and circles. If these features are not available in the hardware, these figures must be drawn by a series of short line segments which approximate the curved figure.

Three-Dimensional Line Generation: Three-dimensional representation of vectors is available in some CRTs. Shading or intensity modulation are available, which makes 3-D pictures more realistic. Manufacturers offer the means to scale pictures. The size of the image is changed without changing its shape.

Peripheral Storage: Many graphics terminals have various peripheral devices for data storage or printing of data. Disks and magnetic tapes are available for an auxiliary storage device which can be used by the program in the graphics terminal. Many systems provide the user with a means of saving data on tape cassettes or on removable disk packs for future use. When the user returns to the terminal at a later time, he can resume his work by mounting the disk pack or tape which has his data. In many instances the user would like to have the information on the graphics terminal printed for future reference. This printed material is referred to as a hard-copy of the graphics terminal image and is produced on various types of printers that can be available at the terminal. Hard-copies can also be made from the graphics terminal display to such devices as microfilm and plotters.

Central Control Computer

The central control computer can vary from a mini-computer (generally 16 bit word length) to a very large computer, some with word length up to 60 bits. The computer may be running only graphics terminals or also other devices for other purposes. The input/output system, interrupt system, and instruction set are discussed.

INPUT/OUTPUT SYSTEM: Computers have generally 1-3 major I/O systems: direct I/O, direct memory access (DMA), and channels. Direct I/O is the simplest form of I/O, where one character at a time is input or output. Each character I/O must be done by the program. The DMA I/O system generally inputs and outputs words automatically from some pre-defined buffer location. The program must only define the buffer and start the operation. The DMA hardware performs the entire buffer transfer by cycle stealing from memory. During this time the program continues its own function without any apparent delay. The channel is similar to DMA, but more sophisticated. In some cases, it is capable of performing instructions known as channel instructions which can be executed at the same time the computer program is running. The channel can be viewed as a small computer itself dedicated to I/O. The channel can usually selectively drive more than one device at different times or simultaneously.

INTERRUPT SYSTEM: The interrupt system of the central control computer directly affects the response time that the student experiences at the graphics terminal. The time that is spent in servicing an interrupt includes the following: 1) latency -- the time it takes the CPU to recognize that an interrupt has occurred, 2) overhead -- the time it takes to find out what device caused the interrupt and to save the status of the computer, 3) the time it takes to execute the specific interrupt routine, and 4) the time it takes to restore the status of the machine.

The three common types of interrupt systems are polling, single level interrupts, and priority interrupts. In polling, the central computer checks each possible interrupt in turn to see if it needs servicing. Whenever an interrupt is active, it is serviced to completion, and then the polling continues. No priority system exists with polling. A single level interrupt system is one in which all interrupts trap to one pre-defined location. In the single level interrupt system, a greater amount of time must be spent in identifying the interrupting device than in polling. A priority interrupt system is one in which the interrupts trap to different locations. Each location has an assigned priority. A higher priority interrupt can interrupt the servicing of a lower priority interrupt.

Some interrupt systems allow the priorities to be dynamically changed under program control.

INSTRUCTION SET: Some of the main points to consider when looking at the instruction set of the central computer of a CAI/graphics system are the following: floating point instructions, block move instructions, and addressing modes, such as byte and bit addressing capabilities. Most computers offered today have adequate instruction sets to perform any of the functions required for controlling CAI/graphics system.

Communications

Communications in graphics systems involve two types of linkages: 1) the direct link by a wire of a graphics display terminal to a computer, and 2) the linking of the graphics display terminal to a central computer through a common carrier such as a telephone system.

The direct linkage of the computer to its graphics terminal involves using a wire or cable from the central computer's I/O system to the terminal. This type of communication is less prone to communication noise and fast transmission rates are possible at low cost. The distance that the terminal can reside from the central computer is limited. Generally, thirty to forty feet is the maximum. When a large number of terminals are involved, installations

will choose to have the central computer wired directly to a multiplexor which in turn will be directly wired to all terminals. This type of arrangement alleviates the need for a large number of wired interfaces coming from the computer to each terminal. Generally, only one wired interface is made between the central computer and the multiplexor. A variety of multiplexors are built. There are simple units that buffer one character at a time from one line to complex units with memory. Some of these complex multiplexors are programmable and act as stand-alone computers. See Figure 3.

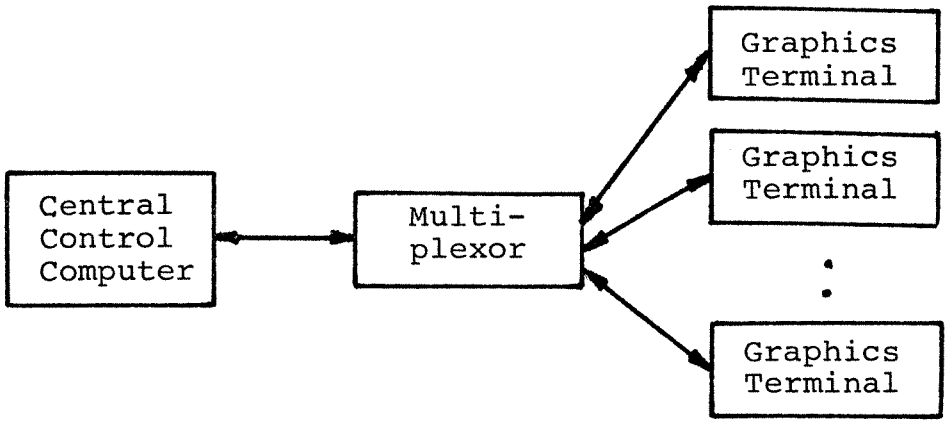

Figure 3

Many computer assisted instructional graphics systems require that the graphics terminals be remote from the central computer.

This type of requirement has promoted the need for communication via common carrier. Many facilities exist for long distance communication, but this paper will be limited to the discussion of communication via telephone lines (voice grade) within the area serviced by one telephone exchange.

When dealing with communications on telephone lines the need for some specific equipment arises. The output signals from the central computer must be converted to voice grade signals and then reconverted again to digital signals at the terminal. A unit called a modem is used to convert these signals. See Figure 4.

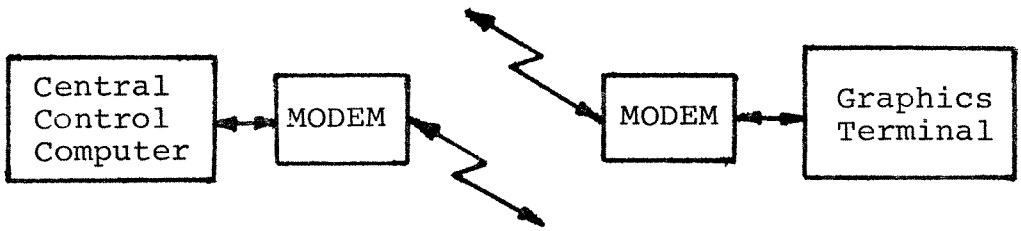

Figure 4

Many systems use an ordinary telephone line that is dialed from the terminal end. This means that the line goes into the normal switching system of the telephone

company. This type of line is limited in the transmission speed to about 300 bits per second because of the switching mechanism used by the telephone company. See Figure 5.

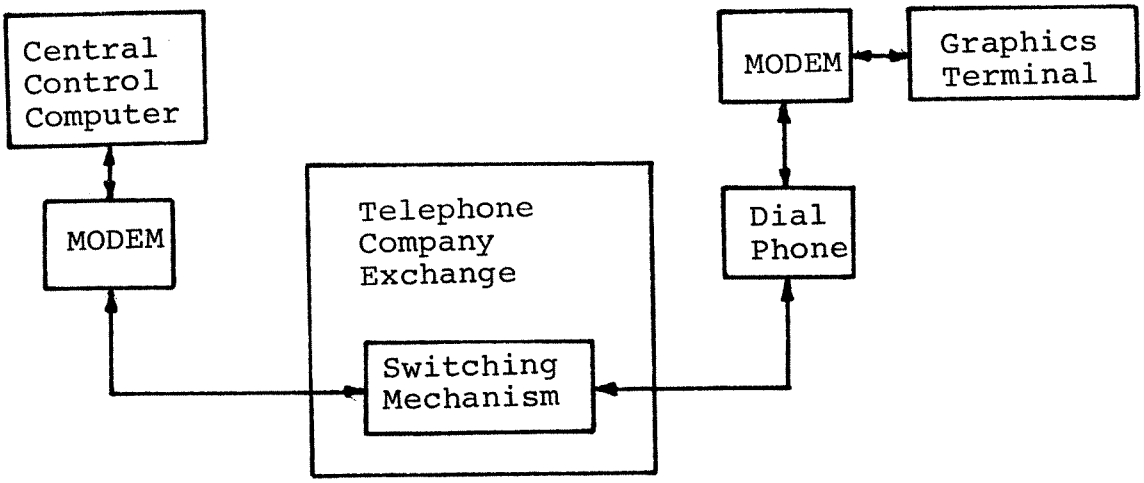

Figure 5

Many graphics systems require faster transmission rates, and therefore use private lines. A private line is an ordinary telephone line that is physically routed around the normal switching mechanisms of the telephone company exchange. This line need not be dialed up; it is always directly linked from the terminal to the central computer. The private line can have rates up to 1800 bits per second. See Figure 6.

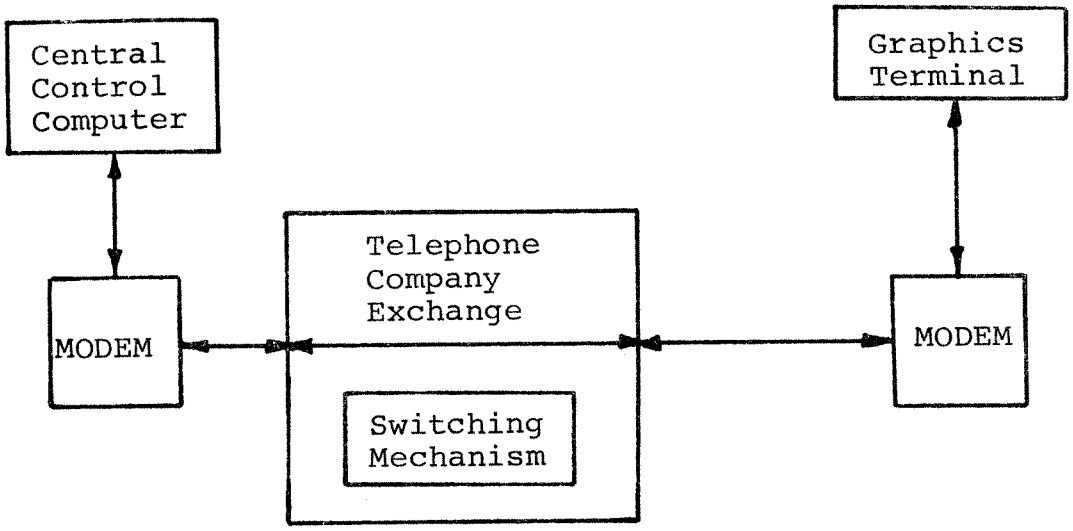

Figure 6

There are many types of communications on the previously described telephone lines. One-direction-only transmission is called simplex transmission. Two-directional transmission but only in one direction at a time is called half duplex. Two-directional communications simultaneously is called full duplex. In computer assisted instruction graphics systems, half duplex is usually adequate. The normal telephone line can carry simplex to full duplex transmission. The choice of the modem and the speed of the transmission generally governs the type of transmission that is possible.

As previously described, some graphics systems have multiplexors tied to the graphics terminals. With telephone communications, multiplexors are also used. See Figure 7.

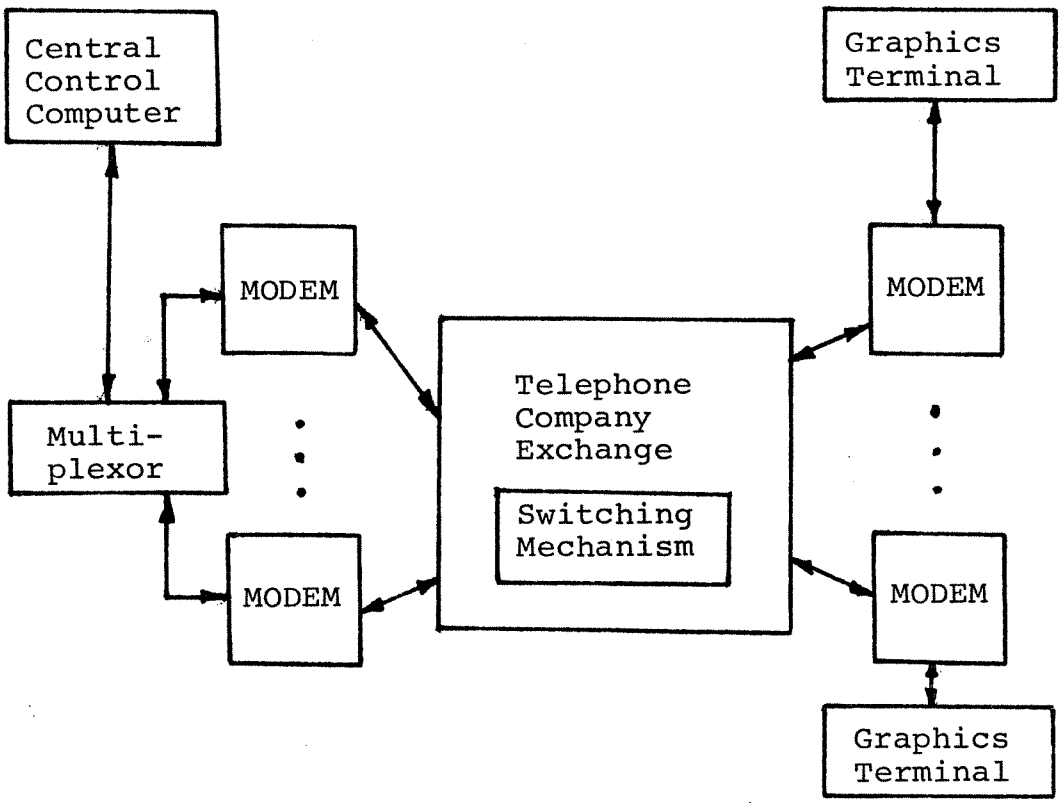

Figure 7

B. SOFTWARE

In a discussion of software for a CAI/graphics system, four distinct areas must be considered: the operating system, the graphics subsystem, the CAI/graphics application and the CAI/graphics author language.

Operating System

Two types of operating systems are generally used with a CAI/graphics system: a general purpose time-sharing operating system and a dedicated CAI/graphics operating system.

The general purpose time-sharing operating system can be used when programs other than those originating from graphics terminals need to be run on the central computer. The graphics terminals can be serviced in foreground time slices and other applications can run as batch jobs in the background. General purpose I/O and interrupt processing routines are features of this time-sharing system because of the large amount of I/O that is done. The advantage of this type of operating system is that it is very flexible. The disadvantage is the run-time overhead which is incurred with a general purpose operating system.

If the central computer is only used for CAI/graphics applications, and if the computer is driving only a few

terminals, say two, a different operating system is generally used. This operating system is called a dedicated system and is designed solely for the CAI/graphics application. If one is dealing with only one or two terminals, generalizing the operating system is too costly in computer run-time and in programming effort. The dedicated operating system generally becomes interrelated with the application and is truly a CAI/graphics system. The disadvantage of designing and implementing a dedicated CAI/graphics system is that expansion of the system generally would involve a redesign, whereas in the general purpose system, expansion can be effected more easily.

Graphics Subsystem

The graphics subsystem generally consists of a set of graphics subroutines which control the graphics terminal. Graphics subroutines, such as scaling, zooming, selective erasing, etc. are a part of the graphics subsystem. A general purpose PLOT package is also included in the subsystem. The uniqueness of the interfaces of the graphics terminals requires that special I/O packages be added to the graphics subsystem in order to support each type of terminal in the system. These I/O routines would perform functions such as data formatting, where the data is put into a form acceptable to the graphics terminal.

The graphics subsystem can be a separate entity of the entire system with certain pre-defined protocol for each routine, and then all course authors, system programmers, and even students can easily use the graphics subroutines without having to understand all the complexities of the graphics terminals.

Sometimes the graphics subsystem is not a separate entity. The I/O routines are embedded in the operating system, and the remainder of the graphics subsystem is a part of the courseware programs. Each routine is written by course authors or systems programmers as the need arises. For example, if a course author needs a scaling routine, he writes his own special purpose scaling program or borrows one from another author, and it becomes part of his courseware.

Graphics Application (Courseware)

The graphics application consists of the programs that communicate with the graphics terminal, which has recently been referred to in the literature as courseware. The courseware is called up by the terminal user and run in a time slice. The courseware program is usually programmed in an author language or in some suitable high level language which facilitates the programming of interaction, answer processing, and graphics I/O. Sometimes

the assembly language of the central computer is used to generate the courseware, but this approach is generally taken only for a small dedicated graphics system.

Some of the traditional instructional strategies for courseware include: drill and practice, tutorial, inquiry, simulation, and gaming.

Drill and practice CAI programs generally supplement some other type of instruction. New material is not presented in drill and practice programs, but they merely review the concepts presented elsewhere. Drill and practice programs are especially helpful if they are programmed to "learn" the student's weaknesses and then place emphasis on the types of problems with which he has difficulty.

Tutorial CAI programs are intended to be the only mode of instruction that a student receives on a subject. Tutorial programs present factual statements about a subject and then ask questions about the material just presented. Depending on the student's response, remedial branches can be taken. Sometimes the student is asked readiness questions before new material is presented, and if the student responds correctly, he can bypass sections of the lesson. The standard criticism of tutorial mode is that the student is too passive in his participation.

Inquiry CAI programs let the students become more active in the learning process. The student is presented

with a problem or situation, and then the student, using an inquiry vocabulary, can ask questions about the problem until he arrives at a solution. A large data base is maintained by the inquiry program, and the student's questions are answered by accessing the data base.

Simulation and gaming CAI programs are two additional instructional modes, which seem to have the most use for graphics. Business games have been developed, where graphs of a company's status are shown, such as profit, size of company, etc. Laboratory simulation programs show the simulated experiment and the changes that occur in it based on the student's interaction.

The trend in courseware now seems to be a combination of several different modes in one course, where the student has the control of sequencing through a lesson.

The philosophy of instructional strategy adopted by the TICCIT CAI/graphics project is that there are only about 20 different instructional strategies which exist independent of what subject is being taught (McWilliams, 1972). The courseware for their project is being developed by first debugging the instructional patterns. Then the subject matter is entered on printed forms, and finally after a preprocessor accepts as input both the instructional patterns and the subject matter, it produces as output the

courseware. This approach is taken in an effort to save time in the development of courseware.

CAI/graphics Author Languages

Many different author languages are used to develop courseware. The choice of author language depends on many factors which include the following: the level of sophistication of the author, the availability of the language, the versatility of the student terminal, the student record-keeping capability, and the answer processing features.

The author language can be a compiler-type of language, where the courseware is compiled either in background mode as a batch job, or on an entirely different computer system. The object code of the courseware then runs in the foreground of the central control computer. The language can be interpretive, and then the source code of the courseware is interpreted in the foreground as the terminal user executes the lesson.

The CAI/graphics languages that are used have generally been developed by taking an existing CAI language and attaching to it graphics subroutines which support the unique type of graphics terminal to be used. For example, Coursewriter II supports only the IBM 1510 display terminal; TUTOR IV supports the plasma display terminal. Some

interactive graphics languages have been used to develop CAI material, but authors who make this choice sacrifice many CAI features, such as answer matching, criterion branching, and restart capability. The graphics operations possible in a strictly interactive graphics language are often excessive when applied to CAI.

Some existing CAI/graphics languages that are described here include: Coursewriter II, TUTOR IV, and PLANIT. Each of these languages has reasonable features to implement CAI. This portion of the paper discusses only some of the graphics capabilities in the languages.

Coursewriter II was designed and implemented by IBM to be used on the 1500 Instructional System (IBM, 1968). Since this language is one of the oldest CAI/ graphics langauges, the graphics features in it now seem crude and inefficient, although the graphics capabilities are still being used as described here. Coursewriter II has a picture file, which contains all the pictures to be drawn during a lesson. Dynamically creating a new picture based on student response during the lesson is impossible. One course can have 192 pictures, where each picture is defined by a dot matrix which is 36 dots high and 16 dots wide. For example a picture of a bottle could be laid out as in Figure 8. The dot matrix must then be

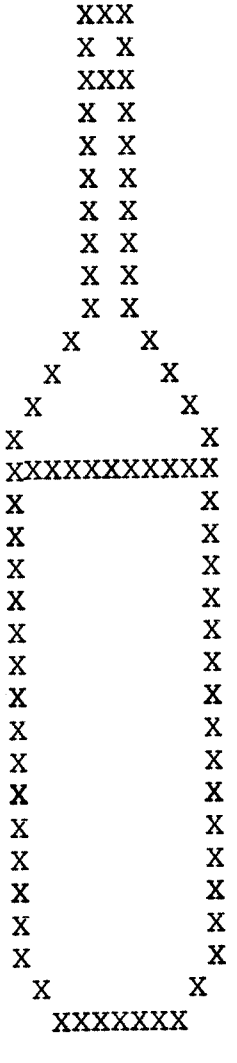

Figure 8

keypunched and stored on the file, where a unique number is assigned to each picture. Then, when the author wants to display a picture, he writes the Coursewriter command: dg rr,cc/xxx, where dg is a mnemonic which indicates display graphic, rr,cc defines the screen location by row and column, and xxx is the unique number of the graphic. The picture cannot be scaled.

Coursewriter does not allow graphic input. A light pen can only be used to point to a picture on the screen. The coordinates of the point at which the student is expected to point are stored by the author as the correct answer, and they are compared against the coordinates of the point at which the student pointed. Coursewriter II was certainly a worthwhile introduction into the CAI/graphics world.

TUTOR is a CAI/graphics language which was developed at the University of Illinois (Avner, 1970 and TUTOR User's Memo, 1973). It is designed to interface with the plasma display terminal. The language has been growing, with respect to its capabilities, since it was

originally developed. As the sophistication and demands of the user increase, and as the student terminal becomes more complex, the language is modified to meet the new requirements.

The most recent version of TUTOR has more graphics capabilities than its predecessors. Some of the graphics commands are as follows: line xy,xy, which connects the two absolute points given, figure x_1y_1,x_2y_2,... x_8y_8, which connects all the points given, circle tag, which draws a circle with the relevant parameters given in the tag field, and dot xy, which plots a single dot on the screen at xy. Besides these basic drawing commands, TUTOR has a PLOT package, which contains commands, such as origin, axes, bounds, scalex, scaley, labelx, labely, etc., whose meanings are obvious. Another graphics command, which is a remnant of an older version of TUTOR, is Plot name, which draws the dot matrix which is called name. The dot matrix is defined by the name Char list command, where list contains the xy coordinates at which a dot is to appear.

Graphics input by the students is possible in the TUTOR language as described in Chapter II of this paper in the University of Illinois Geometry Program.

A picture file capability, similar to that of Coursewriter, is available, however, in TUTOR, the author defines his pictures by drawing them at the graphics terminal. Then the picture definitions are saved on a file and displayed later (Pratt, 1974).

The author language, PLANIT, is one of the more recently developed CAI languages (Bannick, 1970). It was developed by Systems Development Corporation with the goal of creating a machine-independent CAI system. PLANIT has been implemented on several different machines. The student terminal normally used with PLANIT is the teletype, so the graphics are only tabular pictures. The pictures are drawn the same way as in Coursewriter. However, in Coursewriter, a picture which is 36 dots high by 16 dots wide occupies only a small portion of the CRT screen and the dots appear to be smooth lines. On the teletype in PLANIT, the dot matrix takes up a much larger section of the paper and the dots are quite distinct.

Development work in a more elaborate graphics capability is being done, where the student terminal consists of an electronic tablet and stylus (Williams, 1968). A novel feature of this developing system is complete graphics input. The student can answer the questions by writing (drawing) the solutions on the electronic tablet

using the stylus. The set of symbols that are recognized includes the alphabet, numbers, and several special characters, such as standard mathematical notation. The system maintains a unique dictionary of characters for each user, where this dictionary is hand-printed input initially via the tablet. So the system actually "knows" how each user writes. The benefit of this system is that information, such as mathematical expressions, which can include subscripts, superscripts, summation signs, integrals, etc., can be input just as you would normally write them down. The intermediate step of converting an expression to a linear form, as is acceptable in FORTRAN, is omitted. For example, $Y=SIN^2X$ can be written as input rather than Y=SIN(X)**2 as in FORTRAN. This system does not allow any other form of graphics input than that described above; pictures are not valid input. The only type of graphics output is simply an acknowledging echo print of what the student inputs.

This chapter has discussed the hardware and software as separate entities. In the next chapter, guidelines for putting together a CAI system are discussed.

CHAPTER IV

THE CAI/GRAPHICS SYSTEM

When considering the purchase or development of a CAI/graphics system, the educational institution must evaluate all aspects of the system before making a decision as to what type of system is really needed. A number of general questions must be asked and answered before considering the actual type of hardware and software to use in the system. The following questions should be considered:

-What type of student will be taught?

-What is the maximum number of students to be on-line at once?

-What kind of courses will be offered?

-What instructional strategies will be used?

A few general considerations in answering these questions are presented in the first section of this chapter.

What type of student? The age range and abilities or disabilities of the students who are to use the system must be catered to when selecting the hardware or when designing the software. System response time is more

critical among young students, whose attention span is shorter, than it is with older students.

How many students? The number of students to be taught by the system does not imply that they all will be on-line at the same time. Scheduling students on the graphics terminal would be necessary. The maximum number of student users the system can support should be a "known" factor to everyone connected with the system.

What courses? A lot of courseware has been written by many educational institutions. An index of courseware has been published. (LeKan, 1971) The courseware programs in this index are listed by subject. A description of the program, the required readiness of the student, educational objectives, the minimum hardware configuration, and other courseware parameters are also given. This index at first has the appearance of a courseware catalog, where the aspiring CAI teacher simply orders the desired course. However, the courseware, in general is not transferable from one institution to another because of hardware and/or software differences and sometimes courseware authors will not share their work. The amount of effort required to run courseware obtained from a different institution is oftentimes greater than if the courseware were rewritten for the specific system. This transferability problem is significant when considering

non-graphics CAI via the standard interface of a teletype. However, when CAI/graphics is considered, where the type of interface to the graphics terminal is not standard from vendor to vendor, the problem of tranferability of CAI/graphics courseware is almost insurmountable unless two identical systems are involved. Until some sort of standards or guidelines are set up for courseware development, each ambitious CAI/graphics teacher has to write his own style of courseware.

What instructional strategies? Many course authors require extensive student records, so that the lesson can be dynamically tailored for the individual student. Some authors will keep a history of all student responses made during the current course. A complex file system may be needed to support the student files. A file system means more software to develop, and a larger CPU memory and a random access auxiliary storage device to purchase.

After the above questions are explicitly answered, the educational institution is ready to choose specific hardware and software to accomplish the desired goals. The next section of this chapter presents some guidelines in selecting the hardware and software.

A. HARDWARE

What type of graphics terminal? After the courses and instructional strategies are known, the type of graphics terminal needed should first be considered. For example, is a light pen needed, what type of resolution is needed, does the system need vector line generation, does the course require scaling and rotating of pictures? Graphics terminals vary greatly in cost and performance. A programmable graphics terminal is more flexible, but it is also more costly. However, in some instances, a graphics terminal that would have been replaced if it were not programmable, is simply reprogrammed to meet the new needs, which is a cost savings in some respects. The type of courses to be offered dictates whether a simple terminal is needed or a more complex terminal, possibly with its own main computer.

Another consideration on graphics terminal selection is how much information must be transmitted over the communication link to draw one picture, say a circle. Some terminals simply require the coordinates of the center and the radius to be transmitted, whereas other terminals require the coordinates of several short line segments to approximate the circle. Generally, a faster communications link, which is more costly, can compensate for a cheaper "dumb" terminal. If a smart expensive terminal is used,

the communications link does not have to be so fast and expensive because less information is transmitted per picture. A trade-off situation does exist here.

What kind of communication? The type of graphics terminal chosen and the physical lay-out of the system determine the type of communications. Some terminals must be very close to the main computer, while others have phone adapters and can be miles away. If phone communications are chosen, the cost of phone lines over long distances is not trivial.

What main computer? The central control computer should have a good interrupt system, flexible instruction set and reliable I/O system.

The interrupt hardware of the main computer is critical in time-sharing. An adequate priority system of interrupts is recommended, where the software can assign priorities to the interrupts and process them without too much overhead. External interrupts should be assigned specific locations in memory to which they interrupt since interrupt processing time is faster if this capability is available. If this process of interruption and servicing and returning to the interrupted program is fast enough to keep the graphics terminals and any other device running without any noticeable delay, then the interrupt system would prove to be adequate.

The instruction set of the main computer should have floating point instructions because dynamic calculations are often performed in graphics. Ideally one should look for a machine with character or byte manipulation instructions. In many instances, the central computer program (courseware) will be modifying a picture which may require moving byte-length pointers from one location in a buffer to another, or characters may need to be packed or unpacked into a buffer. Block data move instructions are excellent for I/O. Multiple register loads and stores are also good instructions to have for an interrupt routine's entry and exit where the status of the machine must be saved and restored. Automatic stack control for nesting of subroutines is another good instruction set feature to have.

The input/output system for a CAI/graphics system should be studied very closely. It is in the I/O system that most CAI/graphics systems fail to meet their expectations. Most graphics systems should consider direct memory access (DMA) or channel for output if a large number of terminals are to be successfully controlled and refreshed from a central computer. Graphics input generally needs to be serviced character by character, so for input, direct I/O is often desirable. The type of I/O system required for the job is very application dependent.

When analyzing the input/output system for speed to handle communications and processing, some basic parameters should be calculated (Tirrell, 1971). During one second of machine time, "X" number of machine cycles are available for either communications I/O or processing. The maximum number of machine cycles in one second required to handle communications can be calculated, which would yield some information as to how much time is left for processing per second. For example, if the speed of the communications line is 2400 bps and if the data character size and buffer width is 8 bits, then 300 transfers per second is the maximum necessary to drive one line, where each transfer takes one machine cycle. If 100 similar lines exist, then 30,000 machine cycles per second (maximum) are busy just with communications. An average computer with a machine cycle time of 2 microseconds has 500,000 machine cycles per second available. So in this case 470,000 cycles are left for processing, which includes execution of all courseware as well as servicing any interrupts. A mix of necessary "processing" instructions per second could also be determined. Then if the sum of the two requirements for machine cycles per second exceeds the number of available machine cycles per second, queuing of data must be considered or possibly an overloaded

system would result. Even with queuing, a saturation point exists, where the system never catches up.

B. SOFTWARE

What operating system? A general purpose time-sharing operating system should be designed for a CAI/graphics system where the central computer can control more than a few terminals. Although the type of general purpose operating system that is chosen is dependent on the application requirements, it should operate in its own right, distinct from the application which is running, which in this case is a CAI/graphics application. A primary factor to be considered, when choosing an operating system design, is what the central computer is going to be used for. Is the computer dedicated to CAI/graphics jobs or does it run other unrelated jobs as well? If the CAI/graphics system is involved with many terminals, and if a general purpose time-sharing operating system is used for the central computer, the central computer can be used to run applications unrelated to CAI/graphics without any appreciable change to the operating system.

What author language? The author language must be flexible enough to handle all instructional strategies for a wide range of subjects without consuming a lot of the author's time. It is apparent from the languages

described in Chapter III that an adequate CAI/graphics language has not yet been written. According to Suppes (1970) the most satisfactory author language for non-graphics CAI has not even been developed. The additional requirement for an author language to support graphics input and output processing further emphasizes the lack of a satisfactory CAI/graphics author language, since the study of graphics input/output is also still in the developmental stages.

In this section of the paper, some of the necessary requirements in a CAI/graphics language are described and justified.

One of the basic features a CAI/graphics language should have is the ability to retrieve and add to a file of "picture" definitions, where these pictures can be displayed using different scale factors and at different locations on the screen. For example, if a flowcharting lesson were being presented, the decision box, when first introduced, could be the central figure on the screen with an explanation about it at the bottom of the screen. Later in the lesson, when the decision box is used in conjunction with other flowcharting symbols, it could be drawn smaller at the desired screen location. The decision box picture would simply be part of the picture

definition file associated with the flowcharting lesson. The author language to display the decision box picture could then be written as follows:

```
DISPLAY DECBOX AT LOC 10,10 IN SIZE 4 X 4,
```

where DECBOX is the name in the file directory associated with the picture definition of a decision box, and LOC 10,10 IN SIZE 4 X 4 indicates that the decision box should be completely contained in a box which is 4 units long by 4 units wide, and that the lower left hand corner of the box is at absolute screen location 10,10. An alternative for the author without the picture file capability is to draw the decision box using absolute coordinates each time it is to be drawn on the screen. The author would have to write something similar to this:

```
DRAW LINE FROM 12,10 TO 14,12
DRAW LINE FROM 14,12 TO 12,14
DRAW LINE FROM 12,14 TO 10,12
DRAW LINE FROM 10,12 TO 12,10.
```

The next basic requirement in a CAI/graphics language is for the author to be able to build the picture file easily. The act of picture definition by the author could be a graphics interactive process itself. The author

could draw his desired file of pictures by using the light pen at the CRT, where the movement of the light pen would be followed by a tracking program, or by using an alternative device, such as a mouse or stylus (see Chapter III). In either case, the author could draw a picture, and then enough information to regenerate the picture would be saved on the file. The picture could be given a name, which could later be used when displaying it. After the author drew each picture, he could see it on the screen and change it if he were not pleased with it by interacting with a graphics editing program. If the author could not draw his picture file, he would have to generate it by writing a program, where each picture is described in terms of absolute vertices and lines, an extremely time-consuming task and quite prone to errors.

Some additional graphical output features of a CAI/graphics language that would be desirable are selective erasing of the screen, as well as blanking the screen completely, blinking certain pictures on the screen for emphasis, and a general PLOT package to dynamically plot various functions the author or student may want to see.

Graphical input, where the student draws a picture or perhaps points at something on the screen using a light pen or similar device, could be handled by the

same program (in student mode) which the author uses to define his picture file.

A more elaborate but not really necessary feature of a CAI/graphics language is the ability to do animation. This capability could be quite useful in a simulated laboratory experiment, although the use of it could easily be distorted to appear theatrical and really not necessary for what is being taught.

This section has attempted to give some guidelines to follow in constructing a CAI/graphics system. One of the best things that an educational institution can do in order to avoid disappointment and frustration, is to simulate or model the system prior to developing it and prior to final hardware selection. A simulator should be built with input parameters, such as the speed of the lines, the number of terminals, cycle time of the central computer, etc., and then the system could be optimized and understood before it is built.

V. CONCLUSIONS

CAI/graphics systems are still in their infancy. CAI systems and interactive graphics systems have existed independently for about 15 years. However, neither type of system has really been perfected, so when the two imperfect and undeveloped systems were merged into a CAI/graphics system, the resulting concept obviously needs a lot of research and development before a reasonable and inexpensive system can result.

This paper has explored some of the dimensions of CAI/graphics systems. Several existing CAI/graphics programs have been described in order to understand how far the CAI/graphics concept has been developed at different schools. The components in a CAI/graphics system have been discussed in detail so that some insight can be gained into how they all fit together. Finally, some guidelines have been given for developing a CAI/graphics system.

The future of CAI/graphics systems is clouded by the expense of the systems and by the traditional disagreements which educators have about CAI in general. However, as time passes and standards change, the author feels that

CAI/graphics systems will become more economical and will be readily available in most schools.

BIBLIOGRAPHY

Abboud, V. and V. Bunderson (1971) A Computer-Assisted Instruction Program in the Arabic Writing System. The University of Texas at Austin, Computer-Assisted Instruction Laboratory, Technical Report No. 4.

Atkinson, R. and H. Wilson (eds.) (1969) Computer-Assisted Instruction: a Book of Readings, Academic Press, New York.

Avner, R. and P. Tenczar (1970) The TUTOR Manual. Illinois University, Urbana, Illinois, Computer-Based Education Lab, R-CERL-X-4.

Bennick, F. and C. Frye (1970) PLANIT Author's Guide. System Development Corporation, Santa Monica, Calif.

Boehm, B., V. Lamb, R. Mobley, and J. Rieber (1969) POGO: Programmer-Oriented Graphics Operation. The RAND Corporation, Santa Monica, Calif., RM-5825-PR.

Bork, A. and R. Ballard (1972) Computer Graphics and Physics Teaching. California University at Irvine.

_____ (1971a) Inexpensive Timeshared Graphics on the SIGMA 7. California University at Irvine.

_____ (1971b) Teaching Conversations with the XDS SIGMA 7. California University at Irvine.

Bunderson, V. (1970) Justifying CAI in Mainline Instruction. Proc. of a Conference on Computers in the Undergraduate Curricula, University of Iowa, Iowa City, Iowa.

Canter, J. (1971) Blueprint for an Anatomical Surgical Computer. A Quarterly Report of SIGGRAPH-ACM Computer Graphics, Vol. 5, No. 2.

Chase, E. and D. Brick (1970) Interactive CRT Display Terminals. Modern Data, Vol. 3, No. 6.

Cotton, I. and F. Greatorex, Jr. (1968) Data Structures and Techniques for Remote Computer Graphics. Proc. AFIPS 1968 FJCC, Vol. 33, Part 1.

Dennis, J. (1969) Teaching Selected Geometry Topics Via a Computer System. Illinois University, Urbana, Ill., Computer-Based Education Lab, CERL X-3a.

Dwyer, T. (1972) Teacher/Student Authored CAI--Using the NEW-BASIC System. Comm. ACM, Vol. 15, No. 1.

Frye, C. (1968) CAI Languages: Capabilities and Applications. Datamation, Vol. 14, No. 9.

Gallenson, L. (1967) A Graphic Tablet Display Console for Use under Time-sharing. Proc. AFIPS 1967 FJCC, Vol. 31.

Hagan, T. and R. Stotz (1972) The Future of Computer Graphics. Proc. AFIPS 1972 SJCC, Vol. 40.

Heller, J. and others (1971) Graphics Representation of Musical Concpets: A Computer Assisted Instructional System. Connecticut University, Storrs, Conn.

Hicks, B. and S. Hunka (1972) The Teacher and the Computer, W. B. Saunders Company, Philadelphia.

Hurwitz, A., J. Citron and J. Yeaton (1967) GRAF: Graphic Additions to FORTRAN. Proc. AFIPS 1967 SJCC, Vol. 30.

Hyatt, G., D. Eades, and P. Tenczar (1972) Computer-Based Education in Biology. BioScience, Vol. 22, No. 7.

IBM (1971) Data Communications Primer--Student Text.

_____ (1968) IBM 1500 Coursewriter II--Author's Guide.

Landa, S. (1972) CATTS: Computer-Aided Training in Troubleshooting. The RAND Corporation, Santa Monica, Calif., R-518-PR.

Lekan, H. (ed.) (1971) Index to Computer Assisted Instruction, Harcourt Brace Jovanovich, Inc., New York.

Levien, R. (1972) The Emerging Technology, McGraw-Hill Book Company, New York.

Machover, C. (1972) Computer Graphics Terminals--a Backward Look. Proc. AFIPS 1972 SJCC, Vol. 40.

McWilliams, E. (1972) Large Scale CAI--The NSF Project. Proc. ACM 1972 Annual Conf., ACM, New York.

Newman, W. and R. Sproull (1973) Principles of Interactive Computer Graphics, McGraw Hill Book Company, New York.

Oliver, A. (1969) A Measurement of the Effectiveness of an Interactive Display System in Teaching Numerical Analysis, Ph.D. Dissertaion, University of North Carolina at Chapel Hill.

Papert, S. (1971a) A Computer Laboratory for Elementary Schools. Massachusetts Institute of Technology Artificial Intelligence Memo No. 246.

_____ (1971b) Teaching Children Thinking. Massachusetts Institute of Technology Artificial Intelligence Memo No. 247.

Papert, S. and C. Solomon (1971c) Twenty Things to Do with a Computer. Massachusetts Institute of Technology Artificial Intelligence Memo No. 248.

Pratt, T. (1974) Conversation with C. Kostetsky, Austin, Texas.

Rapkin, M. and O. Abu-Gheida (1968) Stand-alone/Remote Graphics System. Proc. AFIPS 1968 FJCC, Vol. 33, Part 1.

Sleeman, D. and W. Phelps (Undated) A System for Graphical Communication in a CAI Context. The University of Leeds, United Kingdom.

Stifle, J. (1971a) The PLATO IV Architecture. Illinois University, Urbana, Illinois, Computer-Based Education Lab, CERL X-20.

_____ (1971b) A Plasma Display Terminal. Illinois University, Urbana, Illinois, Computer-Based Education Lab, CERL X-15.

Suppes, P. and M. Morningstar (1969) Evaluation of Three Computer-Assisted Instruction Programs. Stanford University, Technical Report No. 142.

Suppes, P. (1970) Systems Analysis of Computer-Assisted Instruction. The Challenge to Systems Analysis, John Wiley and Sons, Inc., New York.

Tirrell, J. (1971) Analysis of the Throughput of a Communications Processor--Part I. Computer Design, September, 1971.

TUTOR User's Memo (1973) Introduction to TUTOR, PLATO IV Version, University of Illinois, Urbana, Illinois.

Williams, T. and C. Frye (1968) An Instructional Application of Computer Graphics. Educational Technology, Vol. 8, No. 11.

Wilson, H. (Undated) An Initial Evaluation of Two Experimental CAI Systems, Stanford University.

Yule, A. (1972) Human Response Times in a Graphic Environment. Computer Bulletin, Vol. 16, No. 6.