A MICROCOMPUTER GRAPHICS PACKAGE FOR USE WITH A HIGH-RESOLUTION RASTER-SCAN DOT-MATRIX PRINTER

by

EARL F. GLYNN II

B.S., Kansas State University, 1975

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by:

Major Professor

TABLE OF CONTENTS

Contents (continued)

1. Introduction

1.1 Purpose, Motivation and Objectives

A general purpose graphics software package was developed for use with a low-cost "personal" microcomputer system without the use of any special purpose hardware. This project was motivated by the acquisition of a graphics printer without adequate software to control its many features.

Software was developed to provide a mechanism to define a picture in terms of a logical screen -- which may or may not correspond to a video display screen -- which could be mapped to a graphics printer. Ideally, the logical screen would be mapped to either a video display terminal or a hard copy device.

The initial implementation supports only "black" and "white" pixels but the software was designed to support a more complex pixel definition. Two- and three-dimensional graphic primitives support user picture definition. Various vector/matrix operations support mathematical transformation of the picture. While the software package was developed for one particular graphics printer, other printers could be used by changing some internal constants and variable definitions and recompiling. Output to a video display terminal is also possible but is not included in this initial implementation.

In this report we assume the reader is either knowledgeable about computer graphics or has access to various textbooks on the subject. Detailed explanations of graphic operations are not included in this report. [NEWM79] and [FOLE82] are excellent references. The following are helpful and interesting: [ANGE81, GILO78, POSD77, ROGE76, SCOT82, WHIT82].

1.2 Background

In the last few years dot-matrix printers have been introduced for high-quality word processing and graphics applications. These printers have a variety of dot resolutions such as 74-by-72 (Centronics 739) 72-by-60/120 (HP 82905), and 84-by-84 (Integral Data Systems 460 or 560) dots per inch (DPI) for print areas as wide as 13.2 inches. In the last year new "quad density" printers have been announced. The NEC 8023 has 144-by-60 DPI resolution; the C. ITOH 8510 has 144-by-144 DPI; the Axiom IMP-4 has 19008 dots/square inch; and an upgrade to the Digital LA 120 offers 165 DPI resolution. These printers and others are currently selling from about $350 to $1500. In early 1982 Integral Data Systems announced a four-color ribbon with its high resolution "Prism" printer [UMLO82]. Mixing allows eight possible colors. The printer sells for about $2000. Higher resolutions and greater color capabilities will probably be developed in the near future.

These dot matrix printers have resolutions that favorably compare to the resolution available on video display terminals currently available. These printers are cheaper than the display terminals and provide hard copy output directly, however, these printers are much slower than display terminals.

1.3 Software Overview

The user defines a picture using world coordinates -- any convenient coordinate system. If desired these world coordinates can be manipulated mathematically using translations, rotations or scaling before they are converted to screen coordinates. The logical screen is defined to be a matrix of picture elements -- pixels. Each pixel is small enough (about 0.01 inch by 0.01 inch) that a circular dot represents what is actually a

rectangular area.

The screen coordinates of points and lines are defined by a set of pixels. Clipping may be necessary to eliminate points or lines which extend outside the logical screen pixel matrix.

The user defines the actual physical size of the logical screen and, therefore, implicitly defines the maximum size of the plot file. The maximum width of the picture is limited horizontally by the printer line length (in practice but not in theory) but vertically only by available diskette storage. Since the number of pixels in the matrix could reach two million or more, and each pixel requires at least one bit of storage, the entire pixel matrix requires perhaps 250 Kbytes of storage. This amount of memory is not generally available on most micros (but probably will be in a few years). To accommodate a potentially very large pixel matrix on existing micros, a demand paging system was developed so that as many pixels as possible can remain in memory for manipulation. Diskettes for micros typically hold 92-256 Kbytes but hard disks with capacities of 5 Mbytes or more are entering the microcomputer market. Efficient diskette storage and in-memory data structures allow some data compression and creation of some pictures which otherwise would exceed available diskette storage.

2. A Graphics Tutorial

Computer graphics is the creation, storage and manipulation of models of objects and their pictures via a digital computer. Computer graphics is used in such diverse areas as mathematics, medicine, architecture, engineering, chemistry, cartography, business, word processing, art, animation and entertainment. Computer graphics is

becoming the preferred interface between humans and computers instead of being considered a special form of communication requiring special I/O software and hardware. Data presented pictorially can be perceived and processed by humans more rapidly and efficiently than textual data. Graphics systems will be more widely available as microcomputer hardware costs continue to tumble. One is prompted to look for computer graphics in unexpected places within the home, office and laboratory in the next few years.

2.1 Interactive Versus Passive Computer Graphics

Interactive computer graphics allows a user to dynamically control a picture's context, format, size or color by means of interaction devices such as a keyboard, lever or joystick. Video displays such as CRTs and TV sets are used to show the picture dynamics. Points and lines must be continuously updated to add a dimension of time to the display. Applications of interactive graphics include flight training simulation, computer-aided design and video games.

Passive computer graphics is involved when using an impact printer or a drum or flatbed plotter. The user controls the picture creation but does not have real-time, dynamic options. Passive graphics is easier to implement than interactive since event handling is not necessary. Processor speed for passive graphics is not as critical since points and lines are not continuously updated. Passive graphics is all that is required in many applications, e.g., graphs, pie charts, histograms, flowcharts, architectural diagrams and circuit schematics.

The remainder of this tutorial emphasizes subject areas used in both interactive and passive graphics. Certain areas of interest only to interactive graphics (e.g., event handling) are not addressed since the

software system was developed to operate as a passive computer graphics package.

2.2 Vector Versus Raster Display

The term "vector" is not used strictly in the mathematical sense of an n-tuple of location coordinates. Graphics literature uses "vector" to describe a line segment or the process of drawing a segment.

Pictures can be created by a vector system with a random-scan -- segments are displayed in any order. A pen plotter in which a pen can be moved in any direction over a piece of paper is a random-scan vector device.

In a raster-scan system a drawing is divided into horizontal lines. Each raster scan traces out a small strip of a picture. U.S. TV sets, for example, have 525 lines and most CRT raster systems use between 256 and 1024 lines. The more lines, the higher the picture quality.

With a black-and-white CRT raster device, a raster scan is a left-to-right sweep of the electron beam which is modulated to create different shades of gray. There is a one-to-one mapping of a memory location to each small segment of the raster scan. Each of these small segments is called a "pixel" -- a picture element. A picture is therefore a matrix of pixels. See [NEWM79, Chapters 15-19] or [FOLE82, Chapters 10-12] for details of raster graphics.

Dot-matrix impact printers are also raster devices. Each pass of the print head traces out typically 7 to 9 rows of dots to form text characters. In the last few years microprocessor-controlled, dot-addressable, impact printers have been introduced as graphics devices.

The IDS 560 printer has a graphics resolution of 84 dots per inch both vertically and horizontally. Dots are formed by print head wires 14

mils in diameter; the printed dot is anywhere between 15 and 17 mils in diameter due to inherent variations in paper hardness, humidity and ribbon wear. Dots are printed on 1/84th-inch centers, about 12 mils apart. (A mil is 0.001 inch).

In graphics mode each pass of the IDS 560 prints 7 rows of dots. Each column of 7 dots within each row is mapped from a byte of memory. Given a byte with bits 7 to 0 (bit 7 the most significant and bit 0 the least significant), bit 0 is mapped to the top row of the raster scan while bit 6 is mapped to the bottom row of the scan. Bits 1 through 5 are mapped in between. Bit 7 (typically the parity bit) is ignored. Other details of the IDS 560 graphics mode can be found in [IDS81].

While the IDS 560 prints 7 rows of dots per raster scan, the microcomputer graphics package was developed to operate as a random-scan vector system.

2.3 Two-Dimensional Picture Definition

Output from graphics systems is in a two-dimensional form whether on a CRT screen, a drum or bed plotter, or an impact printer. This section will introduce concepts and algorithms used in two-dimensional graphics. Three-dimensional graphics involves projections into two dimensions and will be discussed in Section 2.5.

2.3.1 World Coordinates and Windows

The term "world coordinates" is used to describe the Cartesian coordinate system used by a user. The units of the world coordinates can be anything appropriate for a problem definition, e.g., inches, meters, gallons, liters, pounds, newtons, etc. A user should concentrate on the definition of the entities to be plotted and should not be overly concerned about conversions which are automatically performed by a

graphics package.

Many graphics systems require the user to specify coordinates in a device space -- the coordinates needed by the display hardware. The user's data is rarely within the same range as needed by the display hardware and must be mathematically manipulated to fall within the desired range.

The rectangular area bounding the extents of a user's world coordinates which defines the desired picture is called the "window". (Points outside the "window" cannot be "seen" by the graphics software, i.e., they are ignored.) Without an appropriate transformation, the default "positive" direction of the coordinate axes is left-to-right for the "x" axis and bottom-to-top for the "y" axis. A canonical space ranging from 0.0 to 1.0 in both the "x" and "y" Cartesian dimensions is an appropriate default window. However, a user can specify any window by setting the desired minimum and maximum values of the "x" and "y" dimensions.

2.3.2 Points and Lines

The entities described by a user in world coordinates consist of a set of points and lines. A point in world coordinates is mapped into a point in device space. If the dimensions of device space do not correspond to world coordinates, then a simple translation and/or scaling operation will allow world coordinates to be mapped to device space.

A line segment consists of two endpoints and all the collinear points in between. Transforming a line segment from world coordinates to device space is simple: The endpoints of the segment in world coordinates are each mapped to device space. The segment in device space consists of these transformed endpoints and all the collinear points in

between.

When using a random-scan vector device, the points or line segments in device space can be directly plotted. No calculations are necessary to define intermediate segment points. This is not true when using a raster-scan device with a picture consisting of a matrix of pixels. The transformed segment endpoints map directly to specific pixels but all the intermediate pixels between the endpoints must be individually selected.

2.3.3 Clipping

Unless a user is extremely careful lines may extend outside the defined window area. The portion of the line outside the window must be clipped. A common method used for line clipping is the Cohen-Sutherland algorithm. See [NEWM79, pp. 65-67] or [FOLE82, pp. 146-149] for a detailed description of this algorithm. (The Pascal programs in both texts are essentially the same. [FOLE82] "fixes" the single "goto" of [NEWM79].) A brief description follows.

This algorithm first considers the regions in which the line endpoints lie. These regions classically have been assigned binary codes as shown in both [NEWM79] and [FOLE82]. The binary codes seem to complicate the discussion. In this report we replace a set of region codes with a set of directions as if the window were a map surrounded by regions. For example, the window '0000' becomes [], '0010' becomes [east], '0110' becomes [south east], '0100' becomes [south], etc. Consider the rectangular window and the sets of regions shown in the diagram below:

[north west]	[north]	[north east]
[west]	[]	[east]
[south west]	[south]	[south east]

If the set of regions a point lies in is the empty set [], the point is contained within the window. If the set of regions is non-empty, the point lies outside the window. If the union of regions from both endpoints of a line is the empty set, the segment is entirely visible. If the intersection of regions from two points is not the empty set, the segment must lie entirely outside the window and is invisible. Thus, lines which are entirely visible or invisible are quickly processed.

If the line is partially visible, the point of intersection with one edge of the window is found and the segment that lies outside the window is discarded. The algorithm then repeats: The initial visibility test is then applied to the remaining segment and further subdivisions are made until only the visible part of the segment remains.

2.3.4 Viewports

A "window" is a logical (or virtual) screen and is mapped to a physical (or real) screen or a portion of a physical screen. This area of the physical screen onto which a window is mapped is termed a "viewport" or "view". Often the window fills the whole physical screen and the window and viewport have similar definitions. Sometimes many viewports fill a single physical screen.

Since the microcomputer system used in developing this software package did not support a graphics CRT but rather a graphics printer, a "viewport" is defined slightly differently for this package. A "viewport" is treated as a rectangular subset of a "window". The "view"

is used by the clipping modules instead of the "window". This allows a user to define one logical screen containing any number of other logical screens. The intent was to allow a user to restrict graphic operations to only a portion of the window. Many diagrams could be included in one final hard copy plot.

2.4 Matrix Transformations

2.4.1 Homogeneous Coordinates

The representation of an n-component position vector by an (n+1)-component vector is called homogeneous coordinate representation [ROGE76]. In homogeneous coordinate representation the transformation of n-dimensional vectors is performed in (n+1)-dimensional space and the transformed n-dimensional results are obtained by projection back into the particular n-dimensional space of interest. Thus, in two dimensions the position vector [x y] is represented by the three-component vector [hx hy h]. There is no unique homogeneous coordinate representation of a point in two-dimensional space. For ease of calculation and simplicity [x y 1] is used to represent a nontransformed point in two-dimensional homogenous coordinates.

The advantage of introducing homogeneous coordinates occurs in the general 3 x 3 transformation matrix

$$[x'\ y'\ h] = [x\ \ y\ \ 1] \begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

where terms a, b, c and d produce scaling, shearing and rotation, m and n produce translation, and p and q produce a projection. The element s produces overall scaling.

2.4.2 Translation

Translation is the uniform motion of an object along a straight line. A translation could not occur with a transformation matrix without the use of homogeneous coordinates. Given the translation vector $\underline{T}$ with translation components $\underline{Tx}$ and $\underline{Ty}$, the matrix transformation for translation is

$$[x'\ y'\ 1] = [x\ \ y\ \ 1] \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{vmatrix}$$

2.4.3 Scaling

A scaling operation can represent a change in units, or an enlargement or shrinking of the dimensions of an object. Negative scaling values can be used for mirror image "reflections". Given the scaling vector $\underline{S}$ with scaling components $\underline{Sx}$ and $\underline{Sy}$, the matrix transformation for scaling is

$$[x'\ y'\ 1] = [x\ \ y\ \ 1] \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2.4.4 Rotation

Rotation means that each point of an object moves in a circular path around the center of rotation. A rotation transformation matrix can be derived from simple geometry. The transformation matrix *to* rotate point (x,y) through a clockwise angle $\emptyset$ about the origin of the coordinate system is

$$[x'\ y'\ 1] = [x\ \ y\ \ 1] \begin{vmatrix} \cos\emptyset & -\sin\emptyset & 0 \\ \sin\emptyset & \cos\emptyset & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

This transformation matrix can only be used for a rotation about the

origin.

2.4.5 Change in Coordinate Systems

In the above transformations the coordinate system stays unaltered and the object is transformed with respect to the origin. An alternate but equivalent way of thinking of a transformation is a change in coordinate systems. This view is useful when multiple objects, each defined in its own local coordinate system, are combined into a single, global coordinate system.

A translation $\underline{T}$ of a point is a translation $\underline{-T}$ of the coordinate system. A rotation angle $\underline{\emptyset}$ of a point is a rotation angle $\underline{-\emptyset}$ of the coordinate system.

2.4.6 Concatenation of Simple Transformations

Transformations can be combined by matrix multiplication of simple transformation matrices. The order of transformations must be preserved since matrix multiplication is not commutative.

To demonstrate concatenation of transformations consider the rotation about an arbitrary point instead of the origin. Given point (x,y) to be rotated by an angle $\emptyset$ about the point (Rx,Ry). This can be accomplished by translating the origin to the point (Rx,Ry), performing the rotation, and then restoring the original origin:

$$[x'\ y'\ 1] = [x\ \ y\ \ 1] \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -Rx & -Ry & 1 \end{vmatrix} \begin{vmatrix} \cos\emptyset & -\sin\emptyset & 0 \\ \sin\emptyset & \cos\emptyset & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Rx & Ry & 1 \end{vmatrix}$$

2.5 Extension to Three-Dimensional Graphics

Many two-dimensional operations can simply be generalized to produce three-dimensional operations. Some three-dimensional operations do not have two-dimensional analogs, however.

2.5.1 Homogeneous Coordinates

In three dimensions the position vector [x y z] is usually represented by the four-component homogeneous coordinates [x y z 1]. This row vector can be transformed by a general 4 x 4 matrix.

2.5.2 Transformations

Translation and scaling transformations can be easily generalized from two to three dimensions. The translation vector T consists of three components Tx, Ty and Tz. The 3D translation matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$$

Given the scaling vector S with scaling components Sx, Sy and Sz, the 3D scaling matrix is

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations in two-dimensions are assumed to be about the z-axis which is perpendicular to the x-y plane. With three dimensions, rotations can be about any one of the three axes in the plane formed by the other two axes. Positive rotation angles are measured in the clockwise sense when looking along an axis in the direction of the origin. Rotation about the z axis through an angle ϕ is achieved with the transformation:

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the y axis is given by:

$$\begin{vmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotation about the x axis is given by:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Permuting the axes in a cyclic fashion yields the expressions for rotation about the x and y axes from the z-axis rotation matrix.

2.5.3 View Transform

Objects are described in world coordinates. For 3D graphics these world coordinates usually must be converted to "eye coordinates" which have the location of the viewer's eye as the origin. A view transform matrix, composed of the concatenation of 5 simple transformation matrices, performs this change in coordinate systems. See procedure "view_transform_matrix" in Appendix Section A.3 "matrixops UNIT". Other details can be found in [NEWM79, Chapter 22] or [FOLE82, Chapter 8].

2.5.4 Projections

Perhaps the simplest projection from three- to two-dimensions is the "orthographic" projection. With this projection the z-coordinate is ignored and the (x,y) coordinates are used without modification. While this projection is simple it is not appropriate for many 3D applications.

A perspective view can be generated by projecting each point of an object onto the plane of the display screen. The view transform described in Section 2.5.3 is necessary to convert from world coordinates to eye coordinates.

The perspective transformation is different than previous transformations in that it requires dividing the the x and y values (in eye coordinates) by the z value (in eye coordinates). The earlier transformations only involved multiplications and additions. Generating a true perspective image requires dividing by the depth of each point.

2.5.5 Clipping

Clipping of three-dimensional segments can occur after the conversion from world coordinates to eye coordinates but it must be performed before the division by the depth of each point described in the previous section. The clipping operation simply cannot operate on projected line segments. Objects behind the eye can be projected onto the screen. The viewing "window" of 2D clipping is replaced by a "pyramid" in 3D clipping. The algorithm described in 2.3.3 can be used except line intersections with lines are replaced by line intersections with planes. Once 3D segments are clipped and projected they can be displayed.

3. Examples and Discussion

The microcomputer graphics package produced Exhibits 1 through 5. (The programs which produced these Exhibits are in Appendix B.) Most of these programs were originally developed as test cases while developing the various UNITs and PROCEDUREs.

3.1 Two-Dimensional Graphics

3.1.1 Cartography: Map of Kansas

Drawing straight lines is the simplest task performed using 2D graphics. Internally several special cases exist for drawing lines: single point, horizontal, vertical, slope >= 1, slope < 1. The points

for drawing the outline of the State of Kansas by county were obtained several years ago from a U.S. Department of Transportation tape. Since the map involves over 600 segments it was chosen for an initial thorough test of drawing simple lines of various slopes. Errors could be easily spotted. Exhibit 1.1 shows the whole state by county.

The plotting of straight lines was a major milestone in the program development -- further development obviously would have been senseless if simple line segments could not be handled. Memory was still available for user programs after they referenced the bulk of the library procedures. Processing time was slow, but tolerable. A large number of memory frames was available for the paging activity. (About 50 frames were initially available. Future additions have reduced that number to 10-20 with some programs now). All I/O and paging problems were ironed out in this first stage. Problems in mapping pixels to bits were also resolved.

Most of the vector and matrix mathematical operations were implemented in anticipation of 2D transformations. The 2D scaling, rotation and translation matrix operations were implemented and tested using the Kansas map data. Exhibit 1.1 shows scaling and translation while Exhibit 1.3 shows these operations as well as rotation about the "z" axis.

Initially, clipping was performed very crudely to avoid potential subscript range problems. Each pixel was checked to see if it was outside the "view" area. This crude clipping was initially implemented by a single IF statement but was very inefficient. 2D clipping was later "correctly" implemented using the Cohen-Sutherland algorithm from [NEWM79].

Exhibit 1.3 demonstrates intentional distortion to show what could happen with an inappropriate transformation matrix. For "fun" the black and white colors were inverted in this Exhibit.

3.1.2 Orthographic View of a Football Field

Exhibit 2 shows a 2D representation of the KSU football field -- an orthographic 3D view. This example was prepared to test 3D transformations and clipping.

3.2 Three-Dimensional Graphics

Once 2D graphic primitives were successfully implemented, work was begun on 3D graphics. When compiler symbol table space became more and more scarce it became apparent that many changes were necessary. Two- and three-dimensional primitives were combined into single PROCEDUREs. Combining the clipping operations was the most difficult change.

3.2.1 Perspective Views of a Football Field

Exhibits 3.1 through 3.4 show various perspective views of the KSU football field. Choice of a football field was made while attending a game last fall. A change in seats from the 50-yard line at one game to the endzone at another game gave me the idea. The field is useful for demonstrating perspective transformations since most people can visualize being in different seats and having the various views. These three pages of Exhibits take about 6.5 hours to plot from beginning to end. If processing time were faster, a movie composed of many still frames showing the aerial view flying around the stadium would have been considered.

3.2.2 Surface Graphics: z = f(x,y)

One of the original reasons for developing this graphics package was to plot perspective views of surfaces similar to the Surface II Graphics

System [SAMP75]. Exhibits 4.1 through 4.4 show various perspective views of the same surface but without hidden lines removed. One method of removing hidden lines for such surfaces of functional form is given in reference [KUBE68]. To date this hidden line removal algorithm has only been partially implemented.

3.2.3 Surface Graphics: Pressure Map of Well Field

Exhibits 5.1 and 5.2 show a practical application for surface graphics. This surface described by a 16-by-16 grid is the solution of 256 simultaneous equations solved iteratively as the numerical solution of a partial differential equation.