[CP-02-032] On the Origins of Computing and GIST: Part 2, A Perspective on the Role of Peripheral Devices

Abstract

GIS implementations in the late-1960s to mid-1980s required the use of exotic peripheral devices to encode and display geospatial information. Data encoding was normally performed in one of two modes: automated raster scanning and manual (vector) coordinate recording. Raster scanning systems in this era were extremely expensive, operated in batch mode, and were located at a limited number of centralized facilities, such as federal mapping agencies. Coordinate digitizers were more widely distributed and were often configured with dedicated minicomputers to handle editing and formatting tasks. Data display devices produced hardcopy and softcopy output. Two commonly encountered hardcopy devices were line printers and pen plotters. Softcopy display consisted of cathode ray tube devices that operated using frame buffer and storage tube technologies. Each device was driven by specialized software provided by device manufacturers, leading to widespread hardware-software incompatibly. This problem led to the emergence of device independence to promote increased levels of interoperability among disparate input and output devices.

Keywords: computing history, GIS history

Author & citation

Armstrong, M. P. (2019). On the Origins of Computing and GIST: Part 2, A Perspective on the Role of Peripheral Devices. The Geographic Information Science & Technology Body of Knowledge (3rd Quarter 2019 Edition), John P. Wilson (ed.). DOI: 10.22224/gistbok/2019.3.7

Explanation

1. Introduction
2. Data Encoding (Digitizing) Peripherals
3. Memory & Storage Peripherals
4. Display Hardcopy
5. CRT Graphics Displays
6. Storage Tube Displays
7. Input-Output Device Independence
8. Summary

1. Introduction

Early GIS implementations used specialized and expensive equipment to encode digital information, as well as to produce both hardcopy and softcopy maps. The high cost of these peripheral devices meant that they were normally isolated in centralized locations. Tomlinson (1998), for example, reported that early Canada Geographic Information System

(CGIS) work was carried out “without computer screens” and that “as late as 1967, computer graphics screens cost approximately $90,000”. Adjusting for inflation this would amount to approximately $434,000 in 1998 dollars or $682,000 in 2019 dollars (see Additional Resources). With the progression of time, costs dropped and peripherals were dispersed among a larger set of locations, eventually arriving in small labs and offices. The purpose of this paper is to describe these peripheral devices, used from the late-1960s to the mid-1980s, and to show how they affected the development of GIST software and its use.

2. Data Encoding (Digitizing) Peripherals

Though it is now commonly observed that geospatial information is “born digital”, in the early era of GIS, paper maps were primary data sources that needed to be converted into digital representations, a process that required considerable effort and the intervention of complex software and hardware systems (see Dueker, 1975). Several “families” of devices were used to perform this analog to digital transformation, the main distinction being between raster scanning and coordinate-based digitizers. Much effort was also placed in developing approaches to convert between these two representations (e.g. Peuquet, 1981a; 1981b).

2.1 Raster Scanners

Peuquet and Boyle (1984) provide a comprehensive overview of raster scanning and processing, summarizing data organization strategies as well as hardware alternatives available at that time. Scanners operated in either drum or flatbed modes and used several means (e.g., photodiodes and lasers) to create binary arrays that denoted the presence/absence of line work. The operation of automated line following technology is also described, as is a coverage of raster-vector conversion issues such as skeletonization. At that time, raster scanners were “high end” peripherals and as a consequence their use was typically restricted to map production systems in federal laboratories and agencies. Specific instances of operational scanning systems are also described, such as the one developed for the CGIS (Canadian Geographic Information System) and a production system at the US Defense Mapping Agency.

Two additional examples are illustrative:

1. ORRMIS (Oak Ridge Regional Modeling information System) developed an elaborate process for scanning maps (Meyers, Wilson, & Durfee, 1976). Maps were photographically separated into black and white (binary) masks for map categories (e.g., a single land cover type). These were then photographed onto 35mm film and processed as negatives which were sent to the University of Pennsylvania where they were scanned. The results were then stored on magnetic tape and shipped back to Oak Ridge for processing on a mainframe system (IBM/360-91).
2. Lee (1985) describes the Scitex systems, first acquired in 1978, that were used by the USGS to produce 1:100,000 digital line graphs that served as a geographic basis for data collection activities conducted by the US Census Bureau. System costs for a Scitex scanner, edit station and laser plotter ran to $700,000 (approximately $2.75 million in 2019 dollars).

In general, these systems were operated in batch mode and often required manual pre-processing to convert analog maps into a format that could be scanned, as well as considerable post-processing (often hours of CPU time) to obtain vector-format line work that could be used in GIS applications. There were also no standard formats for encoding scanned images.

2.2 Coordinate Digitizers

Digitizing in this mode usually began by taping a map to a large drafting table-like digitizer surface and then establishing a relationship between map coordinates (e.g., Universal Transverse Mercator) and digitizer coordinates. Early digitizers operated mechanically or opto-mechanically with a sliding arm that yielded Y values and a cursor that slid horizontally along the arm to yield X values. Later, two main types of electronic coordinate digitizers were encountered. The first used a solid state approach to determining position. A fine orthogonal mesh of wires was covered by plastic or Formica and electromagnetic induction was used to determine the location of a cursor (often to the nearest 0.01 inch). Cursors were usually either a puck with function buttons and a cross-hair sight, or a pen like device that was traced over line work. For a detailed description of one such digitizing system, see Jenne et al. (1972). The second type of digitizer had linear microphones along the X and Y dimensions of the digitizer surface and the cursor emitted a spark-like sound when a point was encoded. The time difference between when the sound was emitted and when it impinged at each microphone was recorded and a time-distance transformation was made to yield locations; a thermistor was used to adjust for changes in the speed of sound at different temperatures (Figure 1).

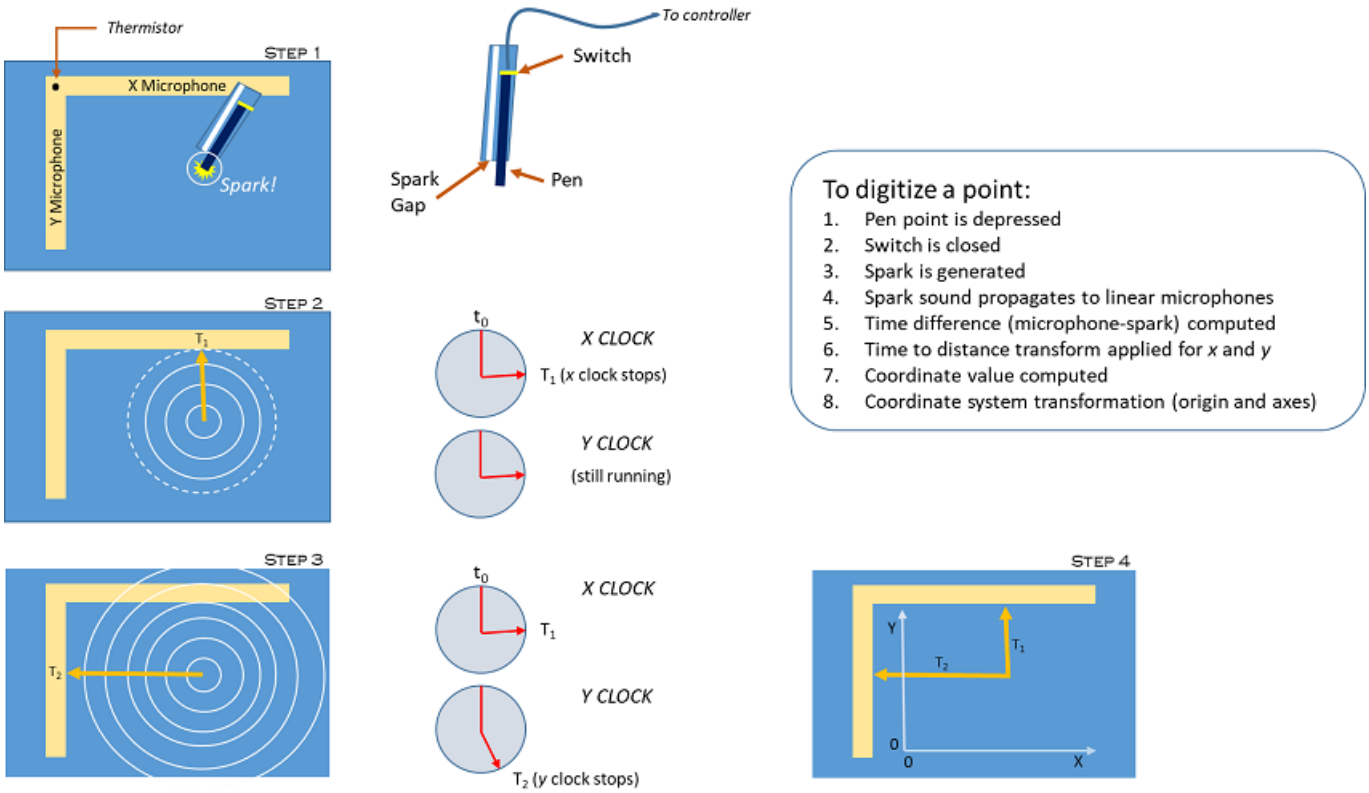


Figure 1. Schematic design of a sonic digitizer (after Eyton & Roseman, 1979).

Coordinates were normally recorded using one of three methods: 1) operator selected

points, 2) time sampling and 3) distance threshold based sampling. In operator selection mode, an attempt is made to encode points that comprise an accurate representation of the input line (see Jenks, 1981). Time mode samples points at a fixed interval (e.g. 0.5 second), irrespective of line complexity, whereas the threshold approach samples a point only when some distance from the previous point is exceeded (e.g. 0.1 cm). For large digitizing projects involving multiple individuals, detailed behavioral protocols were sometimes specified in an attempt to ensure some degree of uniformity in the way that measurements were made (e.g. ISIS, 1982).

2.3 Connecting Digitizers to Computer Systems

Douglas (1982) describes a configuration of GIST peripherals that were connected using a dial-up telephone connection to a centralized computer system. Dueker and Noynaert (1977) describe a different hardware and software complex (Figure 2) that relies on the use of a local minicomputer to handle interactive digitizing and a connection to large-scale centralized systems for additional processing, storage and plotted output. Please note that the IMLAC minicomputer had only 16 kilobytes of memory. Figure 3 shows the control console for the system, replete with binary control switches and red lights to indicate states. And yes, that is a wooden box. The main processing was done on the HP time-sharing system that specialized in running programs written in BASIC; the Iowa digitizing software consisted of approximately 20,000 lines of BASIC (Dueker & Noynaert, 1977). The results of the editing and formatting software also served as inputs to batch geospatial software (e.g. SYMAP) running on a large scale IBM System 360/65 that had less than two megabytes of memory.

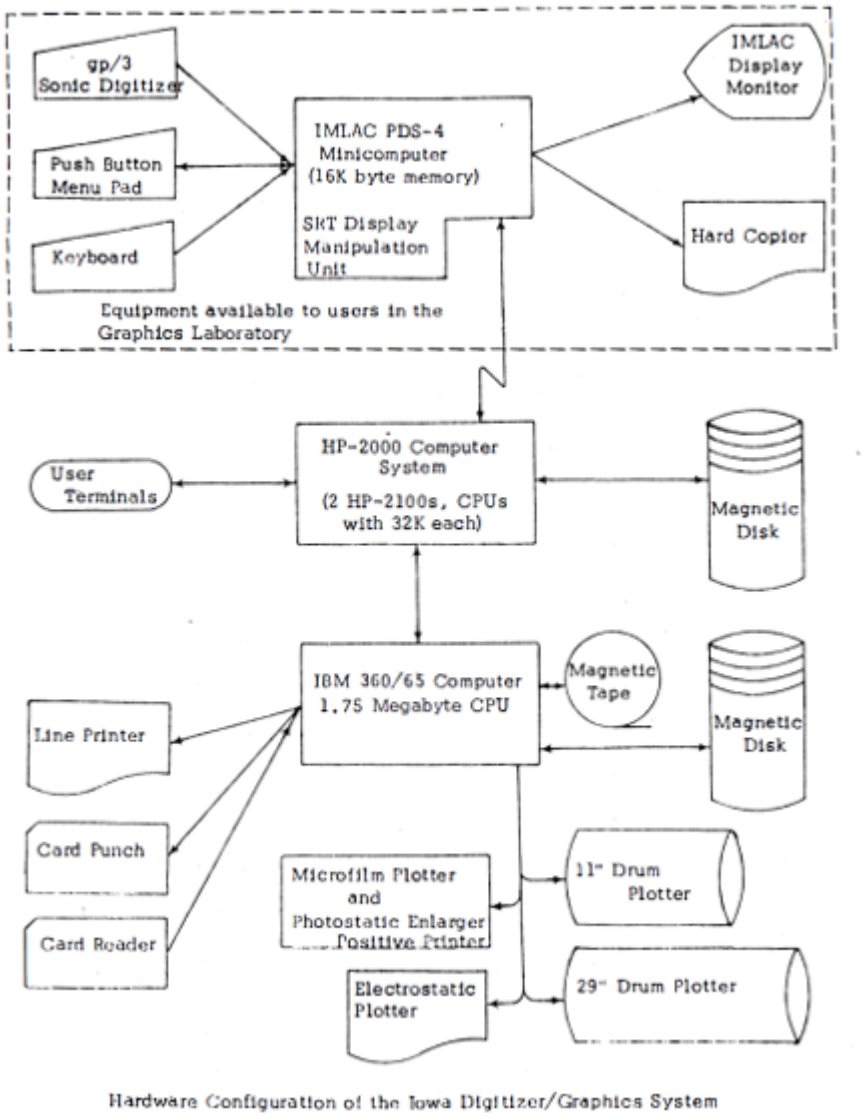


Figure 2. Configuration of hardware in the Department of Geography at The University of Iowa (Source: Dueker & Noynaert, 1977).

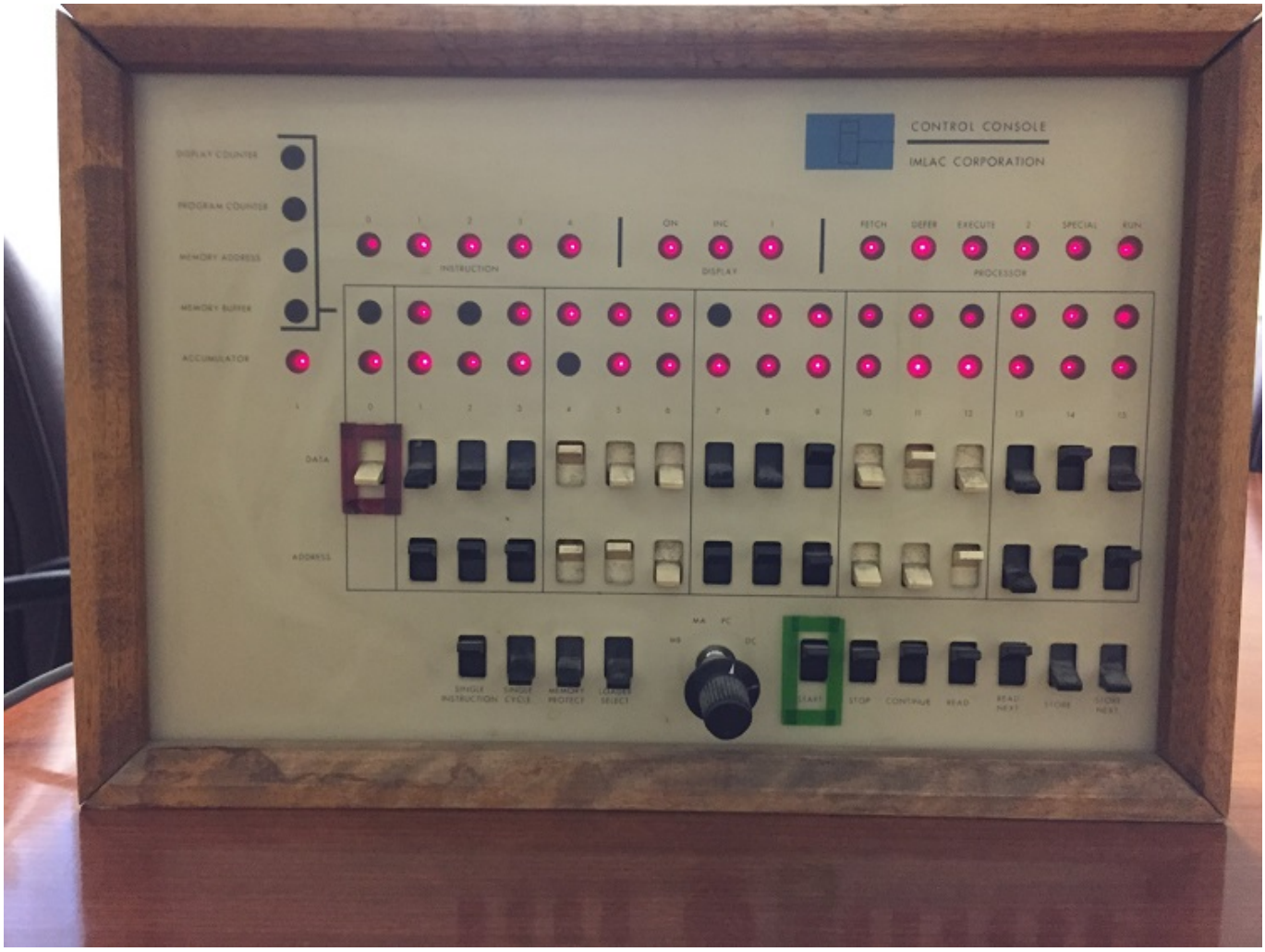


Figure 3. The control console for the IMLAC PDS-4 computer. (Source: Author)

3. Memory & Storage Peripherals

As suggested in the previous section, the amount of memory present in systems of this era was often measured in kilobytes. Disk drives occupied considerable space (think dishwasher), were very expensive, and not particularly capacious. In fact, the New York State LUNR (Land Use and Natural Resource) system data (Tomlinson, Calkins, & Marble, 1976) were stored on two removable IBM 2316 disk packs, each similar in shape to an over-sized curling stone with a capacity of 30 megabytes (this capacity increased in later versions). One model of the 2314 control unit (which housed the 2316 removable disk pack) rented for $5,675 per month (see Additional Resources) or could be purchased for $256,400 ($1.2 million in 2019).

As a consequence of these costs, it was common to use magnetic tape as a means of secondary storage for GIST applications. While tapes are relatively inexpensive, they are limited by low data throughput. And perhaps more importantly, physical access to particular records must be made by sequentially reading through the linear address ranges on the tape. This is sometimes referred to as sequential access memory. Many clever ways of organizing linear structures for two-dimensional data have been devised. One method was developed by Guy Morton for the CGIS in an attempt to physically organize storage

such that records stored close together in a sequential file (1D) are also close in geographic (2D) space (and vice versa) (Morton, 1966; Tomlinson, 1970). Of course, organizing data on hard disks is also important since a good organization strategy can reduce seek times and disk thrashing.

Figure 4 shows a simple example of a Morton sequence in which a combination of decimal, binary and interleaved binary values is used to support efficient conversions between two dimensional cell addresses (row and column) and a position along a linear address space. The X and Y axes are labelled with row and column decimal values and their binary equivalences, and each cell contains a decimal Morton sequence value (top) and its corresponding binary coded decimal value. There are two worked examples shown. For Morton decimal cell 3, its binary representation is 0011. This value also contains its row and column values that are interleaved in the odd and even places of 0011. The two X values are 01_2, which is 1 decimal and the same holds for Y, so its location in the grid is (1,1). Turning to Morton cell 13, which is 1101_2, the X values are 10_2, which is 2 decimal and the Y values are 11_2, which is 3 decimal. The grid coordinate location is therefore (2,3). The order of the scan is shown in the inset on the right side of the figure.

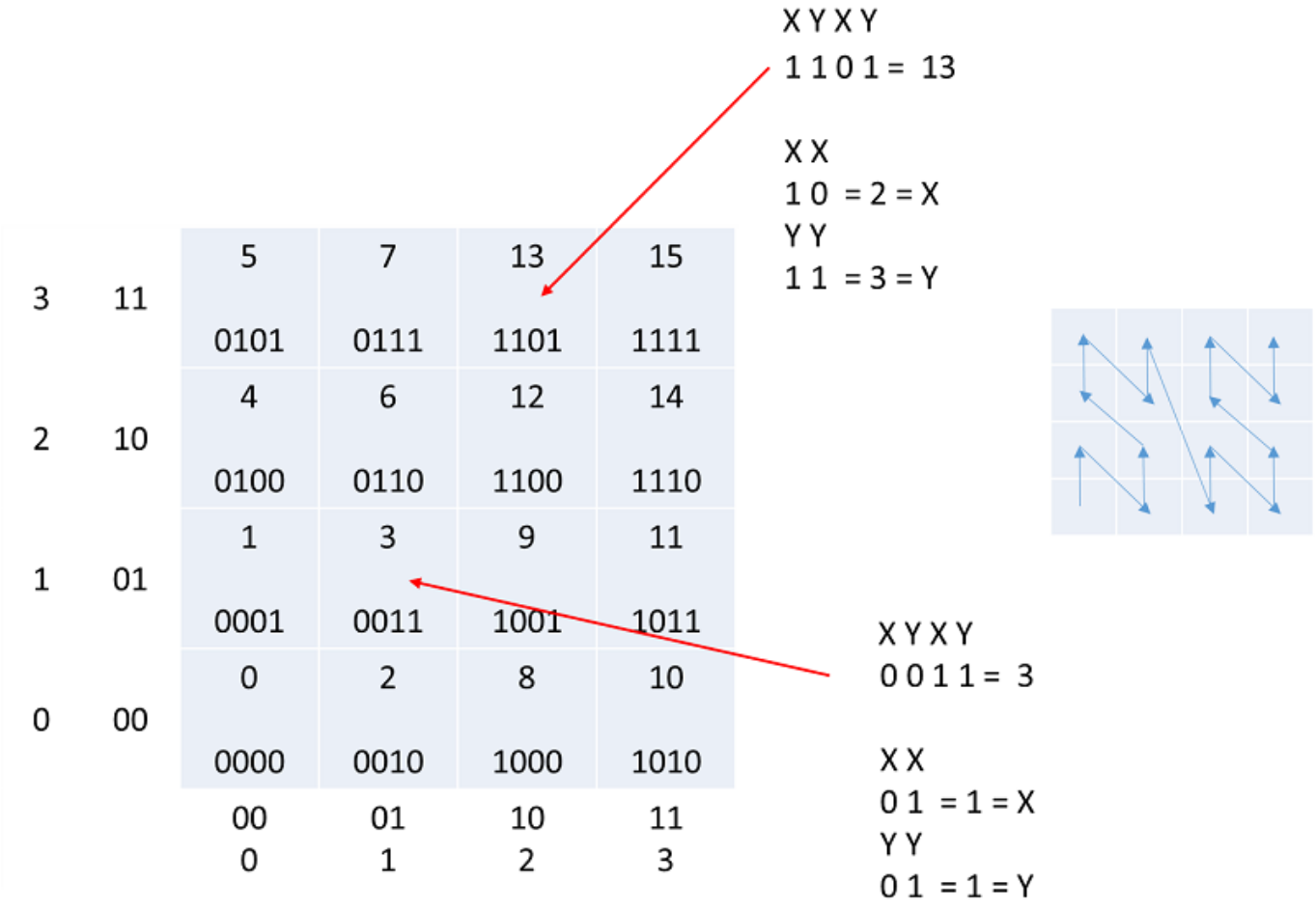


Figure 4. An illustration of Morton sequencing. An actual implementation would have much larger x,y dimensions.

4. Display Hardcopy

Hardcopy maps were produced using two main types of output devices: line printers and pen plotters. Line printers, which were widely available, functioned as their name implied: an entire line of output, usually consisting of 132 characters, was impact-printed simultaneously; then the print form (which normally had green horizontal bars) would advance to print the next line of output. Output speeds in excess of 500 lines per minute were commonly encountered when such devices were used under normal conditions. However, special programming instructions would signal to the printer that it should not advance to the next line, and in such cases “overprinting” occurred. In this case, symbols or characters would print on top of each other to yield coarse-resolution gray tones. This method was reported in Perry and Mendelsohn (1966) and was developed further for pictorial applications by MacLeod (1970) and Singleton (1974). Figure 5 shows an excerpt from a FORTRAN source code comment block that describes the line printer overprint sequence obtained using an IBM 1403 line printer that was capable of printing at 1100 lines per minute. MacDougall (1976, Chapter 4) provides FORTRAN code that illustrates the use of overprinting in statistical map production. In addition, instructions would often be given to operators to turn the forms over (FORMS=REVERSE) so that maps would be printed on white paper and not have green lines running through them (Figure 6).

```
C     THE OVERPRINT CHARACTER SET AND CORRESPONDING PRINT DENSITIES ARE:   PRNP043
C     BLANK      0.00          X          0.40          0+'.          0.67 PRNP044
C     -          0.15          A          0.42          0+'.=         0.79 PRNP045
C     =          0.22          M          0.45          0X'.-         0.85 PRNP046
C     +          0.25          0-         0.53          0X'.HC        0.89 PRNP047
C     )          0.29          0=         0.56          0X'.HB        0.93 PRNP048
C     1          0.33          0+         0.60          0X'.HBV       0.97 PRNP049
C     Z          0.37          0+'        0.64          0X'.HBVA      1.00 PRNP050
C                                                                          PRNP051
```

Figure 5. FORTRAN code fragment (comment block) showing overprint characters and the resulting print densities, which are only approximate given possible overprint misalignments (Source: Singelton, 1974).

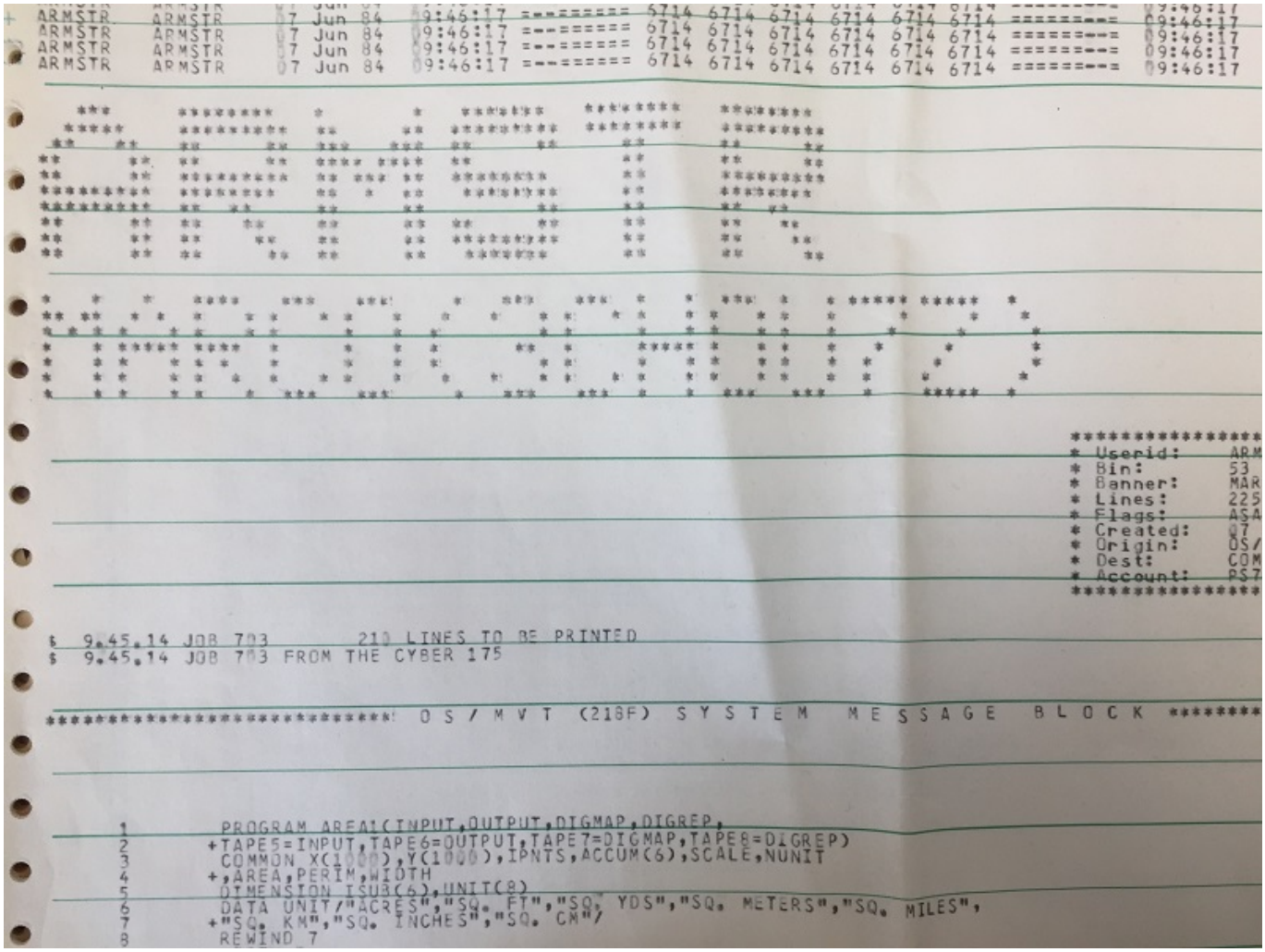


Figure 6. Line printer output of this type was the norm in the 1970s and 1980s (Source: Author).

Pen plotters used liquid ink or ball-point pens to create finer resolution maps under computer control. There were two main types: plotters that had pens mounted on gantries that moved (X,Y) across a flat platen and a more commonly encountered plotter with a drawing surface mounted on a drum and an arm that traversed the Y dimension of the drum. Pen positioning in the X-dimension was achieved by rotating the drum. Plotter resolution was typically 0.025 cm (0.01 in) and the maximum Y value was usually either 25.4 cm (10 in) or 73.66 cm (29 in) depending on drum size. One company (CalComp) commanded a large share of the market and provided a widely-used set of FORTRAN graphics subroutines that controlled the operation of the plotter (CalComp, 1976). At initialization, a reference point is established at (0.0, 0.0). The most commonly used plotting instruction was CALL PLOT (X, Y, IC), where X and Y are the real valued X-coordinate and Y-coordinate locations to which the pen will move and IC is an integer that controls pen mode (up, down) and whether the origin should be re-established at the conclusion of the move.

IC = 2 the pen draws a line to X, Y

IC = -2 the pen draws a line to X, Y and that location becomes the new (0.0, 0.0)

IC = 3 the pen is up (no line) and move to X, Y

IC = -3 the pen is up and that location becomes the new (0.0, 0.0)

Other common subroutines are FACTOR (factor), where factor is a real-valued scale factor and SYMBOL (X, Y, SIZE, STRING, THETA, N) which plots a character string at X, Y that is SIZE inches high and THETA is a plot angle specified with 0 along the X axis and 90 along the Y axis.

CALFORM, a widely distributed choropleth mapping program distributed by the Harvard Laboratory for Computer Graphics and Spatial Analysis, used CalComp plotters (Chrisman, 2006). The program worked by plotting polygon boundaries and applying symbols to the areas in the form of points and cross-hatchings. CALFORM performed well except when the plotter malfunctioned, as shown in Figure 7 when it appears to have not processed IC = 3 instructions. One drawback of the program was its inflexibility, though the source code could be modified, for example, to expand the range of line types available for a plotted map. Figure 8 shows a map in which the lines representing streams have been drawn in a heavier weight, with dashes used to further differentiate streams from basin boundaries. Figure 9 is a listing of the JCL used to compile and execute a file containing the FORTRAN source code for a modification of CALFORM that was being debugged. The first line is the job name, and lines beginning with /* specify parameters establishing, for example, which account to charge, and the amount of memory required in kilobytes. The line beginning with: // EXEC indicates that the source code should be compiled, loaded and executed to produce a plot file. 128 kilobytes of memory is requested. The time (29 minutes) and plot length (119 inches) are set just below the limits that would cause the job to be placed in a slower queue (anything greater than or equal to 30 minutes or 120 inches).

For a reference on FORTRAN code to perform a wide variety of cartographic and other geospatial operations, with a particular focus on pen plotter output, the interested reader is referred to Yoeli (1982). Some of the code is inconsistent with best practices, however. For example, the subroutine that draws a circle does not exploit quadrantal symmetry (Blakemore, 1983).

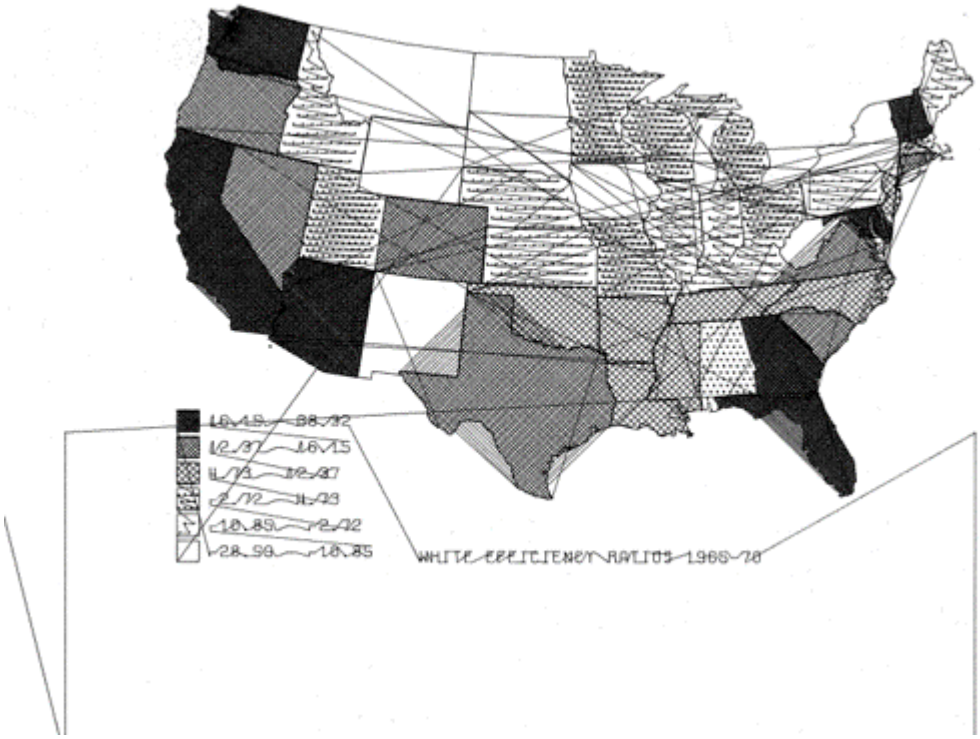


Figure 7. CALFORM plot job with evident plotter error (failure to interpret pen up instruction) (Source: Author).

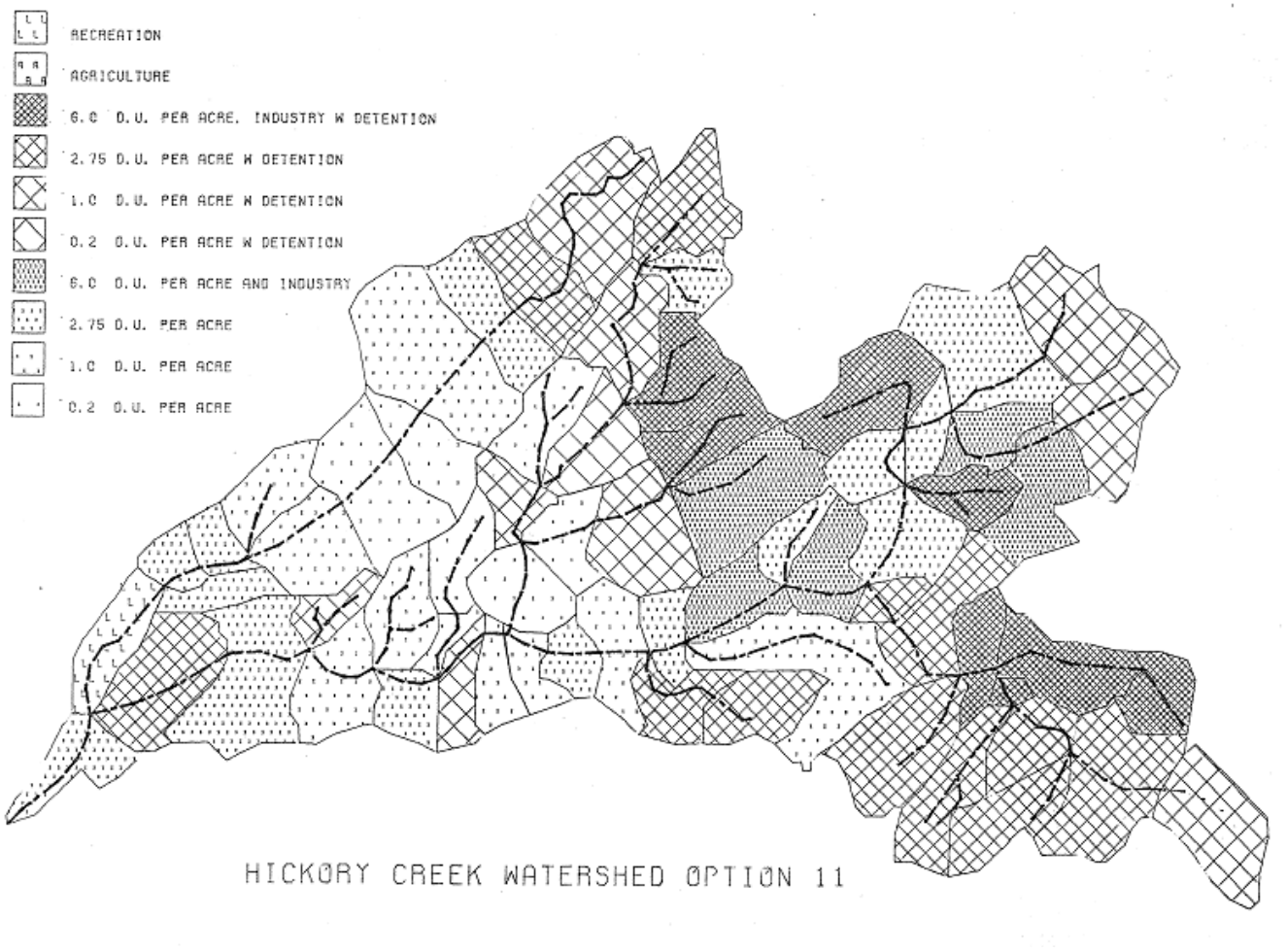


Figure 8. Plot produced with modified CALFORM to represent streams in a basin using different pen types and line patterns (Source: Author).

```
//CALFRM JOB
/*ID PS=1046
/*ID CODE=KANSAS
/*ID REGION=128K
/*ID TIME=(1,5)
/*ID LINES=2500
/*ID PRINT=LOCAL
/*ID PLOT=YES
// EXEC FORTLOPT,REGION=128K,PARM.GO='TIME=00.29.00,PL=119'
C     ------------------------------------------------------    00000100
C     +                    CALFORM                         +    00000120
C     +                  VERSION  1.2                      +    00000140
C     +                28 OCTOBER  1974                    +    00000160
C     +                                                    +    00000200
C     +        A PROGRAM FOR CONFORMANT MAPPING            +    00000220
C     +        ON CALCOMP PLOTTERS                         +    00000240
C     +        ORIGINATED BY  R.S. CARTWRIGHT 1969         +    00000260
C     +        MODIFED BY KATHELEEN REINE, D. HUGHES       +    00000280
C     +        NICK CHRISMAN, AND GEOFFREY CLEMM           +    00000290
C     ------------------------------------------------------    00000300
```

Figure 9. Listing of simple Job Control Language used to compile and execute a FORTRAN program (CALFORM) and produce a file to be plotted on an IBM 360 system at The University of Illinois in 1979 (Source: Author).

As an aside, the ability to program pen plotters such that line spacing for choropleth polygon shading could be varied as a function of a mapped variable served as the rationale for a somewhat controversial paper published in the early 1970s. Tobler (1973) stated directly that “It is now technologically feasible to produce virtually continuous shades of grey by using automatic map drawing equipment. It is therefore no longer necessary for the

cartographer to “quantize” data by combing values into class intervals.” This prompted a response from Dobson (1973) who argued that if the non-quantized approach was adopted, map reading would be more difficult as a consequence of human psychophysical limitations. Other researchers during that time also took note of the ability to vary line spacing to produce gray-scale drawings (see, for example, Phillips, Ransom & Singleton, 1974).

5. CRT Graphics Displays

Softcopy displays have been available from the very early days of computing, with many early systems using cathode ray tube (CRT) displays (see Prince, 1971). And while plasma displays were invented in the early 1960s, and made a brief foray into GIST (Slottow, 1978), Tektronix revolutionized graphics by producing a relatively low cost monochromatic display based on the concept of a storage tube as described in section 6.

Though most CRTs have disappeared because they are large, heavy, heat producers, they were once the dominant technology used for computer-controlled displays. Figure 10 shows a cross-sectional schematic of a primitive stroke vector CRT display where the electron gun produces electrons that are focused and deflected using voltage differentials to address different screen locations at which phosphors are stimulated by the electron beam to emit light. An entire image would be painted repeatedly, though as image complexity increased, the ability of the controller to scan a complete image before phosphors would fade became a problem, particularly if communication was over a phone line.

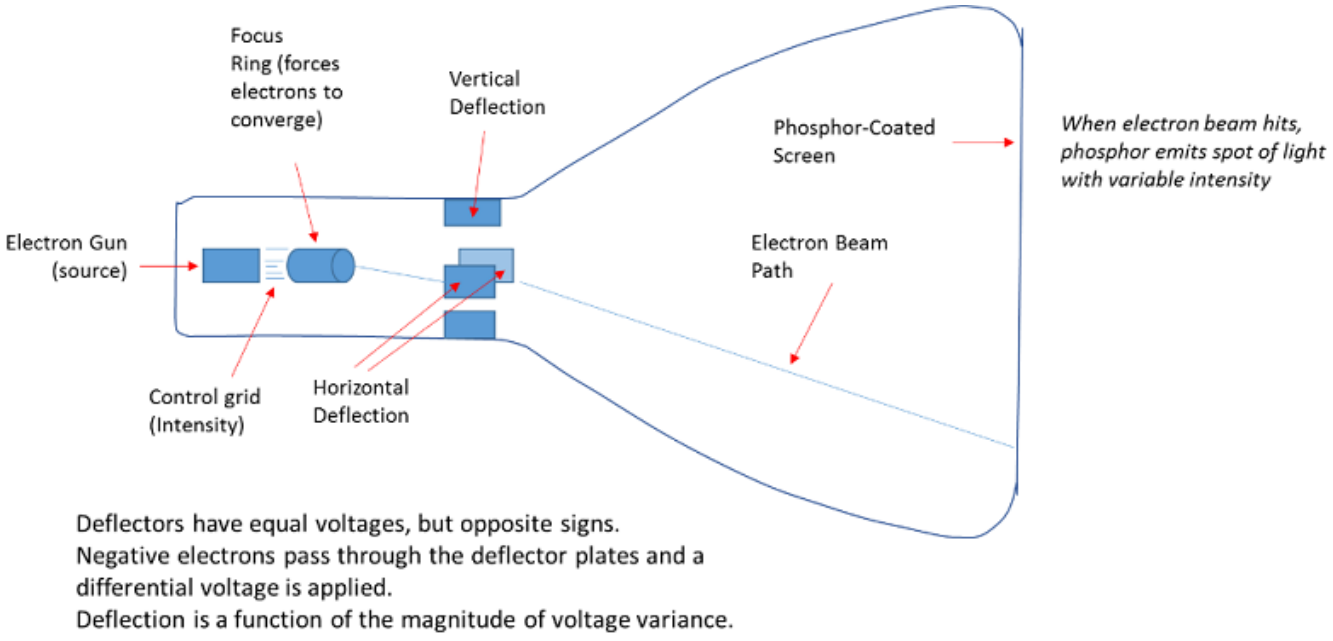


Figure 10. Schematic cross sectional view of a simple CRT display (after Berger, 1986).

Individual screen phosphors have different properties such as persistence and color. High persistence phosphors were not used if screens were refreshed for dynamic graphics; in such cases, the persistence would cause the image to smear. Figure 11 shows how the amount of light output increases rapidly during the excitation phase and after the beam passes, it enters the persistence (decay) phase.

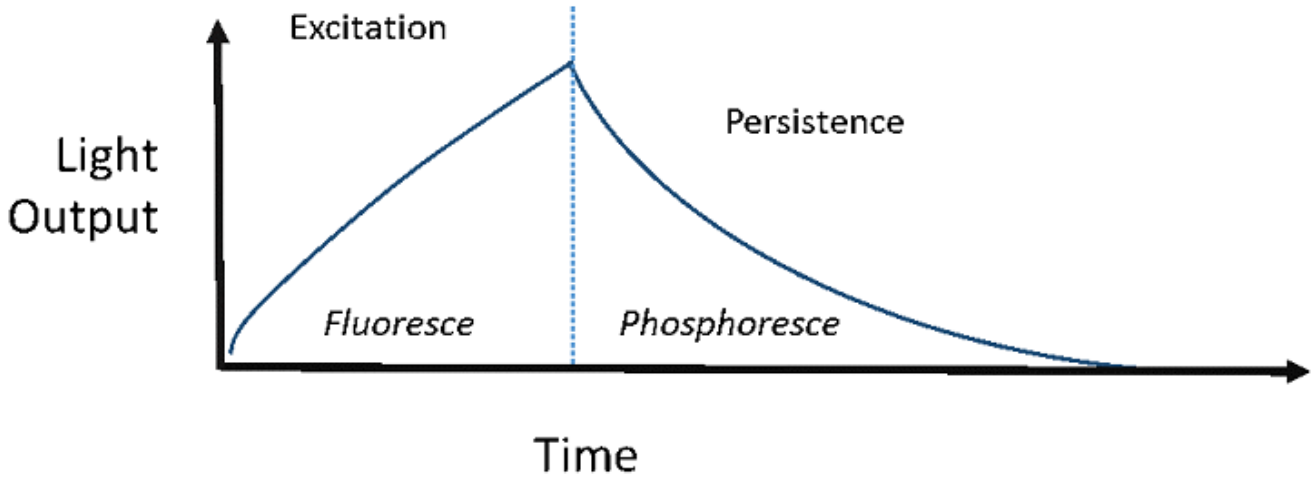


Figure 11. Simplified response graph of a phosphor that has been energized by electron gun output.

More advanced raster graphics have a fixed resolution and depend on the presence of a frame buffer that stores information about the states of individual pixels in a display. Raster CRT costs originally were high because of the expense of the frame buffer memory that needed to be large enough to manage the entire display. In its simplest form, a single bit plane is used and pixels are either on or off (Figure 12). However, when variable gray levels or color are required, multiple bit planes are needed. Figure 13 shows a simplified view for a small frame buffer arrangement consisting of three bit planes which would provide only very limited control over the quantization levels associated with the intensity level for each pixel in a monochrome display. The depth of the buffer determines the number of different gray levels or colors that can be represented. High end graphics in this era would have 2^8 for RGB, which works out to 2^{24} or 16,777,216 colors. A modest resolution system (512 x 512) by 24 deep would require approximately 6 megabytes of memory. Given the expense of memory (PCs in the early 1980s often had 256 kilobytes installed on the motherboard), the original IBM Color Graphics Adapter was limited to 640 x 200 and 2^4 = 16 colors.

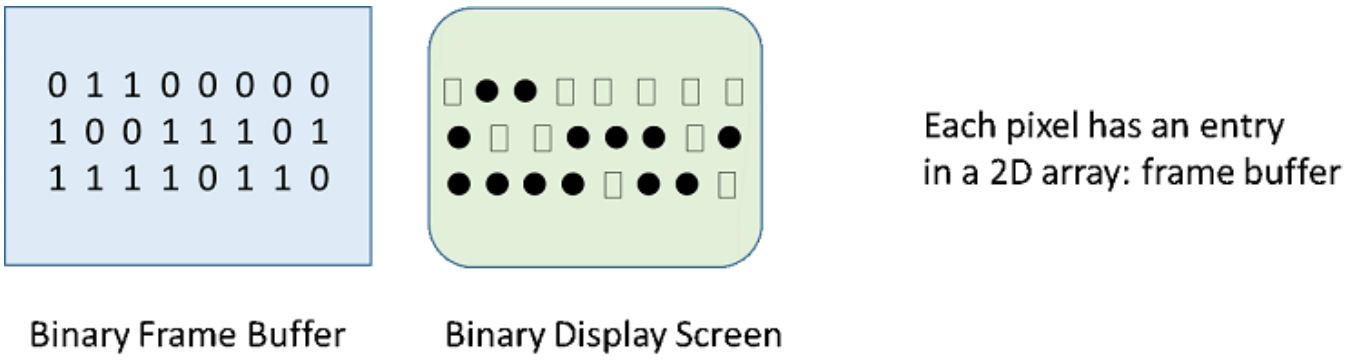


Figure 12. A simple binary frame buffer (after Berger, 1986).

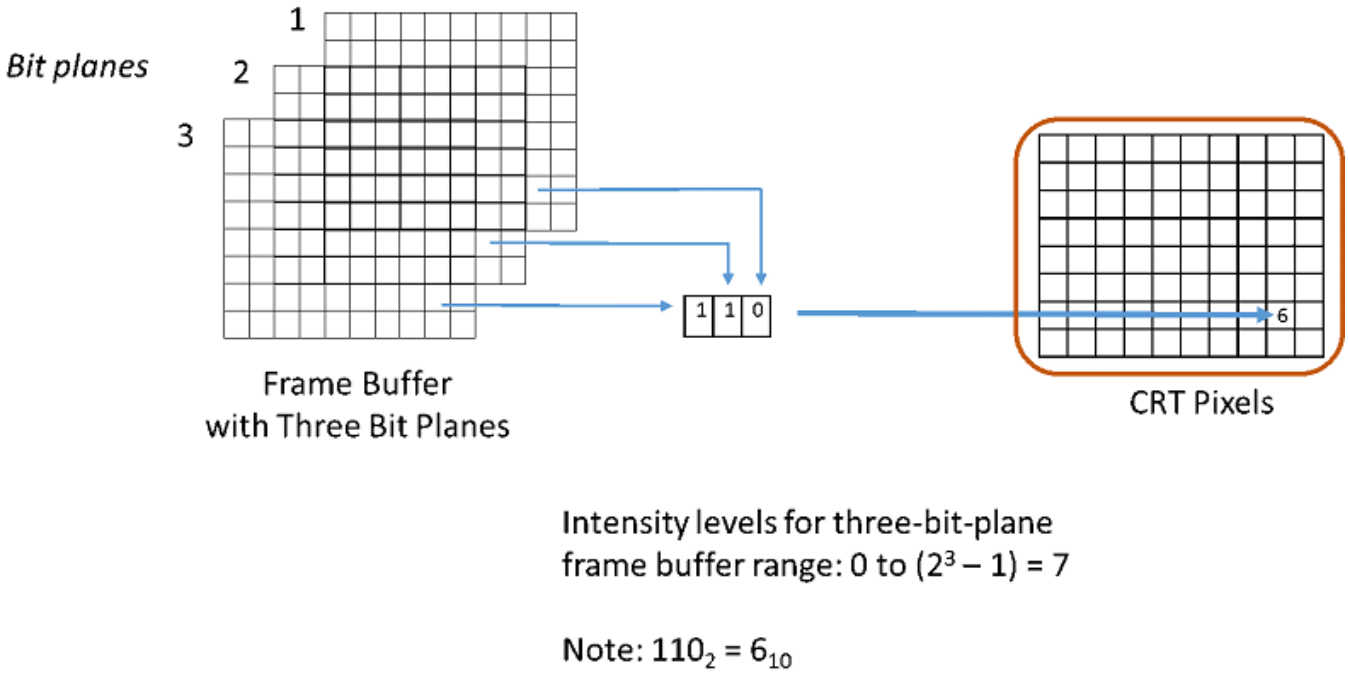


Figure 13. A three-plane frame buffer for a monochromatic display (after Berger, 1986).

6. Storage Tube Displays

As noted earlier, when displays were connected to mainframe computers over conventional dial-up telephone lines, slow communication speeds caused problems with maintaining the current state of the display. These dial-up connections were established using acoustic couplers that worked by placing a conventional old style telephone handset into a device with a microphone and speaker that had been designed with rubberized openings to (mostly) seal transmitted sound impulses. This acoustic coupler converted electrical signals into sounds that were sent across the telephone network to a receiver that converted the sound into an electrical signal. This process is referred to as modulation and demodulation, hence the portmanteau “modem”. Many modems communicated at an upper limit of approximately 300 baud, where baud is an approximation for bits/second. This rate of supply of new bits to a complex display was inadequate. As a consequence, a clever work-around came into common use: the storage tube display.

As shown in Figure 14, a storage tube uses two guns and a storage grid comprised of long-persistence phosphors. An image was built with vector line work that was created with a high intensity writing gun. A secondary gun provided a flood of electrons over the entire grid that was not strong enough to excite a new pixel, but strong enough to continuously refresh those pixels that had already been activated by the writing gun. The storage grid, in this case, acts as an analog frame buffer. While this worked well for some applications, the image on the screen could not be selectively edited. If a change needed to be made, the entire screen needed to be entirely erased by “flashing” it to clear the activated phosphors before rewriting began again on the reinitialized screen. When detailed line work, often required for maps, was displayed on a storage tube, it often took considerable time to build up the image, vector by vector at 300 baud. The advent of 1200 baud modems in the early 1980s was a welcome advance. Textronix dominated the storage tube display market. The

creation of plots for these devices often used a library of plotting subroutines (PLOT-10) that operated at low levels (e.g. draw a line from the current location to a new location specified by X, Y) as well as high levels (e.g. draw a bar chart with the input data contained in vector ZED). For information about PLOT-10 see: http://www.computerhistory.org/collections/catalog/102646117.

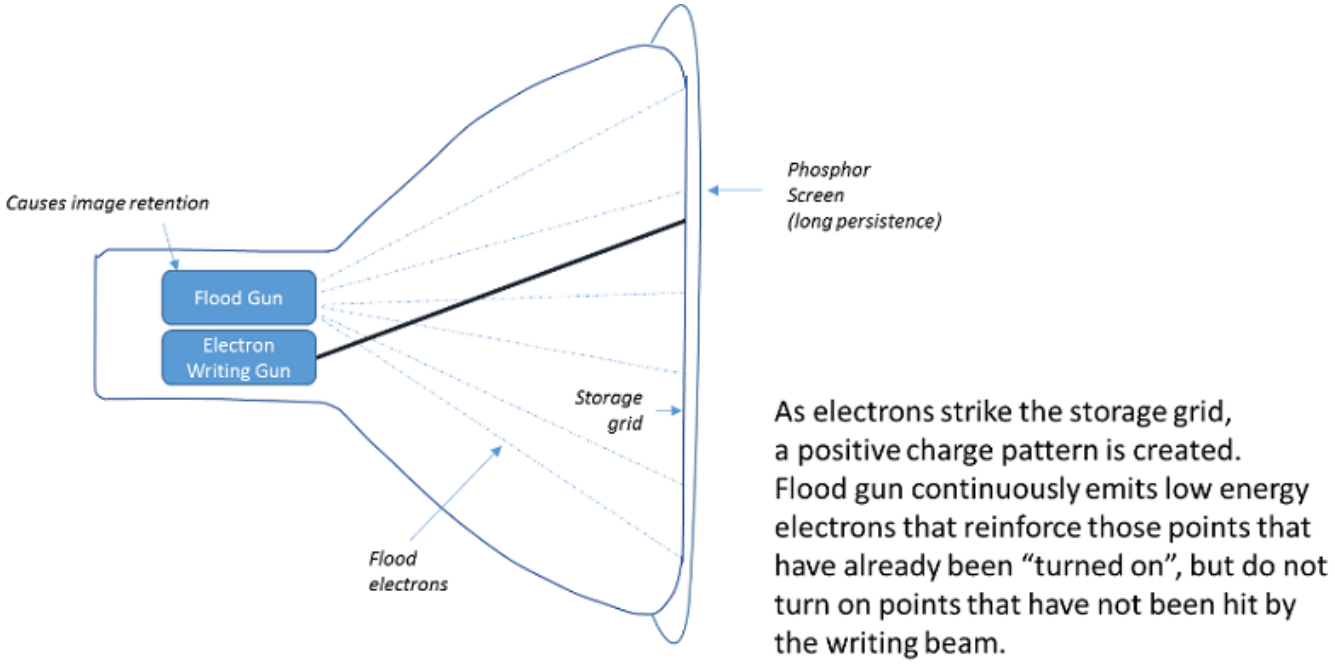


Figure 14. Simplified schematic cross-sectional view of a storage tube display (after Berger, 1986).

7. Input-Output Device Independence

A "Tower of Babel" syndrome pervaded graphics programming in the early days of GIST. Each hardware vendor developed and distributed their own routines that would work with the physical devices they sold. For example, while CalComp would grow to dominate pen plotter market share, hardware from that manufacturer was not universally available and versions of programs had to be "ported" to use hardware from different vendors. This problem spurred computer scientists to advocate for an approach to drawing that was not reliant on the idiosyncrasies of a particular company or type of device, and in the 1970s this gave rise to the concept of "device independence". Ironically, despite the near universal appeal of the idea, several different instantiations of the concept competed for user acceptance. For example, the ACM-SIGGRAPH Graphics Standards Planning Committee worked on the development of the Core Graphics System (Bergeron, Bono, & Foley, 1978) that was, in part based on earlier work by Bergeron (1976) and others. At the same time, other projects competed for user acceptance. The Graphics Compatibility System (GCS) was widely used on campuses nationwide (GCS, 1979; Puk, 1979a; 1979b) and the Graphical Kernel System (GKS) was eventually adopted by the International Standards Organization (see Sproull, Sutherland, & Ullner, 1985). GKS was widely used on IBM personal computers during the 1980s.

The concept underlying device independence is relatively straightforward when viewed at a high level of abstraction, though details become more complex as specific hardware implementations are considered. As shown in Figure 15, device drivers handle the specific details of the individual hardware items, though the goal is to hide this complexity from the applications programmer. The concept of “write once, run anywhere” applies here.

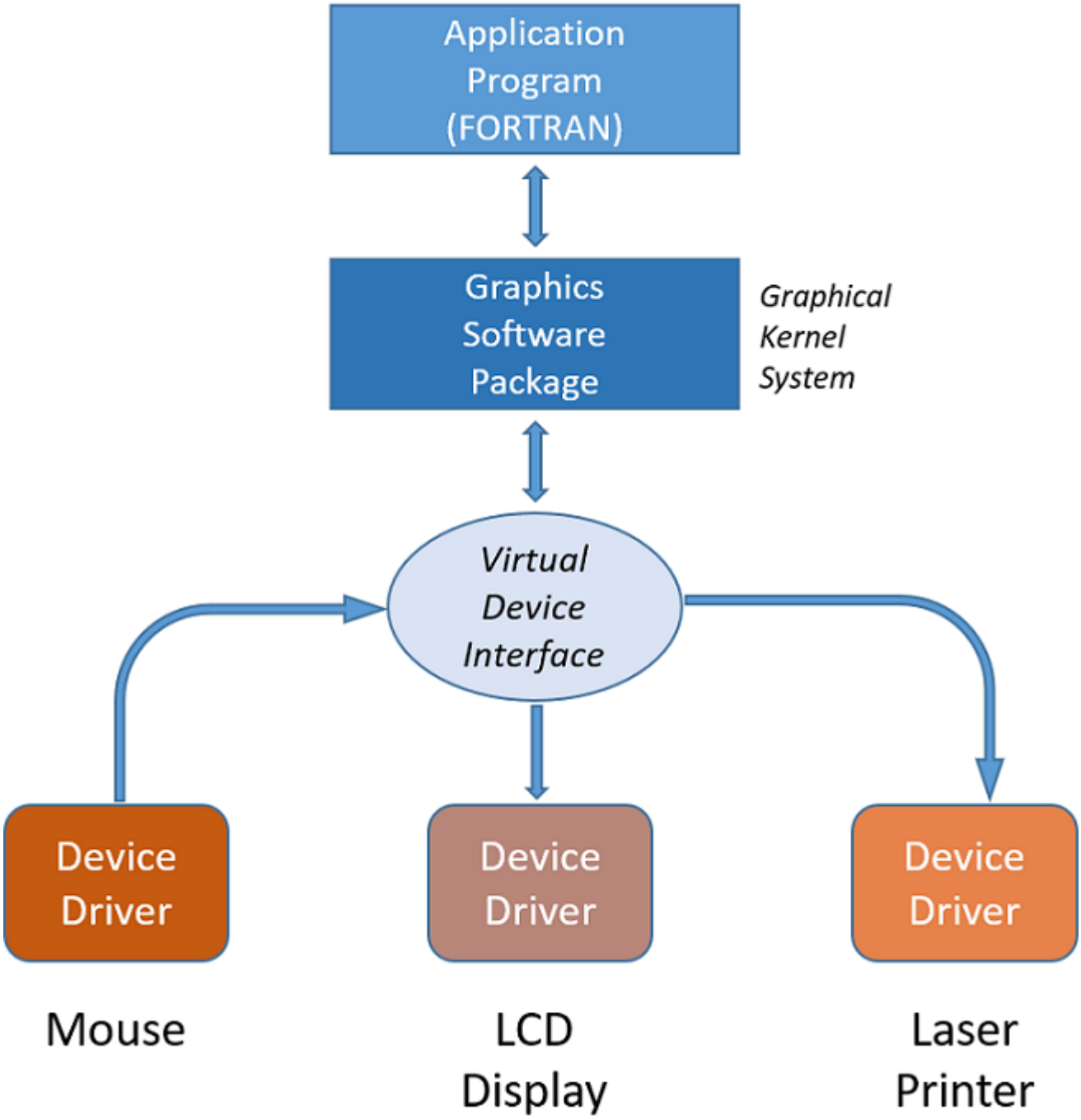


Figure 15. Simplified view showing levels of abstraction associated with the device independent graphics.

8. Summary

This paper provides a brief overview of the role of peripheral devices in the development of GIST implementations during the late 1960s to the mid 1980s, a period that roughly corresponds to the use of mainframe uniprocessor architectures, the rise of minicomputers and the eventual introduction of UNIX workstations that used commodity microprocessors. In each case, the original peripheral devices were scarce, had limited capabilities and were very expensive. These attributes had important effects on what could and could not be done with GIST during that era. Indeed, one pioneer (Dueker, 2019) recently stated: “My whole GIS career was waiting for more computing power and better data. Now we have it.” GIST researchers and users have long benefitted from a virtuous cycle of technological innovation, followed by increased sales volumes, which led to new investments and further improvements in the performance of devices large and small. This trend has continued to the current era where inexpensive peripherals with orders of magnitude in price-performance improvements (e.g., color laser printers and flash drives) now dominate the market.

References

Berger, M. (1986). Computer Graphics with Pascal. Menlo Park, CA: Benjamin/Cummings.

Bergeron, R. D. (1976). Picture primitives in device independent graphics systems. In Proceedings of the ACM Symposium on Graphic Languages (57-60). Association for Computing Machinery.

Bergeron, R. D., Bono, P. R., & Foley, J. D. (1978). Graphics programming using the Core System. Computing Surveys, 10(4), 389-443.

Blakemore, M. (1983). Book review: Cartographic drawing with computers. Earth Surface Processes and Landforms, 8(6), 606-607.

CalComp. (1976). Programming CalComp electromechanical plotters. Anaheim, CA: California Computer Products, Inc.

Chrisman, N. R. (2006). Charting the Unknown: How Computer Mapping at Harvard became GIS. Redlands, California: ESRI Press.

Defense Technical Information Center (DTIC). (1979). Graphics Compatibility System (GCS) Programmer's Reference Manual.

Dobson, M. W. (1973). Choropleth mas without class intervals? A comment. Geographical Analysis, 5(4), 358-360.

Douglas, D. H. (1982). Automated mapping and geographic information processing utilizing timesharing with a centralized computer system. In D. H. Douglas, & A. R. Boyle (Eds.), Computer assisted cartography and geographic information processing: Hope and realism (51-54). Ottawa, ON: Canadian Cartographic Association.

Dueker, K. J. (1975). Geographic data encoding issues. Technical Report 43. Iowa City, IA: Institute of Urban and Regional Research.

Dueker, K. J. (2019). Personal email communication.

Dueker, K. J. & Noynaert, J. E. (1977). Interactive digitizing, editing and mapping: Software and data structure considerations. Technical Report 92. Iowa City, IA: The Institute of Urban and Regional Research.

Eyton, J. R. & Roseman, C. C. (1979). An introduction to FORTRAN and the programming of spatial data. Paper Number 13, Occasional Publications of the Department of Geography. Urbana, IL: University of Illinois at Urbana-Champaign.

ISIS (1982). Encoding manual: River mile index reference maps. Working Paper No. 2. Urbana, IL: Illinois Streams Information System.

Jenks, G.F. (1981). Lines, computers, and human frailties. Annals of the Association of American Geographers, 71(1), 1–10.

Jenne, R. L., Joseph, D. H., Ridley, E. C., & Fabec, R. C. (1972). The Bendix Datagrid Graphic Digitizing System Operational Procedure and Available Software. NCAR Technical Note NCAR/TN-78+IA.

Lee, M. P. (1985). The Scitex raster graphic processing system. In Proceedings of the Workshop on Methods and Techniques for Digitizing Cartographic Data (1-14). Open File Report 85-307. Reston, VA: United States Geological Survey.

MacDougall, E. B. (1976). Computer Programming for Spatial Problems. London, UK: Edward Arnold.

MacLeod, I. D. G. (1970). Pictorial Output with a Line Printer. IEEE Transactions on Computers, vol. C-19, no. 2, pp. 160-162.

Meyers, C. R., Wilson, D. L., & Durfee, R. C. (1976). An application of the ORRMIS geographical digitizing and information system using data from the CARETS project. ORNL/RUS-12. Springfield, VA: National Technical Information Service.

Morton, G. M. (1966). A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. Ottawa, CA: International Business Machines Co. Ltd.

Perry, B., & Mendelsohn, M. L. (1964). Picture generation with a standard line printer. Communications of the ACM, 7(5), 311-313.

Peuquet, D. J. (1981a). An Examination Of Techniques For Reformatting Digital Cartographic Data / Part 1: The Raster-To-Vector Process. Cartographica, 18(1), 34-48.

Peuquet, D. J. (1981b). An Examination Of Techniques For Reformatting Digital Cartographic Data / Part 2: The Vector-To-Raster Process. Cartographica, 18(3), 21-33.

Peuquet, D. J. & Boyle, A. R. (1984). Raster scanning, processing and plotting of cartographic documents. Williamsville, NY: SPAD Systems, Ltd.

Phillips, J. W., Ransom, P. L., & Singleton, R. M. (1975). On the construction of holograms with an ink plotter. Computer Graphics and Image Processing 4(2): 200-208.

Prince, M. D. (1971). Interactive Graphics for Computer-Aided Design. Reading, MA: Addison-Wesley.

Puk, R. F. (1979a). Host-computer implementation guidelines for the three-dimensional graphics compatibility system (GCS). Defense Technical Information Center.

Puk, R. F. (1979b). Device implementation guidelines for the three-dimensional graphics compatibility system (GCS). Miscellaneous Paper 0-79-2. DTIC_ADA070867.

Singleton, R. M. (1974). A FORTRAN routine for simulating pictorial distributions on a line printer. Electromagnetics Laboratory Scientific Report No. 74-9. Department of Electrical Engineering, University of Illinois at Urbana-Champaign, Urbana, IL.

Slottow, G. (1978). Plasma displays. Proceedings of Auto-Carto III (362-386). Fall Church, VA: American Congress on Surveying and Mapping.

Sproull, R. F., Sutherland, W. R., & Ullner, M. K. (1985). Device Independent Graphics. New York, NY: McGraw-Hill Book Company.

Tobler, W. R. (1973). Choropleth Maps Without Class Intervals? Geographical Analysis, 5 (3), 262-265.

Tomlinson, R. F. (1998). The Canada geographic information system. In T. Foresman (Ed.), The History of Geographic Information Systems: Perspectives from the Pioneers (21-32). Upper Saddle River, NJ: Prentice Hall.

Tomlinson, R. F. (Ed.). (1970). Environment Information Systems. In Proceedings of the UNESCO/IGU First Symposium on Geographical Information Systems. IGU Commission on Geographical Data Sensing and Processing.

Tomlinson, R. F., Calkins, H. W., & Marble, D. F. (1976). Computer handling of geographical data. Paris, France: The UNESCO Press.

Yoëli, P. (1982). Cartographic drawing with computers. Computer Applications, Volume 8. Nottingham, GB: Department of Geography, University of Nottingham.