursor positioned for the student response, word selection, or fill-in items. Users are prompted to enter data concerning the correct, incorrect and unexpected answers with the labels of corresponding feedback frames.

(5) Lesson and Course Building - Users can label screen files according to conventions indicated in the manual, and the system will automatically link screens into lessons, through the lesson builder.

(6) Presentation to the student - When the student take a course created with this authoring system, he/she can make a selection from the course menu and proceed to the lesson file containing that selection.

Table 3. Computer Functions Necessary for Writing CBI

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase Screen</td>
</tr>
<tr>
<td>Display Text and Numbers</td>
</tr>
<tr>
<td>Draw Figures and Graphics</td>
</tr>
<tr>
<td>Do Calculations</td>
</tr>
<tr>
<td>Accept User Input</td>
</tr>
<tr>
<td>Judge Responses</td>
</tr>
<tr>
<td>Erase Selected Screen Area</td>
</tr>
<tr>
<td>Store Data</td>
</tr>
</tbody>
</table>

Avner, Smith, and Tenczar (1984) outlined the essential functions for writing interactive computer-based instruction (CBI) materials. The AppleSoft BASIC available on the APPLE II microcomputer allows all
of the functions summarized in Table 3. Indeed, Applesoft BASIC has been used to write sophisticated interactive CBI (Smith, 1981).

**Digitals' Authoring System**

Digitals' Courseware Authoring System (CAS) uses the Digital Authoring language (DAL). Complementing CAS are such graphics utilities as a graphics editor, a character set editor, a data plotting package, a student router, and a student record-keeping system (Reed & Porter, 1984).

Digitals' CBE system is expected to expand to include tutorial templates, drill & practice templates, and interactive video capabilities. To run CAS, a VAX series supermini-computer that will run the VAX/VMS operating system is required.

DAL is a high-level authoring language, modeled very closely after TUTOR -- an authoring language for the PLATO system -- and enhanced with elements of PASCAL. It has a two-level hierarchy, composed of a lesson level and a unit level. The lesson level carries out initializations and charts the flow of the lesson. Units in DAL are similar to procedures in PASCAL and are called by the lesson level.

**Graphic Instructions**

ReGIS, for Remote Graphics Instruction Set, is the graphics language understand by the terminal. There are two ways to generate
ReGIS code.

DAL contains a large variety of graphics commands, which can be sent to the terminal, interpreted as ReGIS graphics instructions, and displayed on the screen.

ReGIS commands can also be generated by the graphics editor, which can save the display in a slide file. This file can be integrated into a CBE lesson by using the DAL 'SLIDE' command. Relationships between DAL, CGE, and ReGIS are shown in Figure 5.

Figure 5. Relationships of Graphics Instructions
The Courseware Graphics Editor

The Courseware Graphics Editor (CGE) is a tool that is used to produce graphics displays. The user creates a display in CGE by placing objects such as boxes, circles, line drawings or text on the screen at the desired locations. For each object, the user may declare various attributes associated with that object, including size, rotation, color, filling, etc. A file that contains either Digital Authoring Language instructions is then saved by the request of the user from the CGE operation (Boysen, 1987).

Summary

This chapter presents a review of interactive video literature concerning history, levels, effectiveness, discs and tapes, equipment, and tools. Video discs and video tapes have been briefly discussed in view of hardware features and issues associated with cost and distribution.

Several studies related to the evaluation of the interactive video have also been presented. It appears from these studies that computer-assisted interactive video, generally speaking, is both effective and efficient.

The major findings of this preliminary segment of the research can be summarized as follows:
1. CAVI can be used to effectively improve learning, but it can also have deleterious effects on learner attitudes if used in an improper manner (Dalton, 1986 January).

2. When compared to video tape on the basis of hardware factors, video disc often appears superior. But each of the hardware options have distinct advantages and disadvantages.

3. One of the major limitations of interactive video today is the lack of high-qualities educational software. Software is cited as a major problem by a number of experts (Cambre, 1984; Lindsey, 1984; Hon, 1985). The software problem is compounded by the lack of compatibility of interfaces and interface software.

4. Authoring systems offer obvious advantages. They tend to be very easy to learn and to use, and require only minimal training for creating interactive video. However, authoring systems are often subject to limitations inherent in the design of the system itself. Frame protocol, question options, answer judging, and a host of other options are largely predetermined and not readily manipulatable by the instructional designer.
CHAPTER III. SYSTEM DEVELOPMENT

CIVS is an acronym for Chinese Interactive Video System. The major task of this study was to develop the CIVS system, a computer-assisted video learning environment with Chinese character display capability. Specifically, CIVS is an interactive video/voice system, which interfaces a computer with a VCR, slide projector, and voice processor. The sequence, rate of presentation, and conditions of the video lesson are under direct control of the computer program. The computer is thus used to teach, query, remediate, or otherwise support the audio-visual lesson. The unique capabilities of the CIVS are that with the support of the Chinese ET -- a trademark of a Chinese word generating system, the CIVS system allows teachers to create their Chinese interactive video program with the help of the Chinese authoring system developed in this study.

System Organization

The CIVS system consists of two interfaces and a number of program modules. Included among these are:

Interface: (1) the VCR/slide controller

(2) the voice processor

Program: (1) the VCR/slide control program
(2) the voice control program
(3) the courseware graphics editor
(4) the test-editing program

Each of the programs in the CIVS consists of a number of subroutines. The overall structure of the system is illustrated in Figure 6. The system consists of five portions: VCR/slide, voice, Graphics editor, test, and extra routines. Two interface have been made for VCR/slide and voice control. A number of editor and subroutine were also developed with this system.

Authoring an interactive program for the CIVS system requires five hardware components (see Figure 2):

1. IBM XT Computer:
   - 640K RAM
   - two-360K floppy disk drives
   - a hard disk drive
   - keyboard
   - Enhance Graphic Adapter
   - Monochrome Graphic Adapter
     (the ET Chinese system uses video memory in the MGA)
   - interactive video control interface
   - ET 2416 Chinese ROM card

2. Voice Processor:
   - dynamic microphone
   - voice processor interface
Chinese Interactive Video System

<table>
<thead>
<tr>
<th>VCR/Slide</th>
<th>Voice</th>
<th>Graphics Editor</th>
<th>TEST</th>
<th>Extra Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Software</td>
<td>Interface</td>
<td>Software</td>
<td>Character Set</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>VCR/Slide Tester (VCRTEST)</td>
<td>VCR/Slide Tester (VCRTEST)</td>
<td>Voice Voice</td>
<td>Voice</td>
<td>Voice</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Play</td>
<td>VCR/Slide</td>
<td>VCR/Slide Tester (VCRTEST)</td>
<td>Voice Voice</td>
<td>Voice</td>
</tr>
<tr>
<td>Forward</td>
<td>VCRFWDTO(1)</td>
<td>Record</td>
<td>VCRREWDTO(1)</td>
<td>MemoToRAM</td>
</tr>
<tr>
<td>Pause</td>
<td>VCRPAUSE</td>
<td>RAMToMemo</td>
<td>Audio/ON</td>
<td>VCRAUDTO(1)</td>
</tr>
<tr>
<td>Audio/ON</td>
<td>VCRAUDTO(1)</td>
<td>Save</td>
<td>Audio/DFF</td>
<td>SLIDE(2)</td>
</tr>
<tr>
<td>Slide/ON</td>
<td>VCRAUDTO(1)</td>
<td>Save</td>
<td>Audio/DFF</td>
<td>SLIDE(2)</td>
</tr>
<tr>
<td>Slide/DFF</td>
<td>VCRAUDTO(1)</td>
<td>Save</td>
<td>Audio/DFF</td>
<td>SLIDE(2)</td>
</tr>
<tr>
<td>Slide/FWD</td>
<td>VCRAUDTO(1)</td>
<td>Save</td>
<td>Audio/DFF</td>
<td>SLIDE(2)</td>
</tr>
<tr>
<td>Slide/BCK</td>
<td>VCRAUDTO(1)</td>
<td>Save</td>
<td>Audio/DFF</td>
<td>SLIDE(2)</td>
</tr>
</tbody>
</table>

Remark:
- Letters in the parentheses are filenames.
- Numbers in the parentheses are the number of arguments.

Figure 6. Overall Structure of the CIVS System
3. Video:
   . video cassette recorder (SANYO-VCR 4600)
   . video camera

4. Monitor:
   . EGA monitor (TVM-MD11)
   . TV Mouse (see Figure 2 & 15)

5. Slide:
   . slide projector (KODAK 270)
   . photo camera

The software required by the CIVS and those developed in this study is:

1. MS DOS (Microsoft Disk Operating System)
2. Chinese Operating System (ET-2416)
3. Turbo PASCAL 3.0
4. BASIC interpreter: BASICA
5. The CIVS authoring system

From these components, a number of different kinds of interaction environment may be created. Broadly speaking, they may be categorized into either of two types: single-user and multiple-user.

**Voice Processor**

The CIVS system promises two-way communication with the computer, which uses voice input and voice output in an educational environment.
The major goal of the voice processor in the CIVS system is to provide voices for short periods of time. As many as seven segments of 4-second voices can be saved into a voice file on the disk. Voices provided by the voice processor can be accessed quickly and these recordings can serve as feedback for students' learning. Although the voice explanation and feedback can be provided from VCR, such a system would probably be too slow to be practical. The voice processor can offer an accurate and real-time recording, and although it can be no more than 4 seconds in length, it should be sufficient time for a vast number of circumstances. Using the VP-1000 IC chip, the voice record/playback interface allows the IBM XT/AT computer to perform the voice record and playback functions.

An interface card that plugs into the I/O channel of an IBM PC is used to transmit voice data between the computer and the voice processor. This controller communicates with the CPU via a 62-pin connector that plugs into the PC bus. The data buffer and address-decoding circuits that comprise the interface is then placed into the PC bus for voice-data transmission.

The block diagram of the voice processor is illustrated in Figure 7. The arrows in the diagram show the path through which the voice signal passes. Single-lined arrows indicate the way the voice is saved onto the disk, while double-lined arrows indicate the way the voice is played back to the speaker.
The VP-1000 speech processor is a CMOS LSI device for speech analysis. This chip can be connected to an external RAM in order to construct a real-time recording and playback circuitry. The chip can produce a very high-quality voice at the sampling rate of 24K to 32K bps (bit per second), as well as an acceptable voice when the sampling rate is lowered down to 12K bps (for a detailed description of VP-1000, see Appendix A). In order to connect the voice processor to the computer, a voice interface was designed. Figure 8 shows the circuitry of the voice-reproduction interface. A static 8K RAM (6264) serves as a buffer, while a number of TTLs control the route of the voice data between programmable peripheral interface PPI-8255 and voice processor VP-1000.
Figure 8. Voice Processor Circuitry
The table below illustrates the function of each chip in the voice-processing interface:

Table 4. List of IC Description of Voice Processor

<table>
<thead>
<tr>
<th>IC Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPI-8255</td>
<td>programmable peripheral interface</td>
</tr>
<tr>
<td>VP -1000</td>
<td>CMOS LSI for speech processor</td>
</tr>
<tr>
<td>SRAM-6264</td>
<td>64K bits static RAM</td>
</tr>
<tr>
<td>74LS00</td>
<td>quadruple 2-input positive-WAND gates</td>
</tr>
<tr>
<td>74LS30</td>
<td>8-input positive-NAND gates</td>
</tr>
<tr>
<td>74LS139</td>
<td>dual 2-line to 4-line decoders</td>
</tr>
<tr>
<td>74LS157</td>
<td>quadruple 2-line to 1-line data selector</td>
</tr>
<tr>
<td>74LS245</td>
<td>octal bus transceivers with 3-state outputs</td>
</tr>
</tbody>
</table>

The Intel-8255 is a general-purpose programmable I/O device designed for use with Intel microprocessors such as the Intel 8088. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and is used in 3 major modes of operation. In the first mode (Mode 0), in which the voice processor is used, each group of 12 I/O pins may be programmed in sets of 4 to be input or output. The statements below briefly describe the principles of the interface circuitry.

1. I/O address and port definition

   DATA DIRECTION SELECT
   ADDRESS : 3E7  <= H90  H80
   PORT A : 3E4   INPUT  OUTPUT
   PORT B : 3E5   OUTPUT OUTPUT
   PORT C : 3E6   OUTPUT OUTPUT
2. In Figure 9, the pins of 5, 6, 7 of port C are used to control the PLAYBACK or RECORD operations.

<table>
<thead>
<tr>
<th>PC7</th>
<th>PC6</th>
<th>PC5</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>RECORD</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>PLAYBACK</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

If not selected, the level of bits of the data port are "HIGH." In order to operate the RECORD function, it is necessary to send a low pulse to the pin labeled "R" in the VP-1000 at Figure 8. Table 5 shows the function table of the TTL SN74LS139:

Table 5. Function Table of SN74LS139

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>B</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>H</td>
<td>X</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

|=======|=======|=======|     |     |     |     |
Thus, the order of the commands for voice recording are:

- \text{OUT \ 3E7, \ 90 (SET 8255 DATA PORT DIRECTION)}
- \text{OUT \ 3E6, \ 80 (SET G LOW ENABLE, A AS LOW, \\ & B AS HIGH)}
- \text{Delay \ (SET Y2 AS LOW)}
- \text{OUT \ 3E6, \ CO (SET G AS HIGH CHIP DISABLE)}

and the order of the commands for voice playback are:

- \text{OUT \ 3E7, \ 90 (SET 8255 DATA PORT DIRECTION)}
- \text{OUT \ 3E6, \ AO (SET G LOW ENABLE, BOTH A \\ & B AS HIGH)}
- \text{Delay \ (SET Y3 AS LOW)}
- \text{OUT \ 3E6, \ CO (SET G AS HIGH CHIP DISABLE)}

3. The RAM-6264 is a storage buffer that temporarily stores voice data between PPI-8255 and VP-1000. During recording, the data are transmitted from VP-1000 to PPI-8255, via RAM-6264; and during playback, the data are sent from PPI-8255 to VP-1000, via RAM-6264. The statements below show the way PPI-8255 is programmed. In Figure 9, port B and bits 0,1,2,3 of port C are the channels for sending the data that are used to control the address of RAM-6264 or VP-1000. Port A is then used to transmit the voice data.

- **Recording**: PPI 8255 <= RAM 6264 (means data from RAM to PPI)
  
  \text{SET \ 3E7 \ as \ H90 \ then:}
  
  PORT A : \text{3E4 IS THE PORT FOR INPUT}
  PORT B : \text{3E5 IS THE PORT FOR OUTPUT}
  PORT C : \text{3E6 IS THE PORT FOR OUTPUT}

- **Playback**: PPI 8255 => RAM 6264 (means data from PPI to RAM)
  
  \text{SET \ 3E7 \ as \ H80 \ then:}
  
  PORT A : \text{3E4 IS THE PORT FOR OUTPUT}
  PORT B : \text{3E5 IS THE PORT FOR OUTPUT}
  PORT C : \text{3E6 IS THE PORT FOR OUTPUT}
4. A group of write-in commands should be executed after each piece of information is sent into the RAM 6264.

   OUT 3E6, data of port C + H60
   OUT 3E5, data of port B
   OUT 3E4, data
   OUT 3E6, data of port C + H40
   OUT 3E6, data of port C + H60

In Appendix C, lines 1100 to 1230 of the program show the sequence of the command. The program in Appendix C is provided for reference purpose, since the BASIC routine is too slow to be practical: it takes 2 minutes and 45 seconds to move the 8K bytes data from buffer to main memory.

5. In order to activate the PPI-8255, the control select address should be set to binary 11111001xx which is between 3E4 to 3E7 in hexadecimal.

   A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
   1 1 1 1 1 1 0 0 1 X X
   _______ _________ _________
       3 E 4-7

6. Figure 9 shows the IBM I/O configuration and pins on each slot of the IBM PC.
Figure 9. IBM I/O Configuration
Voice Editor

The voice editor allows the author to build individual voice segments. A voice file can be created by means of the menu-driven operation, which is an easy method for authors to process their voice data.

In the beginning stages of voice design, it is important to define the location of voice data in the main memory. In total, 64K bytes of memory are needed for the voice space. They consist of 56K (7x8K) bytes for 7 voice segments, and 8K bytes for the control routine, which was written in assembly language.

The 64K byte was located at the address of $8000:0000 in the main memory. After all of the 7 segments of voice data were completely loaded, the 64K byte started from address $8000:0000 to $8000:FFFF then it was saved into the disk as a voice file that could be called back when students used the system. Figure 10 shows the relationship and path of the voice data.

The program listed in Appendix D is the voice editor, which helps authors to build their voice files. Line 340 to line 470 and line 600 to line 750 in Appendix D set data to the memory as assembly-control routines.

In order to hasten the processing speed, the routine was written originally in assembly language by the DEBUG program of DOS utilities. This control routine was located at the beginning of the address $8000:0000. Two assembly routines were developed.
The first 60-byte assembly routine for voice control in E.1 of Appendix E was a routine designed to move the voice data from RAM-6264 to the main memory when voice was recorded. Destination location of the main memory was indicated at address $8000:0030 (see Appendix C, line 500). The destination address (offset) indicated by $8000:003C had been set earlier at high-level language from which the voice subroutine was called.

The second 90-byte assembly routine for voice control in E.2 of Appendix E was a routine to move the voice data from main memory to RAM-6264 when voice is playbacked. The target address (offset) indicated by $8000:00FF (see Appendix C, line 790) had been set earlier at the high level language from which the voice subroutine was called.
Figure 11 shows the menu screen displayed by the voice editor. The program offers six functions for voice: recording, playback, data moving from PPI to RAM, data moving from RAM to PPI, file saving, and file loading.

-------------------------------------------
Voice Editor Menu
1. Record
2. Playback
3. Voice RAM to Main Memory
4. Main Memory to Voice RAM
5. Save (7 voice segments)
6. Load (7 voice segments)

Select = 4
Segment (1--7) = 3
-------------------------------------------

Figure 11. Menu of Voice Editor

A PASCAL routine must be written as needed by the main program (see Appendix I for the PASCAL routine). Run the program VOICEPLY.PAS to play the voice back. The operations of the program are shown below:

-------------------------------------------
Please input the voice data filename you want it to play: _
Please input the segment (1-7, 0 = EXIT): _
-------------------------------------------
VCR/Slide Controller (Interface)

A Sanyo video cassette recorder (VCR) and a KORDA 270 slide projector are essential elements of the CIVS SYSTEM. The KORDA 270 slide projector provides a small screen for slide display, which is especially appropriate for small group or individual learning when the display of static pictures is desired.

A card that plugs into the I/O channel of an IBM PC or compatible computer is used to send the control signal from the computer to the devices to be controlled. The block diagram in Figure 12 indicates the block function of the interface in the CIVS system. Blocks in Figure 12 include voice/VCR/slide control. Figure 13 is the interface circuitry by which the computer controls the operation of the VCR and slide projector.

Table 6. List of IC Description of VCR/Slider Controller

<table>
<thead>
<tr>
<th>IC Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS00</td>
<td>quadruple 2-input positive NAND gates</td>
</tr>
<tr>
<td>74LS04</td>
<td>hex inverters</td>
</tr>
<tr>
<td>74LS08</td>
<td>quadruple 2-input positive-NAND gates</td>
</tr>
<tr>
<td>74LS30</td>
<td>8-input positive-NAND gates</td>
</tr>
<tr>
<td>74LS76</td>
<td>dual J-K flip-flop with preset and clear</td>
</tr>
<tr>
<td>74LS154</td>
<td>4-line to 16-line decoders/demultiplexers</td>
</tr>
<tr>
<td>74LS193</td>
<td>synchronous 4-bit up/down counters</td>
</tr>
</tbody>
</table>
Figure 12. Block Diagram of the Voice/VCR/Slide Control Interface
Figure 13. Circuitry of the VCR/Slide Controller
The basic theory of the operations of the VCR/Slide controller is summarized as follows:

1. In selecting the PPI-8255 inside the VCR/Slide interface, the I/O control select address should be set to binary 11111000xx, which is between 3E0 to 3E3 in hexadecimal.

   A9  A8  A7  A6  A5  A4  A3  A2  A1  A0
   l  l  l  l  1  1  0  0  0  x  x
   3   E          0 - 3

2. Using the PPI-8255 as a peripheral controller, the I/O address and mode definition are as follows:

   DATA DIRECTION SELECTION
   ADDRESS : 3E3 <= H8B  (means set 3E3 as H8B)

   PORT A : 3E0  OUTPUT
   PORT B : 3E1  INPUT
   PORT C : 3E2  INPUT

3. Since I/O address 3E3 was presented as H8B, then port A is defined as an output port; and Ports B and C serve as input ports. Each port and its control functions are described below:

   (1) Port A0 to A3

   A decoder (SN74154) was connected behind port A0 to A3. The 74LS154 is a demultiplexer that decodes 4 binary-coded inputs into one of 16 mutually exclusive outputs. Based upon the data in the port A0 to A3, one of the devices connected to 16 pins of the 74LS154 was selected to act. Table 7 shows the function table of port A0 to A3.
Table 7. Function Table of Port A0 to A3

<table>
<thead>
<tr>
<th>PA3</th>
<th>PA2</th>
<th>PA1</th>
<th>PA0</th>
<th>LS154</th>
<th>out</th>
<th>Control Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>VCR stop</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>VCR play</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>VCR fast forward</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>VCR rewind</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>VCR pause</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>VCR counter reset</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>VCR counter clear</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>Interface counter clear</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>13</td>
<td>Slide backward</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td>Slide forward</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Table 8. Function Table of Port A4 to A7

<table>
<thead>
<tr>
<th>Pins</th>
<th>Levels</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA4</td>
<td>0</td>
<td>AUDIO OFF</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>AUDIO ON</td>
</tr>
<tr>
<td>PA5</td>
<td>0</td>
<td>VIDEO From Computer</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>VIDEO From VCR</td>
</tr>
<tr>
<td>PA6</td>
<td>0</td>
<td>Interface Counter OFF</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Interface Counter ON</td>
</tr>
<tr>
<td>PA7</td>
<td>0</td>
<td>SLIDE LAMP OFF</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SLIDE LAMP ON</td>
</tr>
</tbody>
</table>

(2) Port A4 to A7

A number of optical couplers were used here to control the OFF/ON actions of the devices connected. Table 8 indicates the functions controlled by pins A4 to A7 of port A:
(3) Ports B and C

The 16 pins of port B and port C receive the data via the interface counter, which consists of four LS193 chips. The data in the counter are integers from 0 to 32767 that indicates the logical address of the VCR and by which the controller access as the location desired to display or be performed.

The TTL-compatible clock and the data of the counter are converted to binary format prior to being transmitted to ports B & C or PPI-8255.

A data word protocol employing 8 bits of port A is used to transmit data to the electronic switches. Each bit controls the ON/OFF operation of each device connected at the end of transmission lines. The clock of VCR counter is fed into the four 4-bit counter, 74LS139, which counts the number of clocks from the VCR. The 74139 thus stores the number that indicates the specific address of the VCR tape being played. Figure 14 shows the bit control function of the three ports of PPI-8255 in the VCR/Slide controller:
Figure 14. Bit-control Function of the VCR/Slide Controller
The TV-Mouse in Figure 2 is an adapter for television or video-cassette-record viewing and is used with EGA monitor. With a built-in speaker, the TV-Mouse adapter provides for either computer digital video signals or VCR-analog video signal display selection. The display mode can be selected by the mode-control switch on the TV-Mouse. There are many kinds of video signals that can be fed into the TV-Mouse for the EGA monitor to display. Figure 15 shows the possible connect signal cables to the TV-Mouse.

Figure 15. The Video Signal Cable the TV-Mouse
At the time of development, a slide projector emulator (SPE) was designed as a substitute for the KODAK 270 slide projector. The SPE was used to emulate the operation of the slide projector. Thus, without the real slide projector, the CIVS system still can be tested and monitored. Three inputs connect the emulator and the VCR/Slide interface. Figure 16 shows the circuitry of the SPE. The display-enable controls the ON/OFF of the SPE counter display, which indicates the ON/OFF status of the slide bulb. The lines of up and down control the number of counter display, which represents the forward/backward function of the slide picture. The number of the counter on display represents the number of the slide picture being displayed. Figure 16 shows the circuitry of the SPE.

VCR/Slide Test Program

A VCR/slide test program must be designed to test all the functions of the VCR and the slide projector. The program can control such operations of the VCR as stop, play, fast forward, fast rewind, and pause. In order to control the VCR precisely, additional functions such as counter reset and counter clear were offered by the test program. This test program can also check the operations of the interface for the slide control. The ON/OFF functions of the slide's lamp and slide's forward/backward may also be tested by the program.
Figure 16. The Circuitry of the Slide Projector Emulator (SPE)
The operational behavior of devices are controlled by the data
that are presented at port A (I/O 3E4). These bit-array functions are
summarized below:

01 - STOP
02 - PLAY
03 - FAST FORWARD
04 - FAST REWIND
05 - PAUSE
06 - VCR COUNTER RESET
07 - VCR COUNTER CLEAR
08 - CLEAR INTERFACE COUNTER FOR VCR
10 - VCR AUDIO ON
20 - VCR VIDEO (Otherwise, computer video).
40 - INTERFACE COUNTER FOR VCR ENABLE
80 - SLIDE LAMP ON
OD - SLIDE BACKWARD
OE - SLIDE FORWARD

Appendix F illustrates the source code of the VCR/Slide test program.
This program was written in BASIC language.

The operations may be performed after the VCR/slide test program
is executed. First, the VCR can play between the two address numbers
selected by the author. Second, the slide projector can be controlled
by ON/OFF, and the pictures can be moved forward or backwards. Third,
it is possible for the VCR video signal to be set to off while the VCR
audio is set to on, under this condition it means that the VCR is
producing sound only: This voice can help to explain the lesson while
the monitor is displaying the computer screen.

The program is presented with a simple menu that lists some of
the options available to the user. By selecting the number of the
feature, the user may access to the function that he/she desires.
Two possible menus may be displayed in the VCR/slide test program. Figure 17 shows the first menu, and Figure 18 shows the second menu.

-------------------------------
VCR Function Selection Menu

1. Play
2. Fast forward
3. Fast backward
4. -Pause
5. -Stop
6. -Video mode selection (computer/VCR)
7. -Audio (on/off)
8. Slide (on/off)
9. Slide forward
10. Slide backward
11. Exit

Function Selection (1-11)?

-------------------------------
Figure 17. VCR/Slide Function Selection Menu

-------------------------------

VCR Function Testing

press P for VCR Pause
         C for Video from computer
         V for Video from VTR
         A for Audio (on/off)
         S for Slide (on/off)
         F for Slide forward
         B for Slide backward
        ESC for VCR stop

-------------------------------
Figure 18. VCR/Slide Function Testing Menu
The main program of CIVS was written in PASCAL; thus, routines for the VCR/slide control must be written in PASCAL as well. These routines can be executed under the DOS mode. The commands and examples of instructions are as follows:

1. C> VCRINIT (VCR rewinds to the beginning, and resets the VCR counter)
2. C> VCRFWDTO 50 (VCR fast forwards to address 50)
3. C> VCRREWto 20 (VCR fast rewinds to address 20)
4. C> VCRPLAY 60 100 (VCR plays from address 60 to 100)
5. C> SLIDE FORWARD 2 (moves slide forward for 2 pictures)

These five commands are supported by five command files: VCRINIT.COM, VCRFWDTO.COM, VCRREWTO.COM, VCRPLAY.COM, and SLIDE.COM. Source codes for these five programs are listed in Appendix J.

Courseware Graphics Editor

Digitals' Courseware Authoring System (CAS) is currently used to develop the instructional materials on the VAX supermini-computer at Iowa State University. This system provides a complete authoring environment for Computer Based Education (CBE) development. The novice can program a sophisticated CBE lesson with little prior experience using the easy-to-use Digital Authoring Language.

Concerning such a powerful authoring tool, this study suggests developing the Courseware Graphics Editor (CGE) system with an IBM-compatible personal computer and conduct a practical design of a Chinese Courseware Graphics Editor in the IBM micro-computer.
The use of IBM-compatible PCs, instead of mini-computers, will allow the use of numerous utilities and packages that are readily available for IBM-compatible machines. Furthermore, it allows many more users to access this teaching aid. The development of CBE authoring tools on the IBM PC will not only make it feasible for many operations to create instructional programs, but will also enhance the effects by exploiting the educational programs.

In this study, the current CGE on VAX computer was first studied. According to the function and the operation of the Digitals' CGE system, a Chinese Courseware Graphics Editor (CCGE) was then designed as a subsystem of the CIVS in the IBM-compatible computer. Using a Chinese-character generating package (ET Chinese system), this graphics editor not only can display English but can also display colored and sized Chinese words. Because of the development of this Chinese Courseware Graphics Editor, it is much more feasible to implement CBE in Chinese-speaking countries.

The CCGE of the CIVS system allows easy input and editing of graphics displays of teaching frames. The editor is programmed in such a way that a single keypress will draw a circle, draw a line, draw a box, or paint any color inside a closed area. Choosing colors and designing various attributes for these graphics requires single key presses also. Text display, in either English or Chinese, can be done with the aid of the graphics editor. The text in the CCGE can be sized, colored, and displayed in a variety of types by manipulating the programmed commands and parameters. The CCGE editing screen is
illustrated in Figure 19.

A user's guide was documented for the Chinese courseware development program. The CCGE user's guide is reproduced in Appendix B and a program from the CCGE is listed in Appendix G. Two utilities, CGER and CGEW, were designed with the CCGE program. These two subprograms are very helpful when the author wants to rescue the last drawing in case he/she accidentally exited out the CCGE program without saving the screen file.

Figure 19. CCGE Editing Screen
The graphics display method in the CCGE was similar to that of Digital's CGE. Functioning as the ReGIS, or Remote Graphics Instruction Set in the Digital system, the ET codes are the low-level graphics codes understood by the PC DOS. PASCAL commands can also be generated by the graphics editor, which can save the screen display in a slide file. Consisting of PASCAL instructions, these slide files can be integrated into a CBE lesson by using the PASCAL language. Relationships between CCGE, PASCAL, and ET codes are shown in Figure 20.

Figure 20. Relationships Between CCGE, PASCAL, and ET Graphic Codes
The ET system provides capability for users to create their character sets. These character sets can be displayed under the ET Chinese system. The versatility of the character sets is a powerful advantage for authors, enabling them to develop any kind of character shapes, even characters of foreign languages. The creating of character sets are supported by the command file, which is named ETWORD.COM. Type ETWORD under the ET system in order to use the character sets editing function. The ETWORD screen is shown in Figure 21.

---

Figure 21. Character Sets Editing Screen

---
The ability to write a valid test is an art unto itself. The CIVS system allows authors to use a video or voice segment in order to set up a question. However, the presentation of the visual question is not easy work. Authors should start with question information using a rough draft and then develop a question flow chart. Finally, the sequence of questions can be assembled into their final script. The CIVS authoring system provides a script editor, a lesson script builder, for the author to arrange the logic sequence of file presentation.

Another method of answer-processing is to compile the questions by the "test editor." Questions for the test editor of the CIVS system are created by an easy-to-use menu. With the cursor positioned for question presentation, authors embed an answer-processing command with the selection of the driven menu. After selecting either answer processing for query, answer position, correct, OK if, or wrong answer responses, users are prompted to enter data, as appropriate, concerning the correct, incorrect, and unexpected answers with the labels of corresponding feedback frames. Answers are judged against a set of correct and incorrect answers.

Feedback frames may be added to question frames or may be new screens. After receiving feedback, students are branched to the next screen in the case of a correct answer, or back to the question in the
case of an incorrect answer. The subtitles and functions of the test
editor are described below:

Query - to input the question and define the position of display.
Answer - to choose the screen position for the students' answer.
Correct - to input the correct answers, and key in the feedback
frame for the correct answer.

OKif - to judge conditionally correct answers. If students'
answers contain any portion of the correct answer,
they are considered to be correct. The feedback frame
for OKif is also given here.

Help - to process help messages. The messages will be given
when students type "Help" for some sort of assistance.

Wrong - to define the frequency of errors and feedback messages.

Totally, twenty questions can be processed by this test editor. The
data are saved in the file type ITM. Figure 22 shows one of the Query
input screens.

|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| Clear Area | Query | Answer | Correct | OK if | Help | Wrong |

UpLeft: x=10y=10 At: x=12y=12 ForeColor: 11

DownRight: x=60y=18 BackColor: 0

What is the smallest odd number
that is greater than 1000?

11

Figure 22. Query Input Screen
Integrating the Authoring System

The final phase of the development process was integrating the authoring system which involved the integration of all subprograms designed during the previous phases. The learner-controlled strategy was built into the presentation system. According to the files indicated by the user, the system automatically links screens into lessons through the lesson builder. The system prompts users when required screen labels are not found.

The authoring system comes complete with driver and executive programs for the course materials. The author defines the sequence of the presentation in a script that was coded by the script editor inside the CIVS authoring system. These programs run the presentation for the student. The first menu coming up from the CIVS authoring system is shown in Figure 23.

CHINESE INTERACTIVE VIDEO SYSTEM (CIVS)
Authoring-System Main Menu

(1) courseware graphic editor
(2) character set editor
(3) test editor
(4) voice editor
(5) lesson and script editor
(6) execute the lesson

Select > 1

Figure 23. Main Menu of CIVS Authoring System
The functions and program from 1 through 4 have all been discussed previously. Option 5 is a subroutine for authors to build their script, which tells the computer the sequence of presentation. The program provides 16 commands, as follows:

2. Test (question #): presents a question from the number indicated.
4. Voice (segment #): plays the voice segment, as indicated.
5. VCRinit: initiates the VCR, rewinds the VCR to the beginning, and clears the VCR counter.
6. VCRFWDTO (endaddress): VCR fast forwards to the address indicated.
7. VCRREWTO (endaddress): VCR fast rewinds to the address indicated.
8. VCRplay (start, end): VCR plays from start address to end address.
9. VCRaudio (start, end): VCR plays audio only.
10. VCRpause: VCR pauses, screen frozen.
11. Slide (paral, para2): Slide operation command, the two parameters can be one of the combination as follows.

<table>
<thead>
<tr>
<th>Parameter1</th>
<th>Parameter2</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td></td>
</tr>
<tr>
<td>off</td>
<td>#</td>
</tr>
<tr>
<td>Forward</td>
<td>#</td>
</tr>
<tr>
<td>Backward</td>
<td>#</td>
</tr>
</tbody>
</table>
12. ClrScreen: Clears the screen
13. Delay (seconds): Delays for the # of seconds indicated.
15. WaitReturn: Waits until the Return key is pressed.
<table>
<thead>
<tr>
<th>Lesson Script</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 VCRinit</td>
<td>1 CEFile</td>
</tr>
<tr>
<td>2 ClrScreen</td>
<td>2 ShowTest</td>
</tr>
<tr>
<td>3 CEFile</td>
<td>3 LoadVoice</td>
</tr>
<tr>
<td>4 WaitAnyKey</td>
<td>4 PlayVoice</td>
</tr>
<tr>
<td></td>
<td>5 VCRinit</td>
</tr>
<tr>
<td></td>
<td>6 VCRMUTD</td>
</tr>
<tr>
<td></td>
<td>7 VCRREMTD</td>
</tr>
<tr>
<td></td>
<td>8 VCRplay</td>
</tr>
<tr>
<td></td>
<td>9 VCREadio</td>
</tr>
<tr>
<td></td>
<td>10 VCRpause</td>
</tr>
<tr>
<td></td>
<td>11 ShowSlide</td>
</tr>
<tr>
<td></td>
<td>12 ClrScreen</td>
</tr>
<tr>
<td></td>
<td>13 Delay</td>
</tr>
<tr>
<td></td>
<td>14 WaitAnyKey</td>
</tr>
<tr>
<td></td>
<td>15 WaitReturn</td>
</tr>
<tr>
<td></td>
<td>16 Beep</td>
</tr>
<tr>
<td></td>
<td>LoadFile</td>
</tr>
<tr>
<td></td>
<td>SaveFile</td>
</tr>
<tr>
<td></td>
<td>InsertLine</td>
</tr>
<tr>
<td></td>
<td>DeleteLine</td>
</tr>
</tbody>
</table>

Pick a script command: 1 - 16, L, S, I, D or "E" to End Script Editing

>8 From Address: 20 To Address: 40

Figure 24. Screen of Script Building

The overall environment of the CIVS and the process of courseware production is summarized in Figure 25. The flow chart illustrates the steps that must be taken to develop interactive video material on the CIVS system. Double-lined arrows indicate the steps of the data file creation. By using the utilities indicated in the block, users can create data files or the material required by the lesson presentation program. Single-lined arrows indicate the file generated, which would eventually be integrated by the lesson presenter, the CCAS subprogram (source code see Appendix H).
Figure 25. Flowchart of Courseware Production
CHAPTER IV. FINDING AND DISCUSSIONS

This chapter is devoted to reporting the results of the study. The problems were to research, develop, validate, and investigate interactive video interfaces between men and computers -- i.e., the software, and between the multi-media and computers -- i.e., the hardware. Basically, interface-development was conducted in two phases. Part I was the research and development effort needed to create the interface. Part II was the hardware testing and software design.

At an early stage of this project; it was decided that the CAI style to be adopted would be that of the tutorial, the student being asked to undertake observation of materials presented by either a computer or VCR. The intent of the CAI software then was to illicit responses from the students by questioning and providing feedback as necessary. The controlling routines, as well as the design of the authoring system, were based on this model. These routines can be categorized into five groups: VCR/slide, voice, CGE, test, and extra program control routines (see Figure 6).

The differences between the supporting program and the controlling routines in Figure 6 are that the supporting program can be executed alone as it is completely self-contained, and that the controlling routines are procedures that are a part of a program in the large system and need to be executed inside the system program.
The approach taken was based on previous CAL-development experience with materials utilizing micro-computers, involving facilities normally associated with author-languages within the high-level computer language, TURBO PASCAL.

The supporting software was designed for the teacher who knows PASCAL, in case he/she chooses not to use the CIVS system to create courseware.

Throughout the development of the CIVS system, friendliness, or ease-of-use, is particularly considered, not only from the educational perspective, but also from the design perspective. Four system-design considerations are worth mentioning here.

Firstly, the authoring system and all of the software in the system were very carefully designed so as to prevent the interruption of courseware production by an undesired error operation. Normally, the system will give a warning message to the author when an error occurs.

Secondly, colors of each screen were carefully selected, and screens were concisely arranged in order to offer more free-screen space and an easy-to-understand menu for the author.

Thirdly, the names of the subroutines are based on the functions that the subprograms perform, such as VCRFWDTO, SLIDE ON, WaitAnyKey and so on.

Fourthly, most of the hardware-controlling routines can be executed under the DOS mode, which provides an easy way of hardware
testing.

Inside the CIVS system, a voice can be produced by two overlapping media. The voice can be played back either by the voice processor or by the VCR while the VCR is being played for audio only. When deciding whether such a voice should be created by the voice processor or by the VCR, several questions should be answered. The most important is, "Is the length of the voice longer than four seconds?" If the answer to this is "Yes," then the voice should be created by the VCR; but if the answer is "No," the voice processor is the better choice.

Generally speaking, the voice data created by the voice processor will occupy 8K bytes of memory. Although they are fast-accessed and played back, they take up much memory space. On the other hand, it takes a while for the VCR to search and find the location of the voice data in the tape.

Recent direct memory access (DMA) technology includes some advanced voice processors, that do not have time-length limitations. Nevertheless, they still require much memory space.

Regarding the authoring system, Kearsley (1984) indicated that one of the most serious problems with the use of authoring tools is the tendency to view them as complete solutions to the quick and easy development of computer-based instruction (CBI) courseware; but while the use of authoring tools does make the development process more efficient, it is no substitute for good design. High-quality CBI courseware takes talent and considerable time to develop; authoring
tools are no panacea.

However, the most significant limitation of authoring tools, according to Keaseley, were that they pre-fixed the lesson presentation modules. In short, to achieve the full potential of interactive instruction, the author must still use a general-purpose programming language.

Functioning along the lines of Digitals' Courseware Graphics Editor (CGE), the Chinese Courseware Graphic Editor (CCGE) in the CIVS system is a successful design that allows easy input and editing of graphics displays of teaching frames. With the aid of the graphics editor, the author can create a screen with lines, circles, boxes, and arcs. Text display, in either English or Chinese, can be obtained by this CCGE editor. For each object, the user may determine various attributes associated with the object, including size, color, ratio, etc. This on-line editing requires no programming background in lesson-frame production.

Subcommands in the CCGE offer an easy way of screen and file management. Functions of these commands include file loading, file saving, object finding, object moving, etc. In order to create more interactive material, additional functions, such as on-screen PASCAL instruction coding, is hoped to be developed. The slide files (or slide screens) created by the CCGE program now can only be presented in sequential order. If the CCGE can allow the author to input the PASCAL instruction at the same time as he or she places the graphics objects, then the flow of the file presentation can be branched to
anywhere. This additional function would enhance the flexibility of the screen display.

Recent developments in optical disc-storage technology provide many new opportunities for the use of images as an educational resource. The images that are produced by the image-production unit can be fabricated and then committed to storage for later use. There is a variety of different ways in which a video system may be controlled by its user. It is likely that more sophisticated types of workstations will need more of the input/output equipment. Some of the common end-user control methods are listed below:

1. use of a touch-screen system
2. use of a mouse, tracker-ball, or joystick.
3. use of a voice-based system -- these involve interfacing a speech-recognition unit.

Multi-media interfaces to video systems form an important class of display and control facilities. Interfaces of this type may be produced by combining video images from different sources with computer-generated material. Sound screens may also be incorporated and display screens can be made highly creative through the incorporation of a touch-screen system. Systems of this type are extremely useful in the context of multi-media display since they can be used to provide a variety of different stimulating and informative effects.

The fabrication and use of images, both for display and interaction, are processes that play an important supportive role within most training and learning systems.
To provide a general idea of the cost of the system's hardware, the estimated cost of each item is listed below:

1. The computer system (IBM XT): $1,200.00
   - main board with 640K RAM
   - dual 360K floppy disk drive
   - standard key-board
   - 20M harddisk drive
   - Disk Operating System (DOS)

2. The MD 11 multi-sync monitor: $900.00
   - with TV Mouse
   - EGA adapter

3. The video cassette recorder: $300.00

4. The KODAR 340 slide projector: $300.00

5. The ET Chinese System: $300.00
   - ET DOS
   - Chinese word-pattern ROM card

6. The voice process interface: $100.00

7. The VCR/Slide controller: $200.00

-------------------------
Total $ 3,300.00

Note: (1) The above prices are current estimations in the Taiwan market.
(2) The software developed in this project is not included in the cost estimation.
CHAPTER V. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

This chapter summarizes the results of previous chapters and draws conclusions from the research findings. Several recommendation for further-studies are proposed.

Summary of the Research

Interactive video is the combination of computing and video technologies and teaching techniques that have been developed using technologies that are just as valid, individually as when linked. Conventional computer-assisted instruction systems do have a number of significant limitations. For example, they are basically static in nature, and graphic screen transmitting or processing can be quite time-consuming. In the interactive video system, the computer can be used to perform simulations and carry out interactive dialogues while the video can illustrate with moving pictures, sound, and animation.

But no matter how complex the hardware is, the production system used is an extremely critical component in determining how technology will affect education in the future. Poor computer-production systems, leading to poor learning products, can have a deleterious effect on all aspects of learning. The production system is one of the major keys to effective future use of the computer in education.

The CIVS system was the first step toward combining the VCR,
slide projector, voice processor, authoring system, and Chinese
display capability into an interactive video.

This interactive package-development project has been designed to
simplify the tasks associated with preparing programs for the control
of interactive video tape equipment. More particularly, it has been
produced as a tool to facilitate the generation of Chinese
interactive-video instructional materials.

The CIVS system consists of two interfaces and a number of
program modules. Included among these are the following:

**Interface** - (1) the VCR/Slide-controller

(2) the voice-processor

**Program** - (1) the VCR/Slide-control program

(2) the voice-editor program

(3) the courseware-graphics editor
   (and the character-sets editor)

(4) the test-editing program.

The VCR/Slide-control program provides a number of subroutines
with which the user can test the VCR/Slide-projector to make user that
the VCR/Slide-projector are functioning properly.

The instructional options are organized through the use of
"editors," which are given the form of screen-displayed menus. There
is a voice editor, a graphics editor, a character sets editor, a test
editor, and a lesson script editor. A function key displays a help
screen for current editor. The editors themselves each contain
extensive options.
The voice editor helps the author to organize the voice data and data-file; in total, seven voice segments can be saved in a voice file, and each 4-second voice segment takes 8K (8000) bytes of memory space. The courseware graphics editor provides a menu with which users can compile their lesson screens. The screen can include color graphic and color English/Chinese text. A powerful character-sets editor that was developed by the ET company allows the user to define an unlimited number of special symbols or character sets, each character consisting of one to five key assignments. All characters or symbols so defined are sizeable and movable in the Chinese courseware-graphics editor (CCGE). The test editor enables the author to create and update test items. These test items can be retrieved by the presentation routine when students use this learning material. Finally, the script editor provides the menu by which the author can build the script that will ultimately form the final instructional program.

CAI has been introduced to Taiwan, the Republic of China, for almost fifteen years. However, it is still at the stage of early development. The most difficult aspect indicated by previous studies in Taiwan is that the complexity of Chinese input/output (I/O) restricts CAI research and implementation. As state earlier that computers are designed for spelling language such as English, not for a symbolic language such as Chinese. The Chinese Interactive Video System (CIVS) developed in this project provides numerous ways for authors to display their Chinese words. Of the two major video format
in the CIVS, slide and video tape, the slide projector has the ability to display any form of pictures or text as a still frame for any period of time, and is therefore suitable for use where students may be asked to comment on their observations. On the other hand, the video tape can be used to show video materials incorporating audio-visual frames in Chinese, both text and sound. Designed with the ability to display the color Chinese text on the computer screen, the CIVS system does help to solve Chinese output problems that exist in Chinese computer-assisted instruction.

Furthermore, a series of editors by means of screen editing and offering several kinds of Chinese input methods do help the authors to input Chinese words without difficulty.

Besides helping to solve the Chinese CAI I/O problems, this project has also created a tool for CAI courseware development. Previous studies in Taiwan also pointed out that the difficulty for CAI courseware development in Taiwan is the shortage of manpower in this critical area. At schools in Taiwan, there are not many researchers who devote themselves to CAI research. The manpower limitations in Taiwan are further aggravated by the lack of authoring tools. The principal advantage of this system is its obvious ease of use to create Chinese CAI courseware. The CIVS system utilizes a simple-to-follow prompting format that enables the designer to create lessons without having a strong programming background.

The purpose of this project has been devoted to explore the
possibilities of interactive video technology for use in the multimedia teaching system in education in Taiwan. It may not have yet reached its ideal or perfect state, however, it had demonstrated a prototype which is operational. Moreover, additional studies need to be conducted before the Chinese interactive video can be implemented into real class settings on a large scale basis. It is highly hoped that the development of the CIVS system may constitute a small step forward toward implementing computer-assisted instruction in Taiwan, R.O.C.

Conclusions

After the CIVS system was developed, in consultations with a panel of experts from both industry and academic departments, it was agreed that the model and approaches were feasible and workable for interactive video in education in Chinese speaking countries. Obviously, the idea and the model developed in this project should help to overcome some problems indicated by previous studies regarding the implementation of computer-assisted instruction in Taiwan.

As mentioned in Chapter 1, the major problems encountered by courseware makers were: (1) difficulty in displaying high-fidelity pictures, (2) the difficulty in achieving animation screen design, (3) dullness of the computer music, (4) necessity of soniferous explanations, and (5) lack of authoring tools. With the development of the CIVS system, some of these problems can be solved.
Some other conclusions can be drawn by reviewing the outcome of this project:

1. The CIVS authoring system provides a complete authoring environment for computer-based video education (CBE) courseware development. The novice can program a sophisticated interactive CBE lesson with little prior experience providing that one has content expertise. The designer can experiment with the design and flow of the lesson before the lesson is coded, using a variety of graphics and hardware-controlling routines.

2. Since the CIVS system is initially designed for the use of Chinese in interactive video, some additional functions have not yet been included in this project, such as image-processing technology and mouse-input devices. It is expected that these two functions will be developed and will enhance branching and interactive capabilities, and allow the user to specify branching based upon single or multiple responses.

3. Besides a well-designed material-production system, a major responsibility of courseware production is the qualified teacher or author. The best teachers have adequate knowledge of learning theory, are capable of making very powerful predictions about effective learning, and have insight into how students will behave or react to given stimuli in a
structured situation. The full video production team also requires actor, cameraman, photographers, video tape editors, graphics designers, and systems programers. Producing an interactive video material is clearly not the same as writing an conventional CAI program.

(4) Video discs do several things that video tapes cannot. Though video discs are superior to video tapes in many ways, video tapes will nevertheless be more appropriate in some cases. For example, a teacher can reproduce or create a video tape with less equipment than it is possible to create a video disc.

In reviewing this project, it seems obvious that the addition of the video tape template was an invaluable element in creating an effective design. Hardware, software, and instructional strategies are linked and are interdependent. The beauty and benefit of the concept are the minimal outlay of time and money required when compared with the benefits it provides in terms of courseware production.

**Recommendations for Further Studies**

According to the research results and findings, the following recommendations for further studies are proposed:

(1) A field-test for the CIVS system has not been conducted;
therefore, it is recommended that field testing should be carried out in Chinese-speaking countries, in order to collect information that can be used to improve the system developed in this study.

(2) The Chinese Courseware Graphics Editor (CCGE) is a powerful tool in instructional frame-design. It translates the graphics and text screen into PASCAL instructions. Though it is helpful, it helps only the author who understands PASCAL language, should the author choose not to use the authoring system to design all of the lesson. It is suggested that a courseware graphics editor with the abilities to create BASIC, C, PASCAL, or dBASE-III instructions should be included in order to help those programmers whose knowledge is limited to any one of those programming languages.

(3) Using computer-image technology in education is inevitable in the foreseeable future. A variety of image technologies may be employed to improve the display of high-fidelity pictures and the interaction of lessons. For example, a superimposition image enables the mixing of microcomputer and video signals so that they may be displayed together on the same screen. For sophisticated types of video display and interaction, it is recommended that image technology be included in further studies.

(4) As continuation of this work, one might include a video disc
in the system. Furthermore, using the video camera or scanners as video signal input devices might be considered.

(5) The quality of the still-frame from the two-magnetic-head VCR system does not meet the requirement of video instruction. It is suggested that a four-magnetic-head VCR system is preferable, in order to provide high-quality freeze-frames.


ACKNOWLEDGEMENT

This work was funded by National Science Council of Taiwan, the Republic of China. I express my deepest gratitude to the institute. Without its support this project would not have been possible.

Many people have been involved in this work. I am very grateful to my major professor, Dr. William D. Wolansky, for his kindly assistance and support during my time at Iowa State University. It was with his encouragement that this research could be completed. A special appreciation is extended to Dr. William G. Miller for his wealth of knowledge, patience, and time. Very special thanks to my committee members, Dr. Donald J. McKay, Dr. Donald H. Schuster, and Dr. Anton J. Netusil for their insight and their guidance. I have learned a great deal from all of them.

I am grateful to Jane, Zing-Fong (Jackie) who helped to type the report and Anne R. Richards proofreading the report.

I have been very fortunate to have many good friends, they helped me through my study and made it joyful.

Finally, I want to pay my endless gratitude and love to my mother, my father, my wife, and my son.
## APPENDIX

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A</td>
<td>Voice Processor VP-1000</td>
<td>101</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Chinese Courseware Graphics Editor (CCGE) Users' Guide</td>
<td>105</td>
</tr>
<tr>
<td>Appendix C</td>
<td>Voice Editor BASIC Program List (I)</td>
<td>135</td>
</tr>
<tr>
<td>Appendix D</td>
<td>Voice Editor BASIC Program List (II)</td>
<td>137</td>
</tr>
<tr>
<td>Appendix E</td>
<td>Voice Process Assembly Routines</td>
<td>140</td>
</tr>
<tr>
<td>Appendix F</td>
<td>VCR &amp; Slide Projector Test Program</td>
<td>143</td>
</tr>
<tr>
<td>Appendix G</td>
<td>Program List Of The Chinese Courseware Graphics Editor (CCGE Program)</td>
<td>146</td>
</tr>
<tr>
<td>Appendix H</td>
<td>Lesson Presentation Program List (CCAS Program)</td>
<td>178</td>
</tr>
<tr>
<td>Appendix I</td>
<td>Voice Playback Program List</td>
<td>185</td>
</tr>
<tr>
<td>Appendix J</td>
<td>VCR and Slide Control Routines</td>
<td>187</td>
</tr>
</tbody>
</table>
Appendix A. Voice Processor VP-1000
Features:
• High quality speech generation.
• Speech analysis and reproduct with external SRAM.
• Speech synthesis with external EPROM or ROM.
• Compatible to Eletech VP-880 voice development system.
• Compatible to VP-1600 speech controller for dividable speech ROM decoding.
• Memory addressable to 32K x 8 bits.
• Single 3V~6V supply voltage with low power consumption.
• Inexpensive RC oscillator.
• Bit rate adjustable from 9.6K to 128K bps.
• Continuous variable slope data modulation (CVSD) technique.

General Description:
The VP-1000 SPEECH PROCESSOR is a CMOS LSI for speech analysis and speech synthesis application. The chip can be connected to external SRAM to construct a real time recording and playback circuitry. For speech synthesis usage, it can be connected to external speech ROM to playback the stored voice data. Encoding (digitizing) of custom words or phrases must be accomplished by the chip manufacturer or alternately by the individuals using Eletech VP-880 voice development system. The VP-880 voice development kit is designed for speech ROM programming which utilizes IBM PC AT/XT as analysis tool. The system will produce very high quality voice output at the sampling rate of 24K to 32K bps as well as an acceptable voice when the sampling rate is lower down to 12K bps.

Application
• Sound recorder with low standby current.
• Sound effect producer.
• Digital announcer for consumer, industrial, security and telecommunication products.

Pin Assignment

Block Diagram

* "IBM" is a registered trade mark of International Business Machines Corporation.
READ:
Output, active low. Active only in "SPEECH SYNTHESIS" mode.

PLAY:
Input, active low. Triggering on this input shall put the chip into "SPEECH SYNTHESIS" mode.

RESET:
Input, active high. When activated, all the internal counters are cleared and the chip is disabled.

ANG AND ÂNG:
Analog signal outputs with opposite phase.

INTERGRATOR:
Output connected to external intergrator to produce envelope waveform.

ENVELOPE:
Input to be connected to external intergrator output.

TD, TD:
Auxiliary outputs for signal modulation.

COMPODATA:
Input feedback signal from the external comparator output.

R1, C1:
Oscillator pins. Use C1 as the input when employing external clock.

CLOCK DRIVER:
Output pin for the generation of negative voltage.

VDD & GND:
+3V ~ +6V power inputs.

FIGURE 1 Sound recorder with memory retention
Absolute Maximum Ratings*

Supply Voltage, $V_{DD} - V_{SS}$ 0 to 7V
Input Voltage, $V_{IN}$ $V_{SS}$ to $V_{DD}$
Operating Temperature, $T_{OP}$ $-10^\circ C$ to $60^\circ C$
Storage Temperature, $T_{ST}$ $-20^\circ C$ to $80^\circ C$

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Electrical Characteristics ($V_{DD} = 5V$, fosc 64KHz, sampling clock = 32K bps, unless otherwise specified.)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limit</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>Supply Voltage</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>$I_{DD}$</td>
<td>Stand-by Current</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>$I_{drive}$</td>
<td>Clock Drive Current</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>$I_{sink}$</td>
<td>Clock Sink Current</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>Input Voltage</td>
<td>3.5</td>
<td>5</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>Low</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$I_{drive}$</td>
<td>Output Current</td>
<td>Drive</td>
<td>3</td>
</tr>
<tr>
<td>$I_{sink}$</td>
<td>Sink</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$T_{reset}$</td>
<td>Reset Pulse</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>$T_{write}$</td>
<td>Write Pulse Width</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>S/N</td>
<td>Signal-to-Quantized Noise</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

TIMING DIAGRAM

---

Pin Name Description

A0 — A14: Address bus output.

RECORD:
Input, active low. Triggering on this input shall put the chip into "SPEECH ANALYSIS" mode.
Appendix B.

Chinese Courseware Graphics Editor (CCGE)

Users' Guide
TABLE OF CONTENTS

1. INTRODUCTION
   1.1 Introduction
   1.2 The Purpose of the CCGE System
   1.3 System Needs
   1.4 Installation
   1.5 Configuring the System
   1.6 Programs on the CCGE System

2. GETTING STARTED
   2.1 Running CCGE
   2.2 The CCGE Screen
   2.3 Move the Cursor
   2.4 Dots
   2.5 Lines
   2.6 Boxes
   2.7 Circles
   2.8 Arcs
   2.9 Fan Drawing
   2.10 The 16x16 Text
   2.11 The 24x24 Chinese Text
   2.12 Command Features
      2.12.1 Clear Screen
      2.12.2 Redraw Screen
      2.12.3 Save Screen & Create Pascal File
      2.12.4 Begin a New Drawing
      2.12.5 Load a Screen from Disk
      2.12.6 Draw Grids
      2.12.7 Find a Location
      2.12.8 Delete a Graphic Object
      2.12.9 Move Graphics
   2.13 Exit CCGE System
   2.14 Rescue Your Graphics
   2.15 The DAT File and PAS File
   2.16 Compile the Pascal File

3. DATA FILE FORMAT AND INPUT FORMAT
   3.1 The Dot Format 'w'
   3.2 The Line, Box, & Painted Box Format 'l'
   3.3 The Circle and Arc Format 'c'
   3.4 The 16x16 Text Format 'e'
   3.5 The 24x24 Text Format 't'
4. BASIC OPERATION
  4.1 Dots Drawing
  4.2 Lines Drawing
  4.3 Boxes Drawing
  4.4 Painted Boxes (Box Filled)
  4.5 Circles
  4.6 Circular Arcs
  4.7 Print 16x16 Texts
  4.8 Print 24x24 Texts
1. INTRODUCTION

1.1 Introduction

The Courseware Graphics Editor (CGE) is a menu-driven tool which is used to produce graphics displays. It is a relatively new technology that gives the teacher the ability to create a graphical representation of some objects or design on what can be described as a television type display of a computer screen.

Users create a display in CGE by placing objects such as dots, boxes, circles, line drawings, or text on the screen at the desired locations. For each object, users may adjust various attributes associated with that object. The attributes include size, color, gap, logic filling, etc. Once users are satisfied with the display, they may write a file that contains TURBO PASCAL instructions. TURBO PASCAL is a computer language system adapted for the IBM computer. It represents a fundamentally different way of thinking. Learning the PASCAL language is much easier than learning other languages. Everything in PASCAL fits into a logical framework.

The idea of designing the Chinese Courseware Graphics Editor (CCGE) was derived from the CGE program on the VAX mini-computer, which creates a file with Digital Authoring Language (DAL) instructions. The DAL is a high-level language for developing instructional lessons on the Digital Equipment Corporation Vax
computer systems. Instead of DAL, this CCGE package creates PASCAL codes for IBM-compatible computers. Besides the colored texts and graphic objects, the CCGE displays colored Chinese words. We hope that with this important development in Chinese and graphics technology, Computer Assisted Instruction in the Chinese-speaking world will enjoy a prosperous future.

1.2 The Purpose of the CCGE system

First of all, one of the major advantages of using the CCGE system is that teachers in China who don't have any previous computer language experience can design their teaching materials with the help of this system. The CCGE creates the PASCAL codes for the teachers while they place the objects on the screen. The graphic frames, or Chinese teaching frames, can be translated to a file or a unit. The teachers then may link all the small units to an integrated lesson.

The other advantage of using the CCGE system is that without the complicated calculations for the screen coordinates, which can be boring and time consuming, the teacher can create their graphic objects easily.

1.3 System Needs

The CCGE system requires some software and hardware equipment. In order to run the CCGE program, users need the following software:

1. The Chinese ET system. With ET-2416 for mono display and ET-2416F for the EGA system,

2. the TURBO PASCAL 3.0 system,
The CCGE system.

They also need the following hardware:

1. An IBM-compatible computer (pc/xt/at),
2. A hard-disk drive,
3. An EGA monitor and an EGA card for a color system and a monochrome monitor and Hercules adapter for a mono system.

The CCGE system can run in both the mono and color modes. For the mono mode, a Hercules display adapter is necessary. For the color mode, an Enhanced Graphic Adapter (EGA) and an ET Chinese pattern card (ET-2416) are necessary. ET is a trademark of a Chinese word-pattern system that offers almost 13,000 Chinese words for both 16x16 and 24x24 dot-matrix word fonts. The 16x16 fonts are stored in a floppy disk and the 24x24 fonts are stored in a hard disk and ROM card.

1.4 Installation

At the beginning, it is necessary to remind the user that if one is using a color system, one should install both the Chinese ET ROM card and the EGA card on any slot of the computer main board. Refer to the EGA menu to set the dip switches. These switches are found on the EGA card and computer main board.

Before installation, it is recommended that the user make a back-up copy of the distribution disks.

1 Type the following at the DOS command line:

prompt $p$g

This will change the DOS command prompt so that it will always display the current working directory.
(2) Type the following command:

    md et

    This will create a subdirectory with the name ET.

(3) Then type the following:

    cd et

    If all has gone well, the user will see the following prompt:

    c:\ET>

(4) Installing ET system files:
    Insert the first ET disk into drive A, then type the following while in the ET directory.

    copy a:*.*

(5) One at a time, replace the first disk with the remaining disk(s) and repeat the copy command in order to complete the installation procedure. The color ET-2416F has six disks and the mono ET-2416 has two disks.

(6) Return to root directory:

    cd \n
    This command will return the user working directory to root directory. the user will see c:\> on the screen, indicating that the current working directory is the root directory.

(7) Go through steps, (2) through (6) again, in order to install TURBO PASCAL in the hard disk. Commands are as follows:

    md tbpascal

    cd tbpascal

    Insert the first TURBO PASCAL 3.0 disk in drive A. Then type the copy command.

    copy a:*.*

    Insert the second TURBO PASCAL 3.0 disk in drive A, then type the copy command again.
(8) Install the CCGE system on to the hard disk at TBPASCAL directory. Insert the CCGE disk in drive A. Before the user copies the CCGE system on to the hard disk, use DIR command to see the directory on the disk. There are at least three command files on the source disk.

```
DIR/W A:
CCGE.COM
CGER.COM
CGEW.COM
```

Make certain that all three files have been created.

```
copy a:*.*
```

All the software needed by the CCGE system has already been installed onto the hard disk.

1.5 Configuring the System

To edit and compile the turbo pascal program, the user will have to run the TINST program, which was copied into the TBPASCAL directory during installation. In most cases, TINST need only be run once—that is, when the user first install turbo pascal. Change the working directory to the TBPASCAL directory by typing:

```
\ cd \tbpascal
```

To run TINST simply type the following command.

```
TINST
```

The TINST provides interactive ways to guide the user through the configuration process. Be sure to set up the screen mode by selecting (S) for the screen configuration. Set up mono for hercules card and color 80x25 for EGA card.
1.6 Programs in the CCGE System.

There are six system programs in CCGE system.

CCGE.COM: This is the CCGE main program; which will help the user to create the graphic screen and generate a CCGE.DAT file. The CCGE.DAT is a file which keeps the figures required by the graphics. Each time one saves the drawing onto a file, the CCGE will create pascal instructions. Both the DAT file and the PAS file are stored according to the name the user will have given for them.

CCGE.PAS: This is a source program of CCGE.

CGER.COM: This command file can display the graphics from DAT files.

CCGE.DAT: Generally the CCGE.DAT is the last drawing before exiting from CCGE mode. Each time when one exits from the CCGE mode to the DOS mode the CCGE.DAT keeps the records of the last graphic objects. Whenever one enters the CCGE mode the old CCGE.DAT will be erased and rewritten.

CGER.PAS: This is the source files of the CGER.COM.

CGEW.COM: This program helps the user to create pascal instructions and a DAT file from CCGE.DAT, which is very helpful when the user wants to rescue the last drawing in case one has accidentally exited the CCGE without saving the file.

CGEW.PAS: This is the source program of CGEW.COM.
2. GETTING STARTED

2.1 Running CCGE

In order to run the program, it is necessary first to make sure that one is in the TBPASCAL directory. If not, please type CD\TBPASCAL. Type CCGE to run the Chinese courseware graphic editor program.

CCGE

Now the user will go into CCGE mode.

2.2 The CCGE Screen

The first screen of the CCGE program is a title frame (see Figure 2-1).

![Figure 2-1: The CCGE starting title screen](image-url)
Press any key, and enter the CCGE editing mode. The right hand side shows the numbered graphic functions. See Figure 2-2.

Figure 2-2: The Chinese Courseware Graphic Editor (CCGE) Editing Screen.

The graphic functions are as follows:

1. Dot
2. Line
3. Box
4. Circle
5. Arc
6. Painted Box
7. Text
8. Exit
The user may select a graphic function by entering a number from 1 to 8. The cross mark on the screen is the cursor. The user may use arrow keys to move the cursor around the screen.

Use Up-arrow for moving up
Down-arrow for moving down
Left-arrow for moving left
Right-arrow for moving right
Home for moving up-left
Rg-Up for moving up-right
End for moving down-left
Pg-Dn for moving down-right

Use Ins & Del key to change the moving pace.
Ins for 1 and 10 points.
Del for 50 and 100 points

Following this procedure to go through the over-all practice, from functions 1 to 8, to all the commands offered by the CCGE system. Start with chapter 2.3 to draw the graphics showing on Figure 2-3.

Figure 2-3 The graphics on the CCGE screen
2.3 Move the Cursor

To move the cursor around the screen, simply press the arrow keys.

- Press the < key, and observe the cursor move left 1 point. The letters X & Y in the down-left corner indicate the coordinates of the location of the cursor at the graphic mode.
- Press the -> key, and observe the cursor move right 1 point.
- Press the Up arrow key to move cursor 1 point up.
- Press the Down arrow key to move cursor down 1 point.
- Press the PgUp key to move the cursor moved up and right 1 point.
- Press the Ins key, and now observe the moving pace changed from 1 to 10 points at the bottom of the screen.
- Try each arrow key again to see the cursor moving at different pace.
- Press the Del key to change the moving pace to 50. Now try the arrow keys again. Each arrow key will move the cursor for 50 points. To change it to 100; just press Del again.

2.4 Dots

- Move the cursor to (87,20) of the screen. The (87,20) stands for x equal to 87 and y equal to 20.
- Select 1 for the dot-drawing function.
- When command is asking for color, enter 15 in order to select bright-white.

Now a dot is hiding behind the cursor, the cross mark. To see the dot, it is necessary to move the cursor away and redraw the dot by pressing the R key. Try it now, please!

- Press the -> key to move the cursor 50 points away from the dot.
Press R to redraw the screen, and now observe the dot at (87,20).

### 2.5 Lines

The cursor now is right at (137,20).

- Move the cursor to (87,70), and suppose that (87,70) is one end of the line.
- Press 2 to select the line drawing function.
- Press the -> key three times to move the cursor to (187,70).
- Press the <RETURN> key to specify (187,70) as second point.
- Enter 14 for the color yellow; then press the <RETURN> key.

Now there is a yellow line from (87,70) to (187,70).

### 2.6 Boxes

- Press the Down key, then press the <- key twice, and put the cursor at (87,120), which is in one corner of the box.
- Select 2 to draw a box.
- Press the Down arrow key, then press the -> key twice to locate cursor at (187,170).
- Press the <RETURN> key to specify the opposite corner here.
- Enter color 13 for a red box.

### 2.7 Circles

- Press the <- key to move cursor 50 dots left. Locate the cursor at (137,220) of the screen.
- select 4 to draw the circle.
- Press the Ins key to change the moving pace to 10 points.
- Press the -> key three times to move cursor to
(167,220), through which the circle will pass.

. Press the <RETURN> key to specify the location.

. The computer will ask for the ratio of x/y. Just type <RETURN> for the default value 1/1, which is a round circle.

. Enter 12 for the light-red color.

**2.8 Arcs**

. Move cursor to the (437,120) position.

. Select 5 for arc drawing function.

. Press the Del key to set the moving pace at 50 points, then move the cursor 50 points left.

. Press <RETURN> for specifying the radius.

. Now the computer will ask for the start and end angles. Here 0 represents the start angle and 180 represents the end angle.

. Press <RETURN> again, for x/y equal to 1/1 which is default value.

. Assign any color number desired.

The half circle will appear just like an arch on the up-left of the screen.

**2.9 Fan Drawing**

The CCGE system offers a special function that will draw a closed arc like a fan. To draw a fan as in Figure 2-3 the user need to specify a minus number for both the start and end angles. The fan shape in Figure 2-3 has a radius of 50, -45 for the start angle, and -135 for end angle. The small one under the big fan has a radius of 10, -225 for a start angle and -315 for an end angle. The exercise for the fan drawing is described below.
. Move the cursor to (137, 310).
. Select function 5 for arc drawing.
. Move the cursor away for 50 points, then type <RETURN>.
. Set the starting angle at -45, and the end angle at -135.
. Type <RETURN> for 1/1 default x/y ratio value.
. Choose any color from 0 to 15.
. Select 5 again to draw a small one.
. Set radius at 10 by moving the cursor 10 points from the center point.
. Set the start angle at -225, and the end angle at -315.
. Press the <RETURN> key for the default x/y ratio.
. Select the same color as chosen before.

2.10 The 16x16 Text

The CCGE displays both Chinese and English for 16x16 and 24x24 fonts. The 16x16 Chinese text takes two bytes and the English characters take only one byte. Thus, on the screen each Chinese word needs twice the space as the English word does. The Chinese text input method is more complicated than the English method. There are nine Chinese text input methods offered by ET Chinese system. Consult the input method from the ET menu which is offered by ET company. The 16x16 text is working at a text mode which is 80 characters per line and 24 characters from top to bottom. The user may define the text color and background color for those 16x16 texts.
The color selection in 16x16 dots matrix text takes two parameters: the first is text color and the second is background color. The default value for background color is zero.

- Press 7 to select the text printing function.
- Move the cursor to the text location of (28,2). The cursor now is moving at a text mode, not a graphic mode; thus the x and y numbers are smaller than that of the graphics. Besides, one will see that the cursor has also changed to a small box (in mono mode, an under-line) instead of the cross mark in the case of the graphic mode.
- Press <RETURN> to specify the location.
- Key in the text. Now it can be a English or Chinese format.
- Select color 13, If the user wants a background color for the text, it is necessary to type in another color number one space after the first color.

Now key in some Chinese text. The 16x16 Chinese text in Fig 2-3 was input as described below. To input Chinese text hold on ALT/CTRL 3 three keys at the same time. If the bottom line shows some Chinese words, this means that the user is in the Chinese phonetic input mode. Now press DE833 five keys for the first word and ;41 three keys for the second word. After completing the input of the Chinese text, press CTRL and ESC to exit the Chinese input mode. Then press the <RETURN> key and select the color.

2.11 The 24x24 Chinese Text

The CCGE also supports a 24x24 Chinese word function. The 24x24 dot matrix gives a more beautiful word font for Chinese. The more dots, the better the Chinese character will appear. This 24x24
Chinese word font may satisfy business needs in Chinese-speaking countries.

The method of input for 24x24 Chinese is exactly the same as for the 16x16 text, because the Chinese internal code is the same. English text is also offered in the 24x24 font, and it is also larger than in the 16x16 English font. The Chinese ET system takes 24x24 as a graphic shape, so one can easily enlarge the word font. To display the Chinese characters in different logic modes, the user can enter the parameter after the color number. The format is given below.

format: color : gap : x/y-ratio : logic :
bytes : (1 or 2): (1): (3): (1):

In this line, color may be 1 to 2 digits; gap should be 1 digit; x/y-ratio should be 2 digits divided by a slash sign; and logic is letters such as r,s,a,o,x,*. Leave a space between each parameter such as "12 2 2/2 r" or "6 1 3/4 *" and be sure only one space is allowed between each parameter. To create the 24x24 Chinese word as in Figure 2-3, the commands are as follows:

. Move cursor to (388,221).
. Select 7 as text printing.
. Select 2 for 24x24 text.
. Key in 24 for the first two digit number.
. Then press Alter+Ctrl+3 to enter Chinese input mode.
. Press ;41 three keys for the second word; then press the <RETURN> key.
Select any color desired, or place some parameters after the color if it is necessary to enlarge it.

2.12 Command Features

The CCGE system offers several one-key commands for screen and files management. Press the <fl> key, and the command help line will shows up on the bottom of the screen. The help line is reproduced below.

```
Clr-scrn Redraw Save Load New-draw Delete Find Grid Move Include
```

The functions are as follows:

- **C**: Clear the Screen
- **R**: Redraw the Screen
- **S**: Save Screen and Create Pascal Instructions
- **L**: Load a Screen from Disk (Restore the Screen)
- **N**: Begin a New Drawing
- **D**: Delete a Graphic Object
- **F**: Find a Location of a Graphic Object
- **G**: Draw Grids on the Screen
- **M**: Move a Graphic Object to a New Location
- **I**: Include a Screen File into Current Screen

2.12.1 Clear Screen

Pressing C will clear the screen. The graphics still exist on memory, so it is possible to recall simply by pressing R for redraw.

2.12.2 Redraw Screen

Redraw the graphic objects by pressing the R key. The redraw routine draws the graphics according to the data in the CCGE.DAT file.
2.12.3 Save Screen and Create Pascal Instructions

By pressing the S key, the CCGE will ask for a file name. With the name, the CCGE will create a DAT file and a PAS file into the disk. Create some graphic objects and save them onto the disk, giving it a name such as TEST.

2.12.4 Begin a New Drawing

By typing the N key, it is possible to delete the old graphics and start a new drawing. Before typing Y, be sure that the current graphics are no long needed. The new drawing function will not file the current graphic object on the screen.

2.12.5 Load a Screen From Disk

The load option will ask for a file name at the time when L is pressed. It will restore the graphics from the disk to the screen. Be sure that both DAT and PAS stored with the name that is assigned on the disk. Use DIR/w at the DOS mode to see what is available on the disk. Then try to restore it. Now the TEST is stored in the disk. Try it.

2.12.6 Draw Grids

Grids are essential when editing a test and so is the graphic screen, which provides the user an over all relation of text location and graphic coordinates. The CCGE offer 4 cycling key strokes to draw the text and graphic grids. The first G key will draw a text grid. The second G key will draw a graphic grid. And the next G key will clear the first text grid. The 4th G stroke will clear the graphic grid. The next G keys will go into another cycle of the same steps.

2.12.7 Find a Location

The CCGE offers a special function with which the user can search for an object from the screen. Relating with the location of the graphic object, it is possible to draw new graphics. With this function, the user doesn't have to spend too much time in moving the cursor to a detailed point related to the location of an object on the screen.

The F routine will search from the first object drawn, then one by one until the user answers 'y' for yes, meaning that one has reached the point needed. Use the up and down arrows to search for the graphic objects. The bottom of the screen will show the detailed description of the object, for one's convenience. It
is helpful to check the format of the data in chapter 3 of this menu. Press the ESC key to exit the F routine.

2.12.8 Delete a Graphic Object

For those graphics one doesn't want, use the D routine to delete it. The procedure is similar to the Find routine. Use the up and down arrow keys to find desired objects; type Y to delete those one doesn't want. Press ESC to leave the delete routine.

2.12.9 Move Graphics

The M command helps one to relocate specific graphics. Type M and use the arrow to search for desired objects, then press Y to specify the object sought. Move the cursor to the new place where the graphic needs to be relocated. Pressing the <RETURN> key will move the graphics to the new location.

2.13 Exit CCGE system

Upon completion of the work, type 8 or the ESC key to exit the CCGE. The prompt C> indicates to the user that the system is returning to the DOS mode.

2.14 Rescue Your Graphics

The CGER and CGEW are designed to rescue the user's screen in case one accidentally exits the CCGE mono without saving the last drawing. Use the CGER command program to replot the last drawing. If it is exactly what is wanted, then use the CGEW file to resave the screen. The steps are listed below:

. Type CGER first, then observe the last drawing.

. Type CGEW; then give a file name. It will create both DAT and PAS files onto the disk.
2.15 The DAT File and the PAS File

The DAT file stores the related data about the graphic objects that was drawn. Each time when the user put a graphic object is recorded on the screen, the graphic record will be kept in this file. According to the record, the CCGE plots or draws the picture. And according to this file, the CCGE also creates the PAS file.

After saving a TEST file during the first practice with CCGE, now the user can see the TEST.DAT by typing:

TYPE TEST.DAT

For the detailed format or parameters see chapter 3. It may also be helpful to take a look at the TEST.PAS program by typing:

TYPE TEST.PAS

2.16 Compile the Pascal File

Each screen graphics that were drawn and saved on to the disk are in a Pascal file. It will have to be compiled to create a command file. The procedures are listed as follows. Assume that the user has created a file 'TEST' in the disk.

(1). Type TURBO to enter the turbo pascal mode.
(2). Press Y to include error message in the turbo mode.
(3). Press W and identify the file name TEST.
(4). Type R to run the program TEST. Now observe see the same screen as was created in the CCGE mode.
(5). To make a command file, it is necessary to redefine the option by typing the 0 key.
(6). Select C for defining compile the file to a command file; then press Q to exit the option routine.
(7). Press C to compile the file and to store the command file onto the disk.
(8). Press Q to exit the turbo mode.
(9). Type DIR/W. You will see TEST.COM in the disk.
(10). Type TEST to run the command file TEST at the DOS mode.
3. DATA FILE FORMAT AND INPUT FORMAT

The CCGE.DAT stores the graphics data that were drawn. There are six kinds of graphic data lines that may be saved in this data file. The first two bytes of the data line keep letters that stand for different types of graphic objects. These six types are listed below.

(1). w : for dot
(2). l : for line, box, and painted box.
(3). c : for circle and arc.
(4). e : for 16x16 text.
(5). t : for 24x24 text.
(6). p : for paint a closed area.

The file listed blow is the TEST.DAT data file which was made in chapter 2.

```
W 87 20 15
1 87 30 187 70 14
1 87 120 187 70 13 b
C 137 220 30 12 0 0
C 137 310 50 11 -450 -1350
c 137 310 10 11 -2250 -3150
1 387 20 487 70 7 bf
C 437 120 50 6 0 1800
t 387 270 14 2 2/2 »
```

Each data line contains a number for color that is three bytes in length. The numbers of color range from 0 to 15. The mono monitor can only display white or black. It is recommended that the user
selects zero for black and 15 for white. The IBM PC specifies an integer number for predefined content of colors.

<table>
<thead>
<tr>
<th>code</th>
<th>color</th>
<th>code</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>black</td>
<td>8</td>
<td>DarkGray</td>
</tr>
<tr>
<td>1</td>
<td>Blue</td>
<td>9</td>
<td>LightBlue</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>10</td>
<td>LightGreen</td>
</tr>
<tr>
<td>3</td>
<td>Cyan</td>
<td>11</td>
<td>LightCyan</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>12</td>
<td>LightRed</td>
</tr>
<tr>
<td>5</td>
<td>Magenta</td>
<td>13</td>
<td>LightMagenta</td>
</tr>
<tr>
<td>6</td>
<td>Brown</td>
<td>14</td>
<td>Yellow</td>
</tr>
<tr>
<td>7</td>
<td>LightGray</td>
<td>15</td>
<td>White</td>
</tr>
</tbody>
</table>

3.1 The Dot Format 'w'

The 'w' format is defined in a command the line that contains 13 bytes.

data _ w _ _ 8 7 _ _ 2 0 _ 1 5
bytes   2 : 4 : 4 : 3
stand for type x y color

The first two bytes _w for dot; then next 4 bytes for x; 4 bytes for y; and 3 bytes for color.

3.2 The Line, Box, & Painted Box Format 'l'

The 'l' format is defined for line, box, & painted box. This data line consists of 24 bytes.

data _ 1 _ _ 8 7 _ _ 1 2 0 _ _ 1 8 7 _ _ 1 7 0 _ _ 1 3 _ _ b
bytes   2 : 4 : 4 : 4 : 4 : 3 : 3
stand for type xl y1 x2 y2 color add
The 'l' format needs 2 points \((x_1,y_1)\) & \((x_2,y_2)\) for the line, box, or painted box drawing. The additional code at the last 3 bytes are used to specify the type for line, box, or painted box. The addition code here stands for:

- _ _ : for line.
- _ _ b : for box.
- _ b f : for painted box.

3.3 The Circle and Arc Format 'c'

The C format is used to draw circles and circular arcs. This data line consists of 33 bytes.

Data format and description:

```
  _ c _ 1 3 7 _ 3 1 0 _ 5 0 _ 1 5 _ _ _ _ _ _ 4 5 0 _ 1 3 5 0 _ 2 / 3
```

where:

- \(x, y\) : This is the center of the circle.
- \(x, y\) : The radius of the circle & arc.
- \(x, y\) : The aspect is a ratio between \(x/y\). The two digit numbers are divide by a slash sign. The default for \(x/y\) is 1/1.
3.4 The 16x16 Text Format 'e'

The 'e' format is a line that contains both English and Chinese 16x16 text. For 16x16 text, it is possible to select text color and background color. This data line consists of 53 bytes.

Data format and description:

_ e _ 2 8 _ _ 2 _ 1 5 _ _ 6 _ _ ....... _ C A I
2 : 4 : 4 : 3 : 3 : 40 :
type: x : y :color:Bcolr: text string :

As was mentioned before, each Chinese character takes two bytes for its interchange code. On the screen, each Chinese word also takes two columns for display.

3.5 The 24x24 Text Format 't'

Usually the 't' format is used to print Chinese words. The format of 't' is more complicated than the other formats. The ET Chinese system takes the 24x24 text as a graphic shape, thus making it easier to enlarge or to print the word font in different logic types. The logic types include AND, OR, XOR, reverse, or even stereo type words.

Data format and description:

_ t _ 3 8 7 _ 2 7 0 _ 1 5 _ _ 2 _ 2 / 2 _ * _ _ ....... C H I N E S E
2 : 4 : 4 : 3 : 2 : 4 : 2 : 40 :
type: x : y :color:gap:XYratio:log: text string :
where

x,y : The coordinate location of the Up-Left corner of the text string.
color : Range from 0 to 15.
gap : A number for the gap between two words.
XYratio: A ratio of x per y. Two digits a divided by a slash sign.
logic : Some passable letters here are -
's' this is a default with bright text color and black background color.
'r' reverse the background and text color.
'x' each bit inside the 24x24 are displayed by an XOR operation.
'a' each bit inside the 24x24 are displayed by an AND operation.
'o' each bit inside the 24x24 are displayed by an OR operation.
'*' display the Chinese text to a stereo shape.
4. BASIC OPERATION

The CGE.DAT data files are created at the time the user enters the CGGE. Graphic data are generated at the time it is recorded on the screen. The other file with PASCAL instructions are created soon after the S key is pressed for saving the file. One thing to remember is that the user must know how to input those data mentioned in the last few sections. For better understanding, some key points of data input will be summarized in the next few paragraphs.

4.1 Dots Drawing

. Move the cursor to the place where one wants a dot to appear.
. Select function 1.
. Input a color number from 0 to 15.

Because the dot is hidden behind the cursor, to see the dot it is necessary to move the cursor away, and then press R to redraw the object on the screen.

4.2 Lines Drawing

. Move the cursor to the location where one wants the line to appear.
. Select function 2.
. Move the cursor to the other end of the line, then press <RETURN> to specify the second point.
. Input a color number from 0 to 15.

4.3 Boxes Drawing

. Move the cursor to the place where one of the box angles is located.
. Select function 3.
. Move the cursor to the opposite corner of the box, then press the <RETURN> key.
. Input a color number from 0 to 15.
4.4 Painted Boxes (Box Filled)

. Move the cursor to the location where one corner of the filled box is located.
. Move the cursor to the opposite corner of the box, then press the <RETURN> key.
. Give it a number from 0 to 15.

4.5 Circles

. Move the cursor to the point where the center of the circle is located.
. Select function 4.
. Move the cursor to a point where one wants a circle to pass through.
. Input the X/Y ratio. The format for the ratio is '___/___'. The underline stands for a digit, the former for the X scale, and the latter for the Y scale—for example: 2/2 or 2/3.
. Input the color number for the circle.

4.6 Circular Arcs

. Move the cursor to the center of the arc.
. Select function 5.
. Move the cursor to the location which the circle corresponding to the arc is to go through, then press the <RETURN> key to specify the radius.
. Input a start angle between 0 to 360. A minus number will connect a line from the start angle to the arc center.
. Input an end angle between 0 to 360. A minus number will connect a line from the end angle to the arc center.
. Input the X/Y ratio.
. Input a color number from 0 to 15.

4.7 Print 16x16 Texts

. Select 7 for the text function.
. Select 1 for 16x16 text.
. Move the cursor to the place where the text will locate.
. Key in the text. The text can be either English or Chinese.
. Input two color numbers, and put a space between these two numbers. The first number will be for the text color and the next for the background color. The default value for the text background is zero, which is the number for black.
4.8 Print 24x24 Texts

- Move the cursor to a point which is the location of the upper-left of the texts.
- Select function 7.
- Select 2 for 24x24 text printing.
- Key in the text, mostly Chinese texts.
- Input the text color. After the text color, input the other arguments for special printing function, such as 15 2 2/2 *. See the detailed description below:

Data input format and description:

Input format : 15_2_2/2_ *
Space : 1 or 2 : 4 : 2 :
Function : color:gap:XYratio:log:

where

- color : Range from 0 to 15.
- gap : A number for the gap between two words.
- XYratio : A ratio of x per y. Two digits divided by a slash sign.
- logic : 's' A default with a bright text color and black background color.
  'r' reverses the background and text color.
  'x' each bit inside the 24x24 is displayed by an XOR operation.
  'a' each bit inside the 24x24 is displayed by an AND operation.
  'o' each bit inside the 24x24 is displayed by an OR operation.
  '*' displays the Chinese text in a stereo shape.
Appendix C.

Voice Editor BASIC Program List (I)

10 CLS
20 LOCATE 8,30:PRINT"voice menu:"
30 LOCATE 10,30:PRINT"1. record."
40 LOCATE 11,30:PRINT"2. play."
50 LOCATE 13,30:INPUT ">";S
60 IF S=1 THEN 100
70 IF S=2 THEN 200
80 GOTO 10
100 LOCATE 15,24:PRINT "press 'S' for save 1-8 voice."
102 LOCATE 16,30:PRINT "'M' to voice menu."
103 LOCATE 18,51:PRINT ""
104 LOCATE 18,30:INPUT "record voice ( 1-8,S,M ) = ",VS
105 IF VS="s" OR VS="S" THEN GOSUB 2000 : GOTO 10
106 IF VS="m" OR VS="M" THEN 10
110 LOCATE 20,30:PRINT "press any key when ready..":A$=INPUT$(1)
115 LOCATE 20,60 :PRINT ""
120 FOR I=1 TO 100 : NEXT I : OUT &H3E6,&HC0
125 FOR I=1 TO 9000 : NEXT I : COLOR 27,0:LOCATE 20,60:PRINT:"**":COLOR 7,0
190 GOSUB 1000
195 BEEP :LOCATE 20,30 :PRINT STRINGS$(50," "):GOTO 100
200 '
205 LOCATE 15,24:PRINT "press 'L' for load 1-8 voice."
208 LOCATE 16,30:PRINT "'M' to voice menu."
210 LOCATE 18,51:PRINT ""
215 LOCATE 18,30:INPUT "play voice ( 1-8,L,M ) = ",VS
218 IF VS="l" OR VS="L" THEN GOSUB 3000 : GOTO 10
220 IF VS="m" OR VS="M" THEN 10
225 LOCATE 20,30:PRINT "press any key when ready..":A$=INPUT$(1)
228 COLOR 27,0:LOCATE 20,60:PRINT:"***":COLOR 7,0
230 GOSUB 1100
240 OUT &H3F7,&H90:OUT &H3E6,&H80:FOR I=1 TO 100 : NEXT I : OUT &H3E6,&HCO
290 BEEP :LOCATE 20,30 :PRINT STRINGS$(50," "):GOTO 200
999 STOP
1000 :
1005 DEF SEG=&H8000:ADDR=VAL(VS)*8192
1010 OUT &H3E7,&H90:LOCATE 1,1:PRINT ADDR
1020 FOR I=0 TO 31:AI=I*256
1030 OUT &H3E6,I+32
1040 FOR J=0 TO 255
1050 OUT &H3E5,J
1060 POKE ADDR+AI+J,INP(&H3E4)
1070 NEXT J
1080 NEXT I
1090 OUT &H3F6,&HFO
1095 RETURN

1100 :
1105 DEF SEG=&H8000: ADDR=VAL(V$)*8192
1110 OUT &H3E7,&H80: LOCATE 1,70: PRINT ADDR
1120 FOR I=0 TO 31: AI=I*256
1130 OUT &H3E6,I+96
1140 FOR J=0 TO 255
1150 OUT &H3E5,J
1160 OUT &H3E4, PEEK(ADDR+AI+J)
1170 OUT &H3E6,I+64
1180 OUT &H3E6,I+96
1190 NEXT J
1200 NEXT I
1210 OUT &H3E7,&H90
1220 OUT &H3F6,&HFO
1230 RETURN

2000 :
2010 DEF SEG=&H8000
2020 BSAVE "b:voice.dat", &H2000, &HE000
2030 DEF SEG
2040 RETURN

3000 :
3010 DEF SEG=&H8000
3020 BLOAD "b:voice.dat"
3030 DEF SEG
3040 RETURN
Appendix D.

Voice Editor BASIC Program List (II)  
(with assembly routines)

10 REM Program: 'VOICETST' by Jia-Rong Jerome Wen  December 21-1988
20 REM This program offers the functions for Voice test of CAI system
30 OUT &H3E7,&H90
40CLS
50 OUT &H3E6,&HFO
60 COLOR 14,6
70 LOCATE 2,17
   : PRINT"Computer Assisted Video Chinese Instructional System"
80 COLOR 12,0
90 LOCATE 3,25: PRINT"*** (CAVCIS) : VOICE EDITOR ***"
100 LOCATE 4,25: PRINT"BASIC version, by Jerome 11/19-88!"
110 COLOR 10,0
120 LOCATE 6,30: PRINT "1 record"
130 LOCATE 8,30: PRINT "2 play"
140 LOCATE 10,30: PRINT "3 voice RAM --> main RAM"
150 LOCATE 12,30: PRINT "4 main RAM --> voice RAM"
160 LOCATE 14,30: PRINT "5 save 1-7 voice to disk"
170 LOCATE 16,30: PRINT "6 load 1-7 voice from disk"
180 LOCATE 18,30: PRINT "7 Quit"
190 LOCATE 20,30: INPUT"select=", A$: IF VAL(A$)=7 THEN COLOR 7,0:END
200 ON VAL(A$) GOSUB 290,320,230,230,900,950
210 PRINT:PRINT
220 GOTO 30
230 LOCATE 22,30: INPUT"segment (1--7)"; V$: V=VAL(V$)
240 IF V<1 OR V>7 THEN 230
250 IF A$="3" THEN GOSUB 380
260 IF A$="4" THEN GOSUB 600
270 GOTO 30
280 REM ***********************
290 OUT &H3E6,&H80
300 FOR I=1 TO 200: NEXT I
310 RETURN
320 OUT &H3E6,&H90
330 FOR I=1 TO 200: NEXT I
340 OUT &H3E6,&HCO
350 RETURN
360'
370'
380 REM voice RAM --> main RAM
390:
400 OUT &H3E7,&H90: LOCATE 4,1: PRINT "8000:"HEXS(V*2)"000"
410 DEF SEG=&H8000:DEF USR=0
420 RESTORE 470
430 FOR I=0 TO 59
440 READ DA
450 POKE I,DA
460 NEXT I
470 DATA &h50,&h53,&h51,&h1E,&h56,&hB8,&h00,&h80,&h8E,&hD8
480 DATA &hA0,&h3c,&h00,&h90,&hB1,&h0D,&hE0,&h89,&hC6
490 DATA &hB9,&h00,&h20,&hBB,&h00,&hBA,&hE5,&h03,&h88
500 DATA &hD8,&hee,&hBA,&hE6,&h03,&h88,&hF8,&hOC,&h20,&hee
510 DATA &h90,&h90,&h90,&hBA,&hE4,&h03,&hEC,&h88,&h00
520 DATA &h43,&h49,&h75,&hE4,&h5E,&h1F,&h59,&h5B,&h58,&hCB
530 V=VAL(V$)
540 IF V<1 OR V>7 THEN RETURN
550 POKE &h3C,V ' parameter transfer
560 Y=USR(0)
570 DEF SEG
580 BEEP
590 RETURN
600 REM
610 REM main RAM ---> voice RAM
620 REM
630 DEF SEG=&H8000:DEF USR=0
640 LOCATE 4,1:PRINT "8000:"HEX$(V*2)"000"
650 OUT &H3E7,&H80:LOCATE 4,70
660 RESTORE 720
670 FOR I=0 TO 95
680 READ DA
690 POKE I,DA
700 NEXT I
710 V=VAL(V$)
720 DATA &h50,&h53,&h51,&h1E,&h56,&hBA,&he7,&h03,&hb0,&h80
730 DATA &hee,&h88,&h00,&h80,&h8E,&hD8
740 DATA &hAO,&hFF,&h00,&h90,&hBL,&hOD,&hE0,&h89,&hC6
750 DATA &hB9,&h00,&h20,&hBB,&h00,&hBA,&hE5,&h03,&h88
760 DATA &hD8,&hee,&hBA,&hE6,&h03,&h88,&hF8,&hOC,&h60,&hee
770 DATA &h90,&h90,&h90,&hBA,&hE4,&h03,&h8A,&h00,&hee
780 DATA &h90,&hBA,&hE6,&h03,&h88,&hF8,&hOC,&h40,&hee,&h90,&h90
790 DATA &h90,&h90,&h90,&hBA,&hE6,&h3,&h88,&hF8,&hOC,&hee
800 DATA &h43,&h49,&h75,&hCD,&h5E,&h1F,&h59
810 DATA &h5B,&h58,&hCB,&h90,&h90,&h90,&h90,&h90,&h90
820 OUT &H3E7,&h90
830 OUT &H3F6,&hFO
840 IF V<1 OR V>7 THEN RETURN
850 POKE &hFF,V
860 Y=USR(1)
870 DEF SEG
880 BEEP
890 RETURN
900 REM *** SAVE
910 DEF SEG=&H8000
920 BSAVE"VOICE.DAT",&H2000,&H8000
930 DEF SEG
940 RETURN
950 REM *** LOAD
960 DEF SEG=&H8000
970 BLOAD"VOICE.DAT"
980 DEF SEG
990 RETURN
Appendix E.

Voice Process Assembly Routines

E.1: Assembly routine for moving voice from RAM to main memory

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Instruction</th>
<th>Register/Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000:0000</td>
<td>PUSH AX</td>
<td></td>
</tr>
<tr>
<td>8000:0001</td>
<td>PUSH BX</td>
<td></td>
</tr>
<tr>
<td>8000:0002</td>
<td>PUSH CX</td>
<td></td>
</tr>
<tr>
<td>8000:0003</td>
<td>PUSH DS</td>
<td></td>
</tr>
<tr>
<td>8000:0004</td>
<td>PUSH 55</td>
<td></td>
</tr>
<tr>
<td>8000:0005</td>
<td>NOV AX, 8000</td>
<td></td>
</tr>
<tr>
<td>8000:0008</td>
<td>NOV DS, AX</td>
<td></td>
</tr>
<tr>
<td>8000:000A</td>
<td>NOV AL,[003C]</td>
<td></td>
</tr>
<tr>
<td>8000:000D</td>
<td>HOP</td>
<td></td>
</tr>
<tr>
<td>8000:000E</td>
<td>NOV CL, 0D</td>
<td></td>
</tr>
<tr>
<td>8000:0010</td>
<td>SHL AX, CL</td>
<td></td>
</tr>
<tr>
<td>8000:0012</td>
<td>NOV SI, AX</td>
<td></td>
</tr>
<tr>
<td>8000:0014</td>
<td>NOV CX, 2000</td>
<td></td>
</tr>
<tr>
<td>8000:0017</td>
<td>NOV BX, 0000</td>
<td></td>
</tr>
<tr>
<td>8000:001A</td>
<td>NOV DX, 03E5</td>
<td></td>
</tr>
<tr>
<td>8000:001D</td>
<td>NOV AL, BL</td>
<td></td>
</tr>
<tr>
<td>8000:001F</td>
<td>OUT DX, AL</td>
<td></td>
</tr>
<tr>
<td>8000:0020</td>
<td>NOV DX, 03E6</td>
<td></td>
</tr>
<tr>
<td>8000:0023</td>
<td>NOV AL, BH</td>
<td></td>
</tr>
<tr>
<td>8000:0025</td>
<td>CR AL, 20</td>
<td></td>
</tr>
<tr>
<td>8000:0027</td>
<td>OUT DX, AL</td>
<td></td>
</tr>
<tr>
<td>8000:0028</td>
<td>HOP</td>
<td></td>
</tr>
<tr>
<td>8000:0029</td>
<td>HOP</td>
<td></td>
</tr>
<tr>
<td>8000:002A</td>
<td>HOP</td>
<td></td>
</tr>
<tr>
<td>8000:002B</td>
<td>HOP</td>
<td></td>
</tr>
<tr>
<td>8000:002C</td>
<td>NOV DX, 03E4</td>
<td></td>
</tr>
<tr>
<td>8000:002F</td>
<td>IN AL, DX</td>
<td></td>
</tr>
<tr>
<td>8000:0030</td>
<td>NOV [BX+SI], AL</td>
<td></td>
</tr>
<tr>
<td>8000:0032</td>
<td>INC BX</td>
<td></td>
</tr>
<tr>
<td>8000:0033</td>
<td>DEC CX</td>
<td></td>
</tr>
<tr>
<td>8000:0034</td>
<td>JNZ 001A</td>
<td></td>
</tr>
<tr>
<td>8000:0036</td>
<td>POP 51</td>
<td></td>
</tr>
<tr>
<td>8000:0037</td>
<td>POP DS</td>
<td></td>
</tr>
<tr>
<td>8000:0038</td>
<td>POP CX</td>
<td></td>
</tr>
<tr>
<td>8000:0039</td>
<td>POP BX</td>
<td></td>
</tr>
<tr>
<td>8000:003A</td>
<td>POP AX</td>
<td></td>
</tr>
<tr>
<td>8000:003B</td>
<td>RETF</td>
<td></td>
</tr>
</tbody>
</table>
E.2: Assembly routine for moving voice from main memory to RAM

<table>
<thead>
<tr>
<th>Offset</th>
<th>Assembly Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000:0000</td>
<td>50</td>
<td>PUSH AX</td>
</tr>
<tr>
<td>8000:0001</td>
<td>53</td>
<td>PUSH BX</td>
</tr>
<tr>
<td>8000:0002</td>
<td>51</td>
<td>PUSH CX</td>
</tr>
<tr>
<td>8000:0003</td>
<td>1E</td>
<td>PUSH DS</td>
</tr>
<tr>
<td>8000:0004</td>
<td>56</td>
<td>PUSH SI</td>
</tr>
<tr>
<td>8000:0005</td>
<td>BAE703</td>
<td>NOV DX,03E7</td>
</tr>
<tr>
<td>8000:0008</td>
<td>B080</td>
<td>NOV AL, 80</td>
</tr>
<tr>
<td>8000:000A</td>
<td>EE</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>8000:000B</td>
<td>B8080</td>
<td>NOV AX, 8000</td>
</tr>
<tr>
<td>8000:000E</td>
<td>8ED8</td>
<td>NOV DS, AX</td>
</tr>
<tr>
<td>8000:0010</td>
<td>AOFF00</td>
<td>NOV AL, [00FF]</td>
</tr>
<tr>
<td>8000:0013</td>
<td>90</td>
<td>HOP</td>
</tr>
<tr>
<td>8000:0014</td>
<td>B10b</td>
<td>NOV CL, OD</td>
</tr>
<tr>
<td>8000:0016</td>
<td>D3E0</td>
<td>SHL AX, CL</td>
</tr>
<tr>
<td>8000:0018</td>
<td>89C6</td>
<td>NOV SI, AX</td>
</tr>
<tr>
<td>8000:001A</td>
<td>B90020</td>
<td>MOV CX, 2000</td>
</tr>
<tr>
<td>8000:001D</td>
<td>B60000</td>
<td>MOV BX, 0000</td>
</tr>
<tr>
<td>8000:0020</td>
<td>BAE503</td>
<td>MOV DX, 03E5</td>
</tr>
<tr>
<td>8000:0023</td>
<td>88D8</td>
<td>NOV AL, BL</td>
</tr>
<tr>
<td>8000:0025</td>
<td>EE</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>8000:0026</td>
<td>BAE603</td>
<td>NOV DX, 03E6</td>
</tr>
<tr>
<td>8000:0029</td>
<td>88F8</td>
<td>NOV AL, BH</td>
</tr>
<tr>
<td>8000:002B</td>
<td>OC60</td>
<td>OR AL, 60</td>
</tr>
<tr>
<td>8000:002D</td>
<td>EE</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>8000:002E</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:002F</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0030</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0031</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0032</td>
<td>BAE403</td>
<td>NOV DX, 03E4</td>
</tr>
<tr>
<td>8000:0035</td>
<td>8A00</td>
<td>NOV AL, [BX+SI]</td>
</tr>
<tr>
<td>8000:0037</td>
<td>EE</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>8000:0038</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0039</td>
<td>BAE603</td>
<td>NOV DX, 03E6</td>
</tr>
<tr>
<td>8000:003C</td>
<td>88F8</td>
<td>NOV AL, BH</td>
</tr>
<tr>
<td>8000:003E</td>
<td>OC40</td>
<td>OR AL, 40</td>
</tr>
<tr>
<td>8000:0040</td>
<td>EE</td>
<td>OUT DX, AL</td>
</tr>
<tr>
<td>8000:0041</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0042</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0043</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0044</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0045</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0046</td>
<td>90</td>
<td>MOP</td>
</tr>
<tr>
<td>8000:0047</td>
<td>BAE603</td>
<td>NOV DX, 03E6</td>
</tr>
<tr>
<td>Address</td>
<td>Opcode</td>
<td>Instruction</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>8000:004A</td>
<td>88F8</td>
<td>NOV</td>
</tr>
<tr>
<td>8000:004C</td>
<td>0C60</td>
<td>OR</td>
</tr>
<tr>
<td>8000:004E</td>
<td>EE</td>
<td>OUT</td>
</tr>
<tr>
<td>8000:004F</td>
<td>43</td>
<td>IMC</td>
</tr>
<tr>
<td>8000:0050</td>
<td>49</td>
<td>DEC</td>
</tr>
<tr>
<td>8000:0051</td>
<td>75CD</td>
<td>JMZ</td>
</tr>
<tr>
<td>8000:0053</td>
<td>SE</td>
<td>POP</td>
</tr>
<tr>
<td>8000:0054</td>
<td>IF</td>
<td>POP</td>
</tr>
<tr>
<td>8000:0055</td>
<td>59</td>
<td>POP</td>
</tr>
<tr>
<td>8000:0056</td>
<td>SB</td>
<td>POP</td>
</tr>
<tr>
<td>8000:0057</td>
<td>58</td>
<td>POP</td>
</tr>
<tr>
<td>8000:0058</td>
<td>CB</td>
<td>RETF</td>
</tr>
</tbody>
</table>
Appendix F.

VCR & Slide Project Test Program
(with assembly routines)

Filename : VCRTEST.BAS

SANYO VCR CONTROL PROGRAM

01-STOP, 02-PLAY, 03-FF, 04-FR, 05-PAUSE, 06-RESET, 07-CLEAR,
08-COUNT CLEAR, 09-AUDIO, 10-VIDEO, 11-COUNT ENABLE,
12-SLIDE LAMP, 13-SLIDE REVERSE, 14-SLIDE FORWARD,
15-TP(0)- VTR play 0:off 1:on
16-TP(1)- audio 0:off 1:on
17-TP(2)- video 0:computer 1:VTR
18-TP(3)- slide lamp 0:off 1:on
190 CLS:KEY OFF
210 OUT &H3E3,&H8B
220 OUT &H3E0,&H6:FOR I=1 TO 100:NEXT I
230 OUT &H3E0,&H44 : FOR I=0 TO 1000 : NEXT I
240 X=INP (&H3E1)*256+INP(&H3E2):IF X=RX THEN 170
250 RX=X:FOR I=1 TO 200 : NEXT I
260 GOTO 140
270 OUT &H3E0,&H7:FOR I=1 TO 100:NEXT I
280 OUT &H3E0,&H8:FOR I=1 TO 100:NEXT I
290 CLS
300 LOCATE 5,20 :PRINT"VTR FUNCTIONTS:
310 LOCATE 7,20 :PRINT"1. Play"
320 LOCATE 8,20 :PRINT"2. Fast Forward"
330 LOCATE 9,20 :PRINT"3. Fast Backward"
340 LOCATE 10,20:PRINT"4. -Pause (on/off)"
350 LOCATE 11,20:PRINT"5. -Stop"
360 LOCATE 12,20:PRINT"6. -Video (COMP/VTR)"
370 LOCATE 13,20:PRINT"7. -Audio"
380 LOCATE 14,20:PRINT"8. Slide (on/off)"
390 LOCATE 15,20:PRINT"9. Slide Forward"
400 LOCATE 16,20:PRINT"10. Slide Backward"
410 LOCATE 17,20:PRINT"11. Exit to BASIC"
420 LOCATE 19,20:INPUT"Function select (1-11)
430 IF N$="1" THEN LOCATE 7,40 : INPUT "From Address : ",S : S=S*3
440 IF N$="2" OR N$="3" THEN LOCATE VAL(N$)+6,40
450 IF N$="9" OR N$="1" THEN 320
460 ON VAL(N$) GOSUB 380,670,760,,850,890,920,1040
470 CLS : GOTO 200
480:
490 X=INP (&H3E1)*256+INP(&H3E2)
500 IF X+3<S THEN E=S: GOSUB 670 : GOTO 420
410 IF X-3>S THEN E=S: GOSUB 760
420 CLS: LOCATE 9,20 : PRINT" VTR is playing now "
430 LOCATE 11,20 : PRINT" P for VTR Pause."
440 LOCATE 12,20 : PRINT" C for Video from Computer."
450 LOCATE 13,20 : PRINT" V for Video from VTR."
460 LOCATE 14,20 : PRINT" A for Audio (on/off)."
470 LOCATE 15,20 : PRINT" S for Slide (on/off)."
480 LOCATE 16,20 : PRINT" F for Slide Forward."
490 LOCATE 17,20 : PRINT" B for Slide Backward."
500 LOCATE 18,20 : PRINT" ESC for VTR Stop."
510 OUT &H3E0,&H2+TP(3)*&H80:FOR 1=1 TO 200:NEXT I
520 TP(0)=1:TP(1)=1:TP(2)=0:OUT &H3E0,&H42+TP(3)*&H80
530 X=INP (&H3E1)*256+INP(&H3E2) : A$=INKEY$
540 LOCATE 1,60 : PRINT"ct=";INT(X/3)
550 IF X=>55 OR A$=CHR$(27) THEN 640
560 IF A$="p" OR A$="P" THEN GOSUB 950
570 IF A$="v" OR A$="v" THEN TP(2)=1
580 IF A$="c" OR A$="c" THEN TP(2)=0
590 IF A$="a" OR A$="A" THEN TP(1)=ABS(TP(1)-1)
600 IF A$="s" OR A$="S" THEN GOSUB 850
610 IF A$="f" OR A$="F" THEN GOSUB 890
620 IF A$="b" OR A$="B" THEN GOSUB 920
630 GOTO 530
640 TP(0)=0:TP(1)=0:TP(2)=0
650 OUT &H3E0,&H10+TP(3)*&H80 : FOR I=1 TO 200 : NEXT I
660 CLS : RETURN
670 :
680 CLS : LOCATE 12,30 : PRINT"VTR is now Fast forward....."
690 OUT &H3E0,&H3+TP(3)*&H80:FOR I=1 TO 200:NEXT I
700 X=INP (&H3E1)*256+INP(&H3E2)
710 IF X>=E-1 THEN 740
720 A$=INKEY$:IF A$="" GOTO 740
730 GOTO 700
740 OUT &H3E0,&H1+TP(3)*&H80:FOR I=1 TO 200 : NEXT I
750 CLS : RETURN
760 :
770 CLS : LOCATE 12,30 : PRINT"VTR is now Fast Backward....."
780 OUT &H3E0,&H4+TP(3)*&H80:FOR I=1 TO 200:NEXT I
790 X=INP (&H3E1)*256+INP(&H3E2)
800 IF X<=E+1 THEN 830
810 A$=INKEY$:IF A$="" GOTO 830
820 GOTO 790
830 OUT &H3E0,&H1+TP(3)*&H80:FOR I=1 TO 200 : NEXT I
840 CLS: RETURN
850:
860 IF TP(3)=0 THEN TP(3)=1 ELSE TP(3)=0
870 OUT &H3E0,TP(0)*&H42+TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
880 RETURN
890:
900 OUT &H3E0,TP(0)*&H40+TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
910 RETURN
920:
930 OUT &H3E0,TP(0)*&H40+TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
940 RETURN
950:
960 OUT &H3E0,&H5+TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
970 FOR I=1 TO 1500: NEXT I
980 OUT &H3E0,TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
990 A$=INPUT$(1):A$=""
1000 OUT &H3E0,&H5+TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
1010 FOR I=1 TO 1000: NEXT I
1020 OUT &H3E0,&H42+TP(1)*&H10+TP(2)*&H20+TP(3)*&H80
1030 RETURN
1040:
1050 OUT &H3E0,&H0
1060 END
Appendix G.

Program List Of
The Chinese Courseware Graphics Editor (CCGE Program)

Program CoursewareGraphicEditor;

{By: Jia-Rong Jerome Wen, 7E9E:a}
{Date: September 5-1988 }
{At: Iowa State University }
{For : My mamma and my family. }

type
detail=record
  command:char;
  xl,y1,x2,y2:integer;
  radius:integer;
  color,bcolor:integer;
  box:string[3];
  gap,logic:string[2];
  rxry,aspect:string[4];
  arcstart,arcend:integer;
  keyintext:string[80];
end;
modech=char;
regpack = record
  ax,bx,cx,dx,bprsi,di,ds,es,flags: integer;
end;

var
topLine, bottomLine : byte;
xf,yf,xs,ys,snx2,sny24,sny25,
x,y,ic,ix,ly,pic :integer;
coml :detail;
outf,inf,renf :text;
stx,sty,tic,rslt :integer;
t24 :boolean;
deline :string[100];
mode :modech;

procedure Cursor_Control(topLine, bottomLine : byte);
var
  registers : regpack;
begin
with Registers do
begin
  ax := 1 shl 8;
  cx := topLine shl 8 + bottomLine;
  INTR($10, registers);
end;
end;

procedure gotoclr(var scnx,scny:integer);
var fn:integer;
  sp:string[80];
begin
  sp:="";
  for fn:=1 to 70 do insert(' ',sp,1);
  gotoxy(scnx,scny);
  write(sp);
  gotoxy(scnx,scny);
end;

procedure loadfmdat(mode:modech);
var
  crefile,datfile,infile :text;
  f :string[14];
  commandline :string[132];
  n :integer;
begin
  gotoclr(snx2,sny24);
  write('=?i$J@I&W');
  topline:=6; bottomline:=7;
  Cursor_Control(topLine,bottomLine);
  readln(f);
  topline:=48; bottomline:=0;
  Cursor_Control(topLine,bottomLine);
  n:=pos('.',f); if n<>0 then f:=copy(f,1,n-1);
  f:=f+'.dat';
  assign(infile,f);
  {si-}
  reset(infile);
  {si+}
  rslt:=ioresult;
  if rslt>0 then
    begin
      gotoxy(20,24);
      textcolor(14);
      write('d$#(1@I.W!! ! •);
textcolor(15);
repeat until keypressed;
writeln
end
else
begin
    assign(datfile,'cge.dat');
    if mode='l' then rewrite(datfile);
    if mode='i' then append(datfile);
    while not eof(infile) do
    begin
        readln(infile,coiranandline);
        writeln(datfile,coinmandline);
    end;
    close(infile);
    close(datfile);
end;
end;

procedure savetopas;
var
    crefile,datfile,infile:text;
    f:string[14];
    commandline:string[132];
    x,y,n:integer;
    txl,tyl,tx2,ty2,tradius,tcolor,tbcolor,
    tarcstart,tarcend:string[10];
    coral:detail;
begin
    gotoclr(snx2,sny24);
    write('=P?i$J@I&W:');
    topline:=6; bottomline:=7;
    Cursor_Control(topLine,bottomLine);
    readln(f);
    topline:=48; bottomline:=0;
    Cursor_Control(topLine,bottomLine);
    n=pos('.',f); if n<>0 then f:=copy(f,l,n-l);
    f:=f+'.pas';
    assign(crefile,f);
    rewrite(crefile);
    writeln(crefile,'begin clrscr;');
    assign(infile,'cge.dat');
    reset(infile);
    while not eof(infile) do
    begin
        with coral do
        begin
            read(infile,command);
            case command of
            end;
            case command of
'p', 'w': begin
  readln(infile, xl, yl, color);
  str(xl, txl); str(yl, tyl); str(color, tcolor);
  commandline:='write(char(27),'+char(39)+command+txl
  +', '+tyl+', '+tcolor+', '+char(39)+');';
  writeln(crefile, commandline);
end;
'c': begin
  readln(infile, xl, yl, radius, color, arcstart, arcend
  , aspect);
  str(xl, txl); str(yl, tyl);
  str(radius, tradius); str(color, tcolor);
  if arcstart=0 then tarcstart=''
  else str(arcstart, tarcstart);
  if arcend=0 then tarcend=''
  else str(arcend, tarcend);
  commandline:='write(char(27),'+char(39)+'c'+txl+',
  '+tyl+', '+tradius+', '+tcolor+', '+tarcstart
  +', '+tarcend+', '+aspect+', '+char(39)+');';
  writeln(crefile, commandline);
end;
'l': begin
  readln(infile, xl, yl, x2, y2, color, box);
  str(xl, txl); str(yl, tyl);
  str(x2, tx2); str(y2, ty2);
  str(color, tcolor);
  commandline:='write(char(27),'+char(39)+'1'+txl,
  '+', '+tyl+', '+tx2+', '+ty2+', '+tcolor+', '+box+', '+
  char(39)+');';
  writeln(crefile, commandline);
end;
't': begin
  readln(infile, xl, yl, color, gap, rxry, logic, keyintext);
  n:=1;
  while copy(keyintext, n, 1)=' ' do n:=n+1;
  keyintext:=copy(keyintext, n, length(keyintext));
  str(xl, txl); str(yl, tyl); str(color, tcolor);
  if logic=' *' then
    begin
      commandline:='write(char(27),'+char(39)+'t24,'+txl,
      +', '+tyl+', '+gap+', '+rxry+', '+char(39)+'));';
      writeln(crefile, commandline);
      commandline:='write(char(27),'+char(39)+'sd'+tcolor,
      '+keyintext+char(39)+');';
      writeln(crefile, commandline);
      str(xl+l, txl); str(yl+l, tyl);
      commandline:='write(char(27),'+char(39)+'t24,'+txl,
      +', '+tyl+', '+gap+', '+rxry+', x', '+char(39)+'));';
      writeln(crefile, commandline);
commandline:='write(char(27),'+char(39)+'sd'+tcolor +';'+keyintext+char(39)+');'; writeln(crefile,commandline); end
else
begin
     commandline:='write(char(27),'+char(39)+'t24,'+txl +','+tyl+','+gap+','+rxry+','+logic+';' +char(39)+');'; writeln(crefile,commandline);
     commandline:='write(char(27),'+char(39)+'sd'+tcolor +';'+keyintext+char(39)+');'; writeln(crefile,commandline);
     end;
end;
commandline:='write(char(27),'+char(39)+'tl6;' +char(39)+');'; writeln(crefile,commandline);end;
'end:
begin
readln(infile,xl,yl,color,bcolor,keyintext);
n:=l;
while copy(keyintext,n,1)=' ' do n:=n+l;
keyintext:=copy(keyintext,n,length(keyintext));
str(xl,txl); str(yl,tyl);
str(color,tcolor); str(bcolor,tbcolor);
commandline:='gotoxy('+txl+','+tyl+');'
write(crefile,commandline);
commandline:='textbackground('+tbcolor+');'
write(crefile,commandline);
commandline:='textcolor('+tcolor+');'
write(crefile,commandline);
commandline:='write('+char(39)+keyintext+char(39)+');'
write(crefile,commandline);
writeln(crefile,'textcolor(15);');
writeln(crefile,'textbackground(O);');
end;
end;
end;
writeln(crefile,'end.');</refile>
close (crefile);
n:=pos('.',f); if n<>0 then f:=copy(f,1,n-1);
f:=f+'.dat';
assign(datfile,f);
rewrite(datfile);
reset(infile);
while not eof(infile) do begin
     readln(infile,commandline);
writeln(datafile,commandline);
end;
close(datafile);
close(infile);
end;

procedure gettextxy;
var
specials,doneS:boolean;
ch :char;
begin
topline:=6; bottoraline:=7;
Cursor_Control(topline,bottomline);
textcolor(11);
gotoxy(76,24); write(stx:4);
gotoxy(76,25); write(sty:4);
gotoxy(66,25);
textcolor(7);
write(tic:3);
textcolor(15);
done3:=false;
repeat begin
  gotoxy(stx,sty);
  specials := false;
  read(kbd,ch);
  if (ch=#27) then
    if keypressed then
      begin
        specials :=true;
        read(kbd,ch);
      end
    else
      begin done3:=true; gotoxy(1,24); end;
  end
  if not doneS then
    if specials then
      begin
        case ord(ch) of
          75: stx:=stx-tic;
          77: stx:=stx+tic;
          72: sty:=sty-tic;
          80: sty:=sty+tic;
          71: begin stx:=stx-tic; sty:=sty-tic; end;
          81: begin stx:=stx+tic; sty:=sty+tic; end;
          73: begin stx:=stx+tic; sty:=sty-tic; end;
          79: begin stx:=stx-tic; sty:=sty+tic; end;
          82: begin
            if tic>10 then tic:=10 else tic:=11-tic;
gotoxy(66,25);
textcolor(7); write(tic:3); textcolor(15);
if (stx<l) or (stx>72) then stx:=abs(72-abs(stx));
if (sty<l) or (sty>23) then sty:=abs(23-abs(sty));
textcolor(ll);
gotoxy(76,24); write(stx:4);
gotoxy(76,25); write(sty:4);
textcolor(15);
end
else
case ord(ch) of
  l3: done3:=true;
end
end;
until done3;
topline:=48; bottomline:=0;
Cursor_Control(topLine,bottomLine);
end;

procedure redraw;
var
  n :integer;
  coml:detail;
begin
  assign(inf,'cge.dat');
  reset(inf);
  while not eof(inf) do
    begin
      gotoxy(1,1);
      with coml do
        begin
          read(inf,command);
          case command of
            'p','w':begin
              readln(inf,xl,yl,color);
              write(char(27),command,xl,',',yl,',',color,';');
            end;
            'c':begin
              readln(inf,xl,yl,radius,color,arcstart,arcend,aspect);
              write(char(27),command,xl,',',yl,',',radius,',',color
                ',',arcstart,',',arcend,',',+aspect+');
            end;
            'l':begin
              readln(inf,xl,yl,x1,y1,x2,y2,color,box);
              write(char(27),command,xl,',',yl,',',x1,',',y1,'
                ,',x2,',',y2,',',color,box,';');
            end;
            't':begin
              readln(inf,xl,yl,color,gap,rxry,logic,keyintext);
            end;
            default: begin
              writeln('Unrecognized command');
            end;
          end;
        end;
    end;
end;
n:=1;
while copy(keyintext,n,1)=' ' do n:=n+1;
keyintext:=copy(keyintext,n,length(keyintext));
if logic=' *' then
  begin
    write(char(27),'t24,',xl,',',yl,','+gap+' +rxry+',');
    write(char(27),'sd',color,';',keyintext);
    write(char(27),'t24,',xl+l,',',yl+l,','+gap+', '+rxry +',x;');
    write(char(27),'sd',color,';',keyintext);
  end
else
  begin
    write(char(27),'t24,',xl,',',yl ',',+gap+, '+rxry +',+logic+');
    write(char(27),'sd',color,';',keyintext);
  end;
write(char(27),'tl6;');
end;
'e':begin
readln(inf,command,xl,y1,color,bcolor,keyintext);
n:=1;
while copy(keyintext,n,1)=' ' do n:=n+1;
keyintext:=copy(keyintext,n,length(keyintext));
gotoxy(xl,y1);
textbackground(bcolor);
textcolor(color);
write(keyintext);
textcolor(15);
textbackground(0);
end;
end;
end;
gotoxy(l,l);
close (inf);
end;

procedure clrdraw(var mode:char;coml:detail);
var
  tx,ty,tx2,ty2,tco:string[4];
  n,posofco :integer;
procedure getmvcolor;
begin
  tco:=copy(deline,posofco,3);
  n:=1;
  while copy(tco,n,1)=' ' do n:=n+1;
tco:=copy(tco,n,length(tco));
val(tco,coml.color,n);
coml.xl:=x;
coml.yl:=y;
end;

begin
  gotoxy(1,1);
  with coml do
    begin
      command:=copy(deline,2,1);
      tx:=copy(deline,3,4);
      n:=1;
      while copy(tx,n,1)=" " do n:=n+1;
      tx:=copy(tx,n,length(tx));
      ty:=copy(deline,7,4);
      n:=1;
      while copy(ty,n,1)=" " do n:=n+1;
      ty:=copy(ty,n,length(ty));
      val(tx,xl,n); val(ty,yl,n);
      color:=0;
      case command of
        'w': begin
          if mode='v' then begin posofco:=11; getmvcolor; end;
          write(char(27),command,xl,',',yl,',',color,';');
        end;
        'c': begin
          if mode='v' then begin posofco:=15; getmvcolor; end;
          write(char(27),command,xl,',',yl,’ ’+copy(deline,11,4)
               ’ ’,color,’ ’+copy(deline,18,6)+’ ’
               +copy(deline,24,6)+’ ’+copy(deline,30,4)+’ ’);
        end;
        'l': begin
          box:='';
          tx2:=copy(deline,11,4);
          n:=1;
          while copy(tx2,n,1)=" " do n:=n+1;
          tx2:=copy(tx2,n,length(tx2));
          ty2:=copy(deline,15,4);
          n:=1;
          while copy(ty2,n,1)=" " do n:=n+1;
          ty2:=copy(ty2,n,length(ty2));
          val(tx2,x2,n); val(ty2,y2,n);
          if copy(deline,24,1)="b" then box:=’b’;
          if copy(deline,24,1)="f" then box:=’bf’;
          if mode='v' then begin
            x2:=x+(x2-xl); y2:=y+(y2-yl);
            posofco:=19; getmvcolor;
          end;
      end;
      end;
    end;
  end;
end;
155

write(char(27),command,xl,',',yl,',',x2,',',y2,','
 ,color,box,';');
end;
't':begin
  if mode='v' then begin posofco:=ll; getmvcolor; end;
gap :=copy(deline,14,2);
rxry :=copy(deline,16,4);
logic:=copy(deline,20,2);
keyintext:=copy(deline,22,40);
n:=1;
while copy(keyintext,n,1)=' ' do n:=n+1;
keyintext:=copy(keyintext,n,length(keyintext));
if logic=' **' then
  begin
    write(char(27),'t24,',xl,',',yl,','+gap+','+rxry
         +',';');
    write(char(27),'sd',color,';',keyintext);
    write(char(27),'t24,',xl+l,',',yl+l,','+gap+','+rxry
         +','x;');
    write(char(27),'sd',color,';',keyintext);
  end
else
  begin
    write(char(27),'t24,',xl,',',yl,','+gap+','+rxry+
         +logic+';');
    write(char(27),'sd',color,';',keyintext);
  end;
write(char(27),'tl6; ')
end;
e':begin
  bcolor:=0;
  if mode='v' then begin posofco:=ll; getmvcolor;
    bcolor:=color;
    posofco:=ll; getmvcolor;
    xl:=stx; yl:=sty; end;
keyintext:=copy(deline,17,40);
n:=1;
while copy(keyintext,n,1)=' ' do n:=n+1;
keyintext:=copy(keyintext,n,length(keyintext));
gotoxy(xl,yl);
textbackground(bcolor);
textcolor(color);
write(keyintext);
textcolor(15);
textbackground(0);
end;
end;
xf:=xl; yf:=yl;
if command='l' then begin xs:=x2; ys:=y2 end;
procedure getsecondpoint(var com1:detail);
var
  special2, done2: boolean;
  ch : char;
begin
  done2 := false;
  repeat
    begin
      special2 := false;
      read(kbd, ch);
      if (ch = #27) then
        if keypressed then
          begin
            special2 := true;
            read(kbd, ch);
          end
        else
          done2 := true;
      end
      if not done2 then
        if special2 then
          begin
            case ord(ch) of
              75: x := x - ic;
              77: x := x + ic;
              72: y := y - ic;
              80: y := y + ic;
              71: begin x := x - ic; y := y - ic; end;
              81: begin x := x + ic; y := y + ic; end;
              73: begin x := x + ic; y := y - ic; end;
              79: begin x := x - ic; y := y + ic; end;
              82: if ic > 10 then ic := 10 else ic := 11 - ic;
              83: if ic < 50 then ic := 50 else ic := 150 - ic;
            end;
            if (x < 0) or (x > 573) then x := abs(574 - abs(x));
            if (y < 0) or (y > 340) then y := abs(341 - abs(y));
          end;
        textcolor(11);
        gotoxy(76, 24); write(x: 4);
        gotoxy(76, 25); write(y: 4);
        textcolor(15);
        gotoxy(1, 24); write(char(27), '1', lx - pic, ',', ly, ',', lx + pic, ',', ly , ',', 0, ',');
        write(char(27), '1', lx, ',', ly + pic, ',', lx, ',', ly + pic , ',', 0, ',');
        write(char(27), '1', x - pic, ',', y, ',', x + pic, ',', y, ',', 15, ',');
        write(char(27), '1', x, ',', y - pic, ',', x, ',', y + pic, ',', 15, ',');
      end;
    end;
  end;
end;
lx:=x;
ly:=y;
with cm1 do
begin
  write(char(27),'1*,xl-pic,',',yl,',',xl+pic,',',yl
  ',15,,');
  write(char(27),'l',xl,',yl-pic,',xl,',',yl+pic
  ',15,,');
end;
end
else
  case ord(ch) of
    13: done2:=true;
  end;
until done2;
end;

procedure findmovedelete(mode:modech);
var
  n,nx,ny,ncoml,nc,nd:integer;
  com:detail;
  tx,ty,tx2,ty2:string[4];
  deltype:string[12];
  com,ch:char;
  temf:text;
  order,distance;array[1..500] of integer;
  comarray;array[1..500] of string[100];
  special4,done4:boolean;
procedure sort;
var sortline :string[100];
  sortn,sortm,temp,sortx,sorty:integer;
  rx,ry,rsortx,rsorty:real;
begin
  for sortn:=1 to ncoml do
    begin
      sortline:=comarray[sortn];
      tx:=copy(sortline,3,4);
      n:=1;
      while copy(tx,n,1)=' ' do n:=n+1;
      tx:=copy(tx,n,length(tx));
      ty:=copy(sortline,7,4);
      n:=1;
      while copy(ty,n,1)=' ' do n:=n+1;
      ty:=copy(ty,n,length(ty));
      val(tx,sortx,n); val(ty,sorty,n);
if copy(sortline,2,1)='e' then begin
  sortx:=(sortx-1)*8;
  sorty:=(sorty-1)*15;
end;
rx:=x; ry:=y; rsortx:=sortx; rsorty:=sorty;
distance[sortn]:=round(sqrt(sqr(rx-rsortx)+sqr(ry-rsorty)));
end;
for sortn:=1 to ncoml do order[sortn]:=sortn;
for sortn:=1 to ncoml-1 do
  for sortm:=sortn+1 to ncoml do
    if distance[sortn]>distance[sortm] then
      begin
        temp:=order[sortn];
        order[sortn]:=order[sortm];
        order[sortm]:=temp;
        temp:=distance[sortn];
        distance[sortn]:=distance[sortm];
        distance[sortm]:=temp;
      end;
end;
begin
  assign(inf,'cge.dat');
  reset(inf);
  ch:='n';
  nc:=0; nd:=0;
  while not eof(inf) do
    begin
      readln(inf,deline);
      nc:=nc+1;
      comlarray[nc]:=deline;
    end;
  close(inf);
  ncoml:=nc; nc:=1;
  done4:=false;
  if ncoml>0 then
    begin
      sort;
      repeat begin
        special4 := false;
        deline:=comlarray[order[nc]];
        com:=copy(deline,2,1);
        case com of
          'w': deltype='Dot';
          'p': deltype='Paint';
          'l': begin
            deltype='Line';
            if copy(deline,24,1)= 'b' then deltype='Box';
            if copy(deline,24,1)= 'f' then deltype='PaintBox';
end;
'c': if copy(deline,29,1)='0' then deltype:='circle'
   else deltype:='Arc';
't': deltype:='Text 24';
'e': deltype:='Text 16';
end;

nx:=copy(deline,3,4);
n:=1;
while copy(nx,n,1)=' ' do n:=n+1;
   nx:=copy(nx,n,length(nx));

ny:=copy(deline,7,4);
n:=1;
while copy(ny,n,1)=' ' do n:=n+1;
   ny:=copy(ny,n,length(ny));

val(nx,nx,n); val(ny,ny,n);
gotoclr(snx2,sny24);
case mode of
   'd': writeln('Delete(y/n)? ');
   'f': writeln('Finded(y/n)? ');
   'm': writeln('Move (y/n)? ');
end;
gotoxy(30,24);
textcolor(6);
write('=P+v !tUP !uDOWN (S?o>, <Esc> 8u%X');
textcolor(10);
gotoclr(snx2,sny25);
write(deltype, ' ',copy(deline,3,length(deline)));
textcolor(15);
gotoxy(1,24);
if com='e' then
   if topline:=6; bottomline:=7;
   Cursor_Control(topLine,bottomLine);
   gotoxy(nx,ny);
   end
else begin
   write(char(27), 'c',nx,',',ny,',4,15');
   gotoxy(14,24);
end;
topline:=6; bottomline:=7;
Cursor_Control(topLine,bottomLine);
read(kbd,ch);
if (ch=#27) then
   if keypressed then
      begin
         special4 :=true;
         read(kbd,ch);
      end
else
done4:=true;
if not done4 then
if special4 then
begin
if ch=#72 then begin
nc:=nc-1;
if nc<l then nc:=ncoml;
while comarray[order[nc]]='' do
begin
nc:=nc-1;
if nc<l then nc:=ncoml;
end;
end;
if ch=#80 then begin
nc:=nc+1;
if nc>ncoml then nc:=1;
while comarray[order[nc]]='' do
begin
nc:=nc+1;
if nc>ncoml then nc:=1;
end;
end;
end
else
begin
if (ch=#78) or (ch=#110) then begin
nc:=nc+1;
if nc>ncoml then nc:=1;
while comarray[order[nc]]='' do
begin
nc:=nc+1;
if nc>ncoml then nc:=1;
end;
end;
if (ch=#89) or (ch=#121) then case mode of
'd': begin
clrdraw(mode,coml);
nd:=nd+1;
if nd=ncoml then done4:=true;
comarray[order[nc]]='';
if not done4 then
while comarray[order[nc]]='' do
begin
nc:=nc+1;
if nc>ncoml then nc:=1;
end;
end;
'f': begin
161

done4:=true; x:=nx; y:=ny;
if com='e' then begin
   topline:=6; bottomline:=7;
   Cursor_Control(topLine,bottomLine);
   x:=(nx-1)*8; y:=(ny-1)*15;
   topline:=48; bottomline:=0;
   Cursor_Control(topLine,bottomLine);
end;
end;
'm': begin
   gotoclr(snx2,sny24);
   write('=P2>0J4e<P?o>\%t$@7sBI&A&A+V RETURN :');
   with com1 do
      begin
         command:=copy(deline,2,1);
         tx:=copy(deline,3,4);
         n:=1;
         while copy(tx,n,1)=' ' do n:=n+1;
         tx:=copy(tx,n,length(tx));
         ty:=copy(deline,7,4);
         n:=1;
         while copy(ty,n,1)=' ' do n:=n+1;
         ty:=copy(ty,n,length(ty));
         val(tx,x,n); val(ty,y,n);
      end;
   textcolor(7);
   gotoxy(2,25);
   write('!t$W !u$U !v%* !w%k Home!x PgUp!y End!z PgDn!{ ');
   'Ins/Del:$A+$BI<F=',ic:=3);
   textcolor(15);
   gotoxy(1,24);
   if com1.command='e' then
      begin
         topline:=6; bottomline:=7;
         Cursor_Control(topLine,bottomLine);
         stx:=x; sty:=y;
         gettextxy;
         x:=lx; y:=ly;
         topline:=48; bottomline:=0;
         Cursor_Control(topLine,bottomLine);
      end
   else
      begin
         write(char(27),'l',x-pic,',',y,',',x+pic,',y,',15,',');
         write(char(27),'l',x,',',y-pic,',',x,',',y+pic,',15,',');
         getsecondpoint(com1);
      end;
   clrdraw(mode,com1);
   mode:='v';
clrdraw(mode,com1);
mode:= 'm';
with com1 do
begin
xl:=xf; yl:=yf;
str(xl,tx); str(yl,ty);
for n:=1 to 4-length(tx) do insert( '',tx,n);
for n:=1 to 4-length(ty) do insert( '',ty,n);
delete(deline,3,4); insert(tx,deline,3);
delete(deline,7,4); insert(ty,deline,7);
if command= '1' then
begin
x2:=xs; y2:=ys;
str(x2,tx2); str(y2,ty2);
for n:=1 to 4-length(tx2) do insert( '',tx2,n);
for n:=1 to 4-length(ty2) do insert( '',ty2,n);
delete(deline,11,4); insert(tx2,deline,11);
delete(deline,15,4); insert(ty2,deline,15);
end;
end;
com1array[order[nc]]:=deline;
nc:=nc+1; if nc>ncm then nc:=1;
end;
end;
end;
if com='e' then gotoxy(1,24)
else begin
write(char(27),'c',nx,',',ny,',',4,0;');
writeln;
end;
end until done4;
end
else
begin
gotoclr(snx2,sny2);
textcolor(14);
write('d$#(190'N!!!');
textcolor(15);
repeat until keypressed;
end;
case mode of
'd','m': begin
assign(temf,'cge.dat');
rewrite(temf);
for n:=1 to ncom1 do
if comarray[n]='' then writeln(temf,comarray[n]);
close(temf);
end;
procedure pickcolor(var coml:detail; t24:boolean);
var
  n     :integer;
  num,logset:set of char;
  ch    :char;
  err   :boolean;
begin
  logset:=['a','x','o','s','r','A','X','O','S','R','*'];
  num:=['0'..'9'];
  gotoclr(snx2,sny25);
  gotoxy(5,25);
  textbackground(7); textcolor(0); write('O'); textbackground(0);
  for n:=1 to 15 do
    begin
      textcolor(n);
      gotoxy((n+1)*3,25);
      write(n:3);
    end;
  gotoclr(snx2,sny24);
  write('=P?o>CC&b:*');
  with coml do
  begin
    color:=-1; bcolor:=0; box='';
    gap=''; rxry=''; logic='';
    repeat
      err:=false;
      topline:=6; bottomline:=7;
      Cursor_Control(topLine,bottomLine);
      if command='r' then readln(color,box)
      else if t24 then readln(color,gap,rxry,logic)
      else readln(color,bcolor);
    repeat
      topline:=48; bottomline:=0;
      Cursor_Control(topLine,bottomLine);
      if ioresult>0 then err:=true;
      if (copy(box,2,1)='f') or (copy(box,2,1)='F') then command:='f';
      if t24 then
        begin
          if length(gap)=2 then
            begin
              ch:=copy(gap,2,1);
              if not (ch in num) then err:=true;
else gap:='';
if length(rxry)=4 then
begin
  ch:=copy(rxry,3,1); if ch<>'' then err:=true;
  ch:=copy(rxry,2,1); if not (ch in num) then err:=true;
  ch:=copy(rxry,4,1); if not (ch in num) then err:=true;
end
else rxry:='';
if length(logic)=2 then
begin
  ch:=copy(logic,2,1);if not (ch in logset)
  then err:=true;
end
else logic:='';
end;
if err then
begin
  gotoxy(30,24);
textcolor(14);
write( '?i$J8j.F.fS.!$#2E! •);
textcolor(15);
  gotoxy(13,24);
end;
until err=false;
end;

procedure datatofile(var coml:detail);
begin
  with coml do
  if color<>1 then
  begin
    case command of
      'w':begin
      writeln(outf,command:2,xl:4,yl:4,color :3);
      write(char(27),command,xl,',',yl,',',color,';');
    end;
      '1':begin
      writeln(outf,command:2,xl:4,yl:4,x2:4,y2:4,color :3,' ';3)
      write(char(27),command,xl,',',yl,',',x2,',',y2,',',color,';')
    end;
      'r':begin
      command:='l';
      writeln(outf,command:2,xl:4,yl:4,x2:4,y2:4,color :3,'b':3);
      write(char(27),command,xl,',',yl,',',x2,',',y2,',',color,';b');
    end;
      'f':begin
      command:='';
    end;
  end;
end;
writeln(outf,command:2,x1:4,y1:4,x2:4,y2:4,color:3,'bf':3);
write(char(27),command,xl,'}',yl,'}',x2,'}',y2,'}',color,'}',bf');
end;

c':begin
writeln(outf,command:2,x1:4,y1:4,radius:4,color:3
',0':6,'0':6,aspect:4);
write(char(27),command,xl,',',yl,',',radius,',',color,',','
+aspect+');
end;
a':begin
command:='c';
writeln(outf,command:2,x1:4,y1:4,radius:4,color:3
,arcstart"10:6,arcend"10:6,aspect:4);
write(char(27),command,xl,'}',yl,'}',radius,'}',color
,'}',arcstart"10,'}',arcend10,'}',+aspect+');
end;
p':begin
writeln(outf,command:2,x1:4,y1:4,color:3);
write(char(27),command,xl,'}',yl,'}',color,'}');
end;
t':begin
writeln(outf,command:2,x1:4,y1:4,color:3,gap:2,rxry:4
,logic:2,keyintext:40);
if logic='*' then
begin
write(char(27),'t24,',xl,',',yl,',',+gap+','+rxry+','};
write(char(27),'sd',color,';',keyintext);
write(char(27),'t24,',xl+l,',',yl+l,',',+gap+','+rxry+','x'});
write(char(27),'sd',color,';',keyintext);
end
else
begin
write(char(27),'t24,',xl,',',yl,',',+gap+','+rxry+','
+logic+');
write(char(27),'sd',color,'}',keyintext);
end;
write(char(27),'tl6');
end;
e':begin
writeln(outf,command:2,x1:4,y1:4,color:3,bcolor:3,keyintext:40);
gotoxy(xl,yl);
textbackground(bcolor);
textcolor(color);
write(keyintext);
textcolor(15);
textbackground(0);
end;
end
else color:=15;
end;

procedure helptext;
begin
  window(3,2,70,23);
textcolor(14);
gotoxy(2,2);
writeln(' The CCGE system one-key command summary:');
writeln('================================================================================

Clr-scrn Redraw Save Load New-draw Delete Find Grid Move Include

================================================================================

The functions are as follows:

C : Clear the Screen);
R : Redraw the Screen');
S : Save Screen and Create Pascal Instructions');
L : Load a Screen from Disk (Restore the Screen');
N : Begin a New Drawing');
D : Delete a Graphic Object');
F : Find a Location of a Graphic Object');
G : Draw grids on Screen');
M : Move a Graphic Object to a New Location');
I : Include another screen file to the current screen');
P : Change the size of cursor, you need this when painting');
repeat until keypressed;
textcolor(10);
write(char(27),'11,1,573,340,0,bf;');
gotoxy(3,2);
writeln(' The CCGE system uses a special input format for users to');
writeln('define their special needs. This special function can be');
writeln('used for the functions such as Box, 16x16 text, and');
writeln('24x24 text. The parameters for each graphic function are');
writeln('given after color number. It means that you can used the');
writeln('parameters after the color number to define your special');
writeln('parameters after color #. Examples of the format are');
writeln('listed below:

for (3) box color : 13 f   ( f will draw a filled box)');
writeln('for (7) 16x16 text : 14 6   (the first 14 for textcolor)');
writeln('the 6 for background color)');
writeln('for (7) 24x24 text : 11 2 2/3 *');
writeln(' where 11 is a color number');
writeln(' 2 is the gap between each 24x24 text');
writeln(' 2/3 are digits for x and y ratio');
writeln(' * is logic expression, other options for');
writeln(' the logic are S(set), R(reverse), A(and)');
writeln(' O(or), X(XOR), and *(stereo).');
textcolor(12);
gotoxy(23,12); write('13 f');
gotoxy(23,13); write('14 6');
gotoxy(23,15); write('11 2 2/3 *');
window(1,1,80,25);
gotoxy(1,24);
repeat until keypressed;
end;

procedure frame;
var
  menuword:array[1..10] of string[20];
i :integer;
begin
  menuword[1] := '1.5eBI';
  menuword[2] := '2.5e=u';
  menuword[3] := '3./x'N';
  menuword[4] := '4.5e6j';
  menuword[5] := '5.6j)7';
  menuword[6] := '6.6n&b';
  menuword[7] := '7.$e&r';
  menuword[8] := '8.52't';
  menuword[9] := '1m&bC890=s?h(t2N';
  clrscr;
  writeln(char(27),'10,0,574,341,15,b;');
  writeln(char(27),'1578,0,639,341,15,b;');
  writeln(char(27),'10,344,574,376,15,b;');
  writeln(char(27),'1578,344,639,376,15,b;');
  for i:=1 to 8 do
    begin
      gotoxy(74,i+l);
      write(menuword[i]);
    end;
  for i:=1 to 8 do
    begin
      writeln(char(27),'t24,606,'116+i"24','1/1;');
      writeln(char(27),'sd13;',copy(menuword[9],l+(i-l)*2,2));
      writeln(char(27),'t16;');
      gotoxy(1,1);
    end;
  for i:=1 to 6 do
begin
  textcolor(15);
  gotoxy(74,16+i);
  writeln(copy(menuword[10],1+(i-1)*2,2));
end;
end;

procedure getxy;
var
  special, done : boolean;
  ch, check : char;
  pointdraw : boolean;
  texttype, n : integer;
  rx, ry, rxl, ryl : real;
  gx, gy, gc : integer;
  num : set of char;
begin
  textcolor(11);
  gotoxy(74,24); write('X= 287');
  gotoxy(74,25); write('Y= 170');
  textcolor(15);
  gotoxy(1,24);
  stx:=36; sty:=11; tic:=1; gc:=0; ch:=' ';
x:=287; y:=170; ic:=1; lx:=x; ly:=y; pic:=2;
  snx2:=2; sny24:=24; sny25:=25;
t24:=false;
  num:='[0..'9']';
  write(char(27),'l',x-pic,',',y,,x+pic,',',y,',15,,');
  write(char(27),'l',x,',',y-pic,',',x,',y+pic,',15,,');
  done:=false;
repeat begin
  gotoxy(2,24);
  if ord(ch)<59 then
    write(E'=P%}2>0J4e<P(M)w&18m!A&A?o>\C89O<R&!(1!c8)?'
    '');
  textcolor(7);
  gotoxy(2,25);
  write(Alt$W !u$U !v%* !w%k Home!x PgUp!y End!z PgDn!{ ',
    'Ins/Del:$A4+BI<F=',ic:3);
  textcolor(15);
  special := false;
  topline:=6; bottomline:=7;
  Cursor_Control(topLine,bottomLine);
  read(kbd,ch);
  topline:=48; bottomline:=0;
  Cursor_Control(topLine,bottomLine);
  if (ch=#27) then
    if keypressed then
      begin

special := true;
read(kbd, ch);
end
else
begin done := true; gotoxy(1, 24); end;
if not done then
if special then
begin
case ord(ch) of
  59: begin
    gotoclr(snx2, sny24);
    write('Clr-screen Redraw Save Load New-draw Delete'
          ' Find Grid Move Include');
    textcolor(13);
    gotoxy(2, 24); write('C');
    gotoxy(13, 24); write('R');
    gotoxy(20, 24); write('S');
    gotoxy(25, 24); write('L');
    gotoxy(30, 24); write('N');
    gotoxy(39, 24); write('D');
    gotoxy(46, 24); write('F');
    gotoxy(51, 24); write('G');
    gotoxy(56, 24); write('M');
    gotoxy(61, 24); write('I');
    textcolor(15);
  end;
  60: begin
    gotoxy(1, 24);
    write(char(27), 'l', x-pic, ',', y, ',', x+pic, ',', y, ',', 15,,');
    helptext;
    repeat until keypressed;
    gotoxy(1, 24);
    write(char(27), 'l', x, ',', 'y-pic, ', 'x, ', 'y+pic, ', '15,,');
    end;
    75: x := x - ic;
    77: x := x + ic;
    72: y := y - ic;
    80: y := y + ic;
    71: begin x := x - ic; y := y - ic; end;
    81: begin x := x + ic; y := y + ic; end;
    73: begin x := x + ic; y := y - ic; end;
    79: begin x := x - ic; y := y + ic; end;
    82: if ic > 10 then ic := 10 else ic := 11 - ic;
    83: if ic < 50 then ic := 50 else ic := 150 - ic;
if (x<0) or (x>573) then x := abs(574-abs(x));
if (y<0) or (y>340) then y := abs(341-abs(y));
textcolor(11);
gotoxy(76,24); write(x:4);
gotoxy(76,25); write(y:4);
textcolor(15);
gotoxy(1,24);
if pointdraw=true then
begin
write(char(27),'1',lx-pic,',',ly,',',lx+pic,',',ly,',15,,;');
write(char(27),'1',lx,',',ly-pic,',',lx,',',ly+pic,',15,,;');
pointdraw:=false;
end
else
begin
write(char(27),'1',lx-pic,',',ly,',',lx+pic,',',ly,',0,,;');
write(char(27),'1',lx,',',ly-pic,',',lx,',',ly+pic,',0,,;');
end;
write(char(27),'1',x-pic,',',y,,x+pic,',',y,',15,,;');
write(char(27),'1',x,',',y-pic,',',x,',',y+pic,',15,,;');
lx:=x;
ly:=y;
end
else
{1.dot} case ord(ch) of
49: begin
with coml do
begin
xl:=x; yl:=y;
pointdraw:=true;
pickcolor(coml,t24);
command:='w';
end;
datatofile(coml);
end;

{2.line} 50: begin
with coml do
begin
xl:=x; yl:=y;
gotoclr(snx2,sny24);
write('P2>0J4e<P?o>\&t$@BI!A&A+v RETURN :');
getsecondpoint(coml);
x2:=x; y2:=y;
pickcolor(coml,t24);
command:='l';
end;
datatofile(coml);
end;
{3.rect} 51: begin
  with coml do
  begin
    xl:=x; yl:=y;
    gotocl(snx2,sny24);
    write('P2>0J4e<Po>\9o($BI!A&A+v RETURN :');
    getsecondpoint(coml);
    x2:=x; y2:=y;
    command:='r';
    pickcolor(coml,t24);
    end;
    datatofile(coml);
  end;

{4.circle} 52: begin
  with coml do
  begin
    xl:=x; yl:=y;
    gotocl(snx2,sny24);
    write('P2>0J4e<Po>\%b.|!A&A+v RETURN :');
    getsecondpoint(coml);
    rx:=x; ry:=y; rxl:=xl; ryl:=yl;
    radius:=round(sqrt(sqr(rx-rxl)+sqr((ry-ryl)*50/36)));
    gotocl(snx2,sny24);
    aspect='';
    write('P?iSJ#A;P#B*:Sq-H:');
    topline:=6; bottomline:=7;
    Cursor_Control(topLine,bottomLine);
    readln(aspect);
    topline:=48; bottomline:=0;
    Cursor_Control(topLine,bottomLine);
    if length(aspect)=3 then
      begin
        check:=copy(aspect,2,1);
        if check<>'/" then aspect='';
        check:=copy(aspect,1,1);
        if not (check in num) then aspect='';
        check:=copy(aspect,3,1);
        if not (check in num) then aspect='';
      end
    else aspect='';
    {*
      pickcolor(coml,t24);
      command='c';
      write(char(27),'1',x-pic,',',y,',',x+pic,',',y,',0,,;')
      write(char(27),'1',x,,y-pic,',',x,,y+pic,',0,,;')
    end;
    datatofile(coml);
    textcolor(ll);
gotoxy(76,24); write(x:4);
gotoxy(76,25); write(y:4);
textcolor(15);
gotoxy(1,24);
end;

{5.arc} 53: begin
with coml do
begin
xl:=x; yl:=y;
gotoclr(snx2,sny24);
write('PZ>04e<P?o>\%b.\|!A&A+v RETURN :');
getsecondpoint(coml);
radius:=round(sqrt(sqr(x-xl)+sqr(y-yl)));
topline:=6; bottomline:=7;
Cursor_Control(topLine,bottomLine);
gotoclr(snx2,sny24);
write('6)70 B(0-360):'); readln(arcstart);
gotoxy(24,24);
write('6)72 WBI(0-360):'); readln(arcend);
gotoxy(48,24);
aspect:='';
write('P?isJ#A;P#B*:S-H:');
readln(aspect);
topline:=48; bottomline:=0;
Cursor_Control(topLine,bottomLine);
if length(aspect)=3 then
begin
  check:=copy(aspect,2,1);
  if check<>'/'
  then aspect:='';
  check:=copy(aspect,1,1);
  if not (check in num)
  then aspect:='';
  check:=copy(aspect,3,1);
  if not (check in num)
  then aspect:='';
end
else aspect:='';
pickcolor(coml,t24);
command:='a';
write(char(27),'l',x-pic,','y,,x+pic,','y,',0,,');
write(char(27),'l',x,,y-pic,',',x,',y+pic,0,,');
x:=xl; y:=yl;
end;
datatofile(coml);
textcolor(11);
gotoxy(76,24); write(x:4);
gotoxy(76,25); write(y:4);
textcolor(15);
gotoxy(1,24);
end;

{6.Paint} 54: begin
with coml do
begin
    xl:=x; yl:=y;
pickcolor(coml,t24);
    command:='p';
end;
write(char(27),'l',x-pic,',x+pic,',0,,;')
write(char(27),'l',x,',y-pic,', ',x,,y+pic,'',',y+pic,','0,,;');
datatofile(coml);
end;

{7.text} 55: begin
topleft:=6; bottomleft:=7;
Cursor_Control(topleft,bottomleft);
gotoclr(sn2x,sn2y);
write('l. 16x16  2. 24x24:');
readln(texttype);
case texttype of
1: begin
    gotoclr(sn2x,sn2y);
    write('P2>0JE<P?0>&17m!A&A+v RETURN :');
    gettextxy;
    with coml do
    begin
        xl:=stx; yl:=sty;
        gotoclr(sn2x,sn2y);
        write('P?isJ$e&:');
        toplines:=6; bottomlines:=7;
        Cursor_Control(toplines,bottomlines);
        readln(keyintext);
pickcolor(coml,bottomlines);
        command:='e';
    end;
datatofile(coml);
textcolor(11);
gotoxy(76,24); write(x:4);
gotoxy(76,25); write(y:4);
textcolor(15);
gotoxy(1,24);
end;
2: begin
    with coml do
    begin
        xl:=x; yl:=y;
        gotoclr(sn2x,sn2y);
        write('P?isJ$e:&');
        toplines:=6; bottomlines:=7;
        Cursor_Control(toplines,bottomlines);
        readln(keyintext);
t24:=true;
pickcolor(com1,t24);
t24:=false;
command:='t';
end;
datatofile(com1);
end;

{8.exit} 56: begin done:=true; gotoxy(1,24); end;
{ C } 67,99: begin
gotoxy(1,24);
write(char(27),'11,1,573,340,0,bf;');
write(char(27),'1',x-pic,',',y,,x+pic,,,y,,,15,,,');
write(char(27),'1',x,,,y-pic,,,x,,,y+pic,,,15,,,');
end;

{ R } 82,114:begin
gotoxy(1,24);
write(char(27),'11,1,573,340,0,bf;');
close(outf);
redraw;
append(outf);
write(char(27),'1',x-pic,,,y,,,x+pic,,,y,,,15,,,');
write(char(27),'1',x,,,y-pic,,,x,,,y+pic,,,15,,,');
end;

{ D } 68,100:begin
close(outf);
mode:='d';
findmovedelete(mode);
mode:=' ';
gotoxy(1,24);
write(char(27),'11,1,573,340,0,bf;');
redraw;
append(outf);
write(char(27),'1',x-pic,,,y,,,x+pic,,,y,,,15,,,');
write(char(27),'1',x,,,y-pic,,,x,,,y+pic,,,15,,,');
end;

{ P } 80,112:begin
gotoxy(1,24);
write(char(27),'1',x-pic,,,y,,,x+pic,,,y,,,0,,,');
write(char(27),'1',x,,,y-pic,,,x,,,y+pic,,,0,,,');
pic:=2-pic;
write(char(27),'1',x-pic,,,y,,,x+pic,,,y,,,15,,,');
write(char(27),'1',x,,,y-pic,,,x,,,y+pic,,,15,,,');
end;

{ M } 77,109:begin
close(outf);
mode:='m';
175

findmovedelete(mode);
mode='f';
gotoxy(1,24);
append(outf);
write(char(27),'l',x-pic,',',y,',',x+pic,',',y,',15,,;')
write(char(27),'l',x,',',y-pic,',',x,',',y+pic,',15,,;')
end;
{ F } 70,102:begin
    close(outf);
    mode='f';
    findmovedelete(mode);
    mode='f';
    append(outf);
    write(char(27),'l',lx-pic,',',ly,',',lx+pic,',',ly,',0,,;')
    write(char(27),'l',lx,',',ly-pic,',',lx,',',ly+pic,',0,,;')
    write(char(27),'l',x-pic,',',y,',',x+pic,',',y,',15,,;')
    write(char(27),'l',x,',',y-pic,',',x,',',y+pic,',15,,;')
end;
{ S } 83,115:begin
    close(outf);
    savetopas;
    append(outf);
    write(char(27),'l',x-pic,',',y,',',x+pic,',',y,',15,,;')
    write(char(27),'l',x,',',y-pic,',',x,',',y+pic,',15,,;')
end;
{ L } 76,108:begin
    close(outf);
    mode='l';
    loadfmdat(mode);
    mode='f';
    if rslt <=0 then
        begin
            gotoxy(1,24);
            write(char(27),'l',1,573,340,0,bf;');
            redraw;
        end;
    append(outf);
    write(char(27),'l',x-pic,',',y,',',x+pic,',',y,',15,,;')
    write(char(27),'l',x,',',y-pic,',',x,',',y+pic,',15,,;')
end;
{ I } 73,105:begin
    close(outf);
    mode='i';
    loadfmdat(mode);
    mode='f';
    if rslt <=0 then
        begin
            gotoxy(1,24);
            write(char(27),'l',1,573,340,0,bf;');
        end;
redraw;
end;
append(outf);
write(char(27),'l',x-pic,',',y,',',x+pic,',',y,',15,,;');
write(char(27),'l',x,',',y-pic,',',x,',',y+pic,',15,,;');
end;

\{ G \}
71,103:begin
  gc:=gc+1; if gc>4 then gc:=1;
gotoxy(1,1);
case gc of
  1:begin
    for gx:=1 to 573 div 32 do
      write(char(27),'l',gx*32,',1,',gx*32,',340,4;');
    for gy:=1 to 340 div 15 do
      write(char(27),'l',gy*15,',573,',gy*15,',4;');
    end;
  2:begin
    for gx:=1 to 11 do
      write(char(27),'l',gx*50,',1,',gx*50,',340,7;');
    for gy:=1 to 6 do
      write(char(27),'l',gy*50,',573,',gy*50,',7;');
    end;
  3:begin
    for gx:=1 to 573 div 32 do
      write(char(27),'l',gx*32,',1,',gx*32,',340,0;');
    for gy:=1 to 340 div 15 do
      write(char(27),'l',gy*15,',573,',gy*15,',0;');
      write(char(27),'11,150,573,150,7;');
      write(char(27),'11,300,573,300,7;');
    end;
  4:begin
    for gx:=1 to 11 do
      write(char(27),'l',gx*50,',1,',gx*50,',340,0;');
    for gy:=1 to 6 do
      write(char(27),'l',gy*50,',573,',gy*50,',0;');
    end;
  end;
end;

\{ N \}
78,110:begin
  gotoclr(snx2,sny24);
topline:=6; bottomline:=7;
Cursor_Control(topLine,bottomLine);
write(‘+-7sC8;s7s90 (y/n) ?’);
readln(check);
topline:=48; bottomline:=0;
Cursor_Control(topLine,bottomLine);
if (check='y') or (check='Y') then
  begin
    write(char(27),'11,1,573,340,0,bf;');
close(outf);
rewrite(outf);
end;
end;
end;
end;
until done;
end;

procedure title;
begin
clrscr;
write(char(27),'187,31,562,321,10, b');
write(char(27),'1572,331,77,21,10, b');
write(char(27),'t24,167,50, 2, 2/2, ;');
write(char(27),'sd15;9g8#;2'ULP>G');
write(char(27),'t24,168,51, 2, 2/2, x');
write(char(27),'sd15;9g8#;2'ULP>G');
write(char(27),'t16;');
write(char(27),'t24,126,167, 2, 2/2, ;');
write(char(27),'sd14;1m&bC890=s?h(t2N');
write(char(27),'t16;');
write(char(27),'t24,226,277, , , , ;)';
write(char(27),'sd15;@"L: 7E9E:a');
write(char(27),'t16;');
gotoxy(49,20);
textbackground(3);
textcolor(15);
write('1988-9/1');
textcolor(15);
textbackground(0);
repeat until keypressed;
writeln;
end;

begin
topline:=48; bottomline:=0;
Cursor_Control(topLine,bottomLine);
title;
assign(outf,'cge.dat');
rewrite(outf);
frame;
getxy;
close(outf);
topline:=6; bottomline:=7;
Cursor_Control(topLine,bottomLine);
end.
Appendix H.

Lesson Presentation Program List (CCAS Program)

program ChineseCAI_system;

{ Program coded by Jerome Wem. 12/30/1988 at ISU }

type stringtype=string[10];
var script: text;
  commandstring, parameter1, parameter2: stringtype;
  testnumber, voicenumber: integer;
  begin, ending, i, n, x, y: integer;
  buf: array[1..7, 1..8192] of byte;
  nomeaning: string[8];
{$I test.inc}

procedure ReadCGEfile(cgepara: stringtype);

  type
detail = record
    command: char;
    x1, y1, x2, y2: integer;
    radius: integer;
    color, bcolor: integer;
    box: string[5];
    gap, logic: string[2];
    rxry, aspect: string[4];
    arcstart, arcend: integer;
    keyintext: string[80];
  end;
  modech = char;

  var
    outf, inf: text;
    x, y, n: integer;
    coml: detail;
    cgefilename: string[14];

  begin
    cgefilename := cgepara + '.dat';
    assign(inf, cgefilename);
    reset(inf);
    while not eof(inf) do
      begin
        gotoxy(l, l);
        with coml do
          begin
            while not eof(inf) do
              begin
                gotoxy(l, l);
                with coml do
                  begin
                    if
                  end
                end
            end
          end
        end
      end
    end
}
read(inf,command);
case command of
  'w':begin
    readln(inf,xl,yl,color);
    write(char(27),command,xl,','+yl,'',color,';');
  end;
  'p':begin
    readln(inf,xl,yl,color);
    write(char(27),command,xl,','+yl,'',color,';');
  end;
  'c':begin
    readln(inf,xl,yl,radius,color,arcstart,arcend,aspect);
    write(char(27),command,xl,','+yl,'',radius,','+color,'',arcstart,','+arcend,'',aspect,';');
  end;
  'l':begin
    readln(inf,xl,yl,x2,y2,color,box);
    write(char(27),command,xl,','+yl,'',x2,','+y2,'',color,box,';');
  end;
  't':begin
    readln(inf,xl,yl,color,gap,rxry,logic,keyintext);
    n:=1;
    while copy(keyintext,n,1)=' ' do n:=n+1;
    keyintext:=copy(keyintext,n,length(keyintext));
    if logic=' *' then
      begin
        write(char(27),'t24,',xl,','+yl,'',gap,'',rxry,';');
        write(char(27),'sd',color,';',keyintext);
        write(char(27),'t24,',xl+n,','+yl,'',gap,'',rxry,';');
        write(char(27),'sd',color,';',keyintext);
      end
    else
      begin
        write(char(27),'t24,',xl,','+yl,'',gap,'',rxry,'',logic,';');
        write(char(27),'sd',color,';',keyintext);
      end;
    write(char(27),'t16;');
  end;
  'e':begin
    readln(inf,command,xl,yl,color,bcolor,keyintext);
    n:=1;
    while copy(keyintext,n,1)=' ' do n:=n+1;
    keyintext:=copy(keyintext,n,length(keyintext));
    gotoxy(xl,yl);
    textbackground(bcolor);
    textcolor(color);
    write(keyintext);
    textcolor(15);
    textbackground(0);
procedure SlideControl(slidpara:stringtype; slidenum:integer);
var i,j,x,n,r1,r2:integer;
begin
  if slidpara=' on' then
    begin
      port[$3e3]:=8b; delay(10);
      port[$3e0]:=80; delay(10);
    end;
  if slidpara=' off' then
    begin
      port[$3e3]:=8b; delay(10);
      port[$3e0]:=00; delay(10);
    end;
  if slidenum=0 then i:=1 else i:=slidenum;
  if slidpara=' forward' then
    for x:=1 to i do
      begin
        port[$3e3]:=8b; delay(10);
        port[$3e0]:=80; delay(1000);
        port[$3e0]:=8E; delay(100);
        port[$3e0]:=80;
      end;
  if slidpara=' backward' then
    for x:=1 to i do
      begin
        port[$3e3]:=8b; delay(10);
        port[$3e0]:=80; delay(1000);
        port[$3e0]:=8D; delay(100);
        port[$3e0]:=80;
      end;
end;

procedure Vcrinit;
var i,j,x,n:integer;
begin
  x:=0;
  n:=-1;
  port[$3e3]:=8b; delay(10);
  port[$3e0]:=06; delay(500);
  port[$3e0]:=44; delay(100);
  repeat
    n:=x;
  until...
delay(100);
x := port[$3e1]*256+port[$3e2];
until x=n;
port[$3e0] := $00; delay(500);
port[$3e0] := $07; delay(1000);
port[$3e0] := $00; delay(100);
port[$3e0] := $07; delay(1000);
port[$3e0] := $00; delay(100);
port[$3e0] := $08; delay(200);
port[$3e0] := $00; delay(100);
end;

procedure fastforward(endaddress:integer);
var x:integer;
begin
  port[$3e0] := $03; delay(100);
  port[$3e0] := $43; delay(100);
  repeat
    x := port[$3e1]*256+port[$3e2];
    until x>=endaddress*3;
  port[$3e0] := $01; delay(200);
end;

procedure fastbackward(endaddress:integer);
var x:integer;
begin
  port[$3e0] := $04; delay(100);
  port[$3e0] := $44; delay(100);
  repeat
    x := port[$3e1]*256+port[$3e2];
    until x<=endaddress*3;
  port[$3e0] := $01; delay(200);
end;

procedure vcrplay(startadd,endadd:integer);
var i,j,x:integer;
begin
  i := startadd; j := endadd;
  if i<j then
    begin
      port[$3e3] := $8b; delay(10);
      x := port[$3e1]*256+port[$3e2];
      if x + 6 < i * 3 then fastforward(startadd);
      if x - 6 > i * 3 then fastbackward(startadd);
      delay(500);
      port[$3e0] := $42; delay(3500);
      port[$3e0] := $72;
      repeat
        x := port[$3e1]*256+port[$3e2];
        until x = n;
    end;

until x=j*3;
port[$3e0] := $01; delay(200);
port[$3e0] := $00; delay(10);
end;
end;

procedure voiceplay;
var x:integer;
begin
  port[$3e7] := $90;
  port[$3e6] := $a0;
  for x:=1 to 2000 do begin end;
  port[$3e6] := $c0;
end;

procedure loadvoice(parameter: stringtype);
var inpfile: File;
  numr, numw: integer;
  filen: string[14];
begin
  filen := parameter+'.VOD';
  assign(inpf, filen);
  reset(inpf, 1);
  numr := 8192*7;
  blockread(inpf, buf, numr);
  close(inpf);
end;

procedure voice(n: integer);
var i, j: integer;
begin
  port[$3e7] := $80;
  for i:=0 to 31 do
  begin
    port[$3e6] := i+96;
    for j:=0 to 255 do
    begin
      port[$3e5] := j;
      port[$3e4] := buf[n, i*256+j];
      port[$3e6] := I+64;
      port[$3e6] := I+96;
    end;
  end;
  port[$3e7] := $90;
  port[$3e6] := $f0;
end;

function value(parameter: stringtype): integer;
var i, v: integer;
begin
  i := 0;
  repeat
    i := i + 1
    until copy(parameter, i, 1) <> ' ';
  val(copy(parameter, i, length(parameter) - i + 1), v, n);
  value := v;
end;

function nospace(parameter: stringtype): stringtype;
var i, n: integer;
begin
  i := 0;
  repeat
    i := i + 1
    until copy(parameter, i, 1) <> ' ';
  nospace := copy(parameter, i, length(parameter) - i + 1);
end;

begin
  assign(script, 'cai.spt');
  reset(script);
  while not eof(script) do
    begin
      readln(script, commandstring, parameter1, parameter2);
      if commandstring = ' test' then
        begin
          testnumber := value(parameter1);
          test(testnumber);
        end;
      if commandstring = ' cgefile' then
        begin
          parameter1 := nospace(parameter1);
          Readcgefile(parameter1);
        end;
      if commandstring = ' loadvoice' then
        begin
          parameter1 := nospace(parameter1);
          loadvoice(parameter1);
        end;
      if commandstring = ' voice' then
        begin
          voicenumber := value(parameter1);
          voice(voicenumber);
          voiceplay;
          delay(4500);
        end;
    end;
end;
if commandstring='vcrinit' then
  Vcrinit;
if commandstring='vcrfastfrd' then
  begin
    testnumber:=value(parameter1);
    fastforward(testnumber);
  end;
if commandstring='vcrfastrew' then
  begin
    testnumber:=value(parameter1);
    fastbackward(testnumber);
  end;
if commandstring=' vcrplay' then
  begin
    beging:=value(parameter1);
    ending:=value(parameter2);
    vcrplay(beging,ending);
  end;
if commandstring=' slide' then
  begin
    testnumber:=value(parameter2);
    SlideControl(parameter1,testnumber);
  end;
if commandstring=' clrscreen' then
  clrscr;
if commandstring=' delay' then
  begin
    testnumber:=value(parameter1);
    delay(testnumber*1000);
  end;
if commandstring=' waitkey' then
  repeat until keypressed;
if commandstring='waitreturn' then
  readln(nomeaning);
if commandstring=' beep' then
  write(char(7));
end;
close(script);
end.
Appendix I.

Voice Playback Program List

program voiceplay;

{Filename : VOICEPLY.PAS by Jerome Wen}
{This program retrieve the voice data from a voice *.VOD file.}
{It can play 7 segments of voice.}

var n:integer;
    buf:array[1..7,1..8192] of byte;

procedure voiceplay;
var x:integer;
begin
    port[$3e7]:=$90;
    port[$3e6]:=$a0;
    for x:=1 to 2000 do begin end;
    port[$3e6]:=$c0;
end;

procedure loadvoice;
var inpf:file;
    numr,numw:integer;
    filen:string[14];
begin
    write('Please key in the name of voice data file ' ,
         'you want to play:');
    readln(filen);
    assign(inpf,filen);
    reset(inpf,1);
    numr:=8192*7;
    blockread(inpf,buf,numr);
    close(inpf);
end;

procedure segment(n:integer);
var i,j:integer;
begin
    port[$3e7]:=$80;
    for i:=0 to 31 do begin
        port[$3e6]:=i+96;
        for j:=0 to 255 do
begin
    port[$3e5] := j;
    port[$3e4] := buf[n, i * 256 + j];
    port[$3e6] := I + 64;
    port[$3e6] := I + 96;
end;

port[$3e7] := $90;
port[$3e6] := $f0;
end;

begin
    loadvoice;
    repeat
        write('please input the segment (1-7, 0=exit): ');
        readln(n);
        if n <= 0 then
            begin
                segment(n);
                voiceplay;
            end
        until n = 0;
end.
Appendix J.

VCR and Slide Control Routines

J.1 VCR Initial Program (Filename: VCRINIT.PAS)

program VcrInit;

{ Filename: VcrInit by Jerome Wen }
{ This program rewind the VCR to the very beginning }
{ and reset the VCR counter. }
{ This command must be used before you }
{ control the VCR. }

var i,j,x,n:integer;

begin
  x:=0;
  n:=-1;
  port[$3e3] := $8b; delay(10);
  port[$3e0] := $06; delay(500);
  port[$3e0] := $44; delay(100);
  repeat
    n:=x;
    delay(100);
    x:=port[$3e1]*256+port[$3e2];
  until x=n;
  port[$3e0] := $00; delay(500);
  port[$3e0] := $07; delay(1000);
  port[$3e0] := $00; delay(100);
  port[$3e0] := $07; delay(100);
  port[$3e0] := $00; delay(100);
  port[$3e0] := $08; delay(200);
  port[$3e0] := $00; delay(100);
end.
program VcrFastForward;

{ Filename: VCRFWDTO.PAS by Jerome Wen }
{ Program fast forward VCR to the address selected }

var i,j,x,n,r1,r2:integer;
   err:boolean;

procedure parseline;
begin
   err:=false;
   val(paramSTR(1),i,r1);
   if (ParamCount=0) or (r1<>0) then err:=true;
end;

procedure fastforward;
begin
   port[$3e0]:=$3; delay(100);
   port[$3e0]:=$43;delay(100);
   repeat
      x:=port[$3e1]*256+port[$3e2];
      until x>=i*3;
   port[$3e0]:=$01; delay(200);
end;

begin
   parseline;
   if not err then
      begin
         port[$3e3]:=$8b; delay(10);
         x:=port[$3e1]*256+port[$3e2];
         if x+6<i*3 then fastforward;
         port[$3e0]:=$00; delay(10);
      end;
end.
J.3 VCR Rewind Program (Filename: VCRREWTO.PAS)

program VcrFastBackward;
{
Filename: VCRREWTO.PAS by Jerome Wen
Program fast backward VCR to the address selected
}
var i,j,x,n,r1,r2:integer;
   err:boolean;
procedure parseline;
beginn
   err:=false;
   vaKparamSTR(1,i,r1);
   if (ParamCount=0) or (r1<>0) then err:=true;
end;
procedure fastbackward;
beginn
   port[$3e0]:=$04; delay(100);
   port[$3e0]:=$44; delay(100);
   repeat
      x:=port[$3e1]*256+port[$3e2];
      until x<=i*3;
      port[$3e0]:=$01; delay(200);
end;

begin
   parseline;
   if not err then
   beginn
      port[$3e3]:=$8b; delay(10);
      x:=port[$3e1]*256+port[$3e2];
      if x+6>i*3 then fastbackward;
      port[$3e0]:=$00; delay(10);
   end;
end.
program VcrPlayCom;

{ Filename : VCRPLAY.PAS by Jerome Wen }
{ This program select the specified address and play it. }

var i, j, x, n, r1, r2: integer;
  err: boolean;

procedure parseline;
begin
  err := false;
  val(paramSTR(1), i, r1);
  val(paramSTR(2), j, r2);
  if (ParamCount = 0) or (r1 <> 0) or (r2 <> 0) then err := true;
end;

procedure fastforward;
begin
  port[$3e0] := $03; delay(100);
  port[$3e0] := $43; delay(100);
  repeat
    x := port[$3e1] * 256 + port[$3e2];
    until x >= i * 3;
  port[$3e0] := $01; delay(200);
end;

procedure fastbackward;
begin
  port[$3e0] := $04; delay(100);
  port[$3e0] := $44; delay(100);
  repeat
    x := port[$3e1] * 256 + port[$3e2];
    until x <= i * 3;
  port[$3e0] := $01; delay(200);
end;

begin
  parseline;
  if not err then begin
    port[$3e3] := $8b; delay(10);
    x := port[$3e1] * 256 + port[$3e2];
    if x + 6 < i * 3 then fastforward;
  end;
if x-6>i*3 then fastbackward;
delay(500);
port[$3e0]:=$42; delay(3500);
port[$3e0]:=$72;
repeat
  x:=port[$3e1]*256+port[$3e2];
until x=j*3;
port[$3e0]:=$01; delay(200);
port[$3e0]:=$00; delay(10);
end;
end.
program SlideControl;

{ Filename : SLIDE.PAS by Jerome Wen }
{ This program control the functions of the slide project. }
{ This program must be running on DOS }
{ Parameters are: On, Off, forward, backward, and # }

var i,j,x,n,rl,r2:integer;
err:boolean;

procedure parseline;
begin
err:=false;
rl:=0;
if (ParamCount=2) then val(paramStr(2),i,rl);
if (ParamCount=0) or (rl<>0) then err:=true;
end;

begin
parseline;
if not err then
begin
  if (paramstr(1)='on') or (paramstr(1)='ON') then begin
    port[$3e3]:=$8b; delay(10);
    port[$3e0]:=$80; delay(10);
  end;
  if (paramstr(1)='off') or (paramstr(1)='OFF') then begin
    port[$3e3]:=$8b; delay(10);
    port[$3e0]:=$00; delay(10);
  end;
  if paramcount=1 then i:=1;
  if (paramstr(1)='forward') or (paramstr(1)='FORWARD') then
    for x:=1 to i do
    begin
      port[$3e3]:=$8b; delay(10);
      port[$3e0]:=$80; delay(1000);
      port[$3e0]:=$8E; delay(100);
      port[$3e0]:=$80;
    end;
  if (paramstr(1)='backward') or (paramstr(1)='BACKWARD') then
    for x:=1 to i do
begin
  port[$3e3]:=$8b; delay(10);
  port[$3e0]:=$80; delay(1000);
  port[$3e0]:=$8D;
  delay(100);
end;