

AN ASCII TO HPGL CONVERSION SYSTEM

BY

SEJIN CHOI

Bachelor of literature

Kawndong University


Kangnung, Korea

1994

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1997

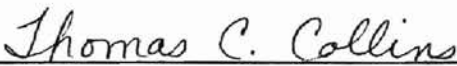
AN ASCII TO HGPL CONVERSION SYSTEM

Thesis Approved:



Thesis Adviser


H. Lu



Dean of the Graduate College

ACKNOWLEDGMENT

I wish to express my gratitude to Dr. K. M. George, my advisor, not only for his guidance and the time spent on my thesis but also for his friendly encouragement and inspiration. I would like to extend my gratitude to other committee members Dr. G. E. Hedrick and Dr. H. Lu for their invaluable advice and encouragement.

My sincere appreciation go to all my family, but especially my father, Chang-sun Choi, for his support and encouragement. My special thanks go to my sister and brother-in-law, Sunny Choi and J. P. Chandler for their support and assistance throughout my study.

I would also like to give my heartfelt thanks to my wife, Suyeon Kim, for her love and encouragement throughout my study. And finally, I would like to give my thanks to my friend, Nak-won Chu, for invaluable advice from his experience.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.	1
Graphical data type.	2
Raster data	3
Geometric data set.	3
Latent image data set	3
Vector format	4
Contour diagram.	5
II. RELATED WORK.	8
Graphical User Interface (GUI)	8
Traditional programming and Event-Driven programming.	8
Comparison of bitmap with vector formats.	9
Computer-Aided Design (CAD).	10
III. HEWLETT-PACKARD GRAPHICS LANGUAGE (HPGL)	12
IV. DATA CONVERSION FROM ASCII TO HPGL	17
Scaling.	18
Polygon mode	19
Labeling	20
Conversion algorithm	23
Graphical User Interface (GUI)	25
V. PERFORMANCE COMPARISON WITH BITMAP	27
VI. CONCLUSION	31
 BIBLIOGRAPHY	 32
APPENDICES	
APPENDIX A - GEOLOGICAL DATA USED IN FIGURE 5.1 - 5.2	35
APPENDIX B - SAMPLE DATA USED IN FIGURE 5.3	38

LIST OF TABLES

Table	Page
3.1 HPGL commands to draw primitives	13
3.2 HPGL commands to set attributes.	14
3.3 HPGL commands to display text.	14
3.4 HPGL commands to control coordinate system.	15
3.5 HPGL commands to control devices	16
5.1 Data storage size in bytes	29

LIST OF FIGURES

FIGURE	Page
1.1 Various types of graphical data.	4
1.2 Example of a contour map	7
4 VFG system architecture.	17
4.1 Isotropic scaling.	19
4.2 Manipulating a polygon	21
4.3 Label origin	22
4.5 GUI in VFG	26
5.1 A bitmap graph	28
5.2 A HPGL graph with the same data as Figure 5.1	28
5.3 Edge/filling method in HPGL.	30

LIST OF ALGORITHMS

Algorithm	Page
ASCII_to_vector_conversion.	23
Line_and_polygon_drawing.	24

CHAPTER I

INTRODUCTION

Computer graphics is a computer generated visual output display technology to create, manipulate, and display pictures. It improves the communications between human beings and computers [12].

In the past, computer output was considered as a tabulated list of numbers and some simplified strings to indicate the meaning of the list. Computer users can gather more information quickly from an output with graphical representation than users can from the text based output and the tabulated list of numbers over the same length of time by simplifying complicated output. This is one of the main reasons computer graphics is considered an important part in modern computer systems - faster communication and greater understanding [12].

Computer-aided design (CAD) has become one of the most popular applications of computer graphics. CAD systems help users to do rapid object design using simple methods. These objects could be buildings, mechanical systems, floor plans, electronic circuit boards, or engineering drawings,

in both two and three dimensions [4]. Nowadays, low-cost CAD systems are available for most personal computers.

There are several data exchange formats for non-CAD applications to gain access to engineering drawings, such as Drawing Exchange Format (DXF) or Hewlett-Packard Graphics Language (HPGL). HPGL especially has become the de facto standard plotter control language because of its wide acceptance. It is widely implemented both on pen plotters and laser printers [7].

In geology or earth science, computer graphics occupies an important part in representing data on a geological surface with a multilevel contour map. Using mathematical functions and statistical methods, the computer represents the contoured surface that gives the best fit to the data to illustrate spatial variability[24].

The goal of this study is to develop a general purpose package to convert general ASCII input data into vector format, HPGL, so that it can interchange data with CAD systems and draw a multilevel contour map.

Several concepts and terms used in this thesis are defined in sections 1.1 - 1.2.

1.1 Graphical data type

As graphical technologies are developing, it has become very important to save pictures to display or manipulate later. Due to the absence of one apparent

standard file format, many data formats have been invented for specific application programs according to each program's own characteristics [7]. These formats can be categorized as raster data, geometric data and latent image data formats.

a. Raster data

A raster data set consists of a set of discrete samples from within some space, either two dimensional (2D raster data set) - digital image refers to 2D raster data set which contains color samples over a grid - or three dimensional (3D raster data set, also called volumetric data set)[5].

b. Geometric data set

A geometric data set consists of a geometrical representation of objects in N-dimensional space where N is a non-negative integer. For example, a line specified by its two endpoints in Cartesian coordinates can be drawn by plotting all the points between the two endpoints. This generating process is called *rasterization* or *rendering* [5].

c. Latent image data set

A latent image data set consists of data which are not intended for graphical output but from which pictures are produced, such as numbers in a computer spreadsheet to be

used for graphical charts. The word latent implies lying hidden and undeveloped. In the broad sense, all data formats are latent image data sets [5]. Figure 1.1 illustrates these relationships.

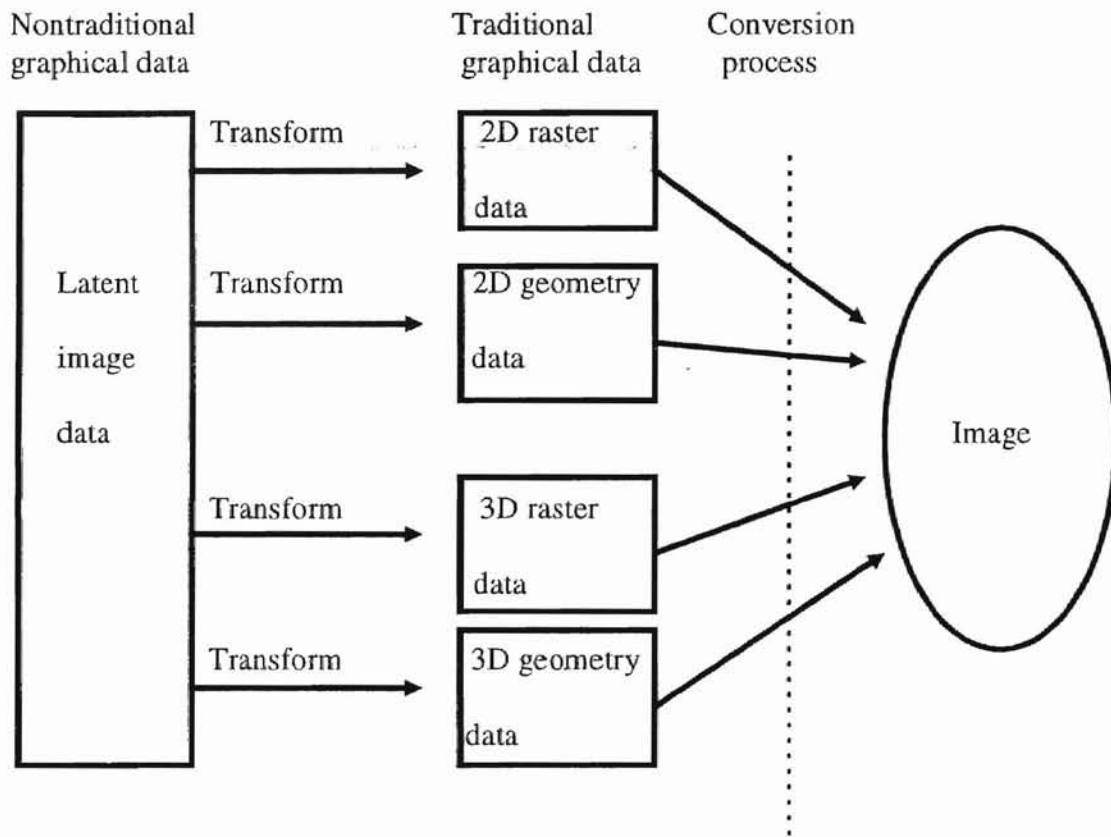


Figure 1.1 Various types of graphical data.

d. Vector format

Vector formats describe an image as a series of lines or shapes. In general, a vector format is used for line art, such as computer-aided design (CAD) drawings. Vector representation has more limitations than bitmap representation in terms of what it can do, but it can be more efficient and flexible for many applications. For example, a line can be specified by its two endpoints, and a circle can be specified by a radius and the coordinates of the center [7]. Drawing Interchange Format (DXF) and Hewlett-Packard Graphics Language (HPGL) are two examples of vector file formats [7].

1.2 Contour diagram

Storing and displaying geometrical data for geographic information systems (GIS) have progressed for the last 20 years [2]. A geometrical data set can be displayed in terms of points, lines and polygons on a map. A map is a plane parallel to the X and Y axis. So map plane is perpendicular to the vertical Z-axis. That is each point in geological data is located by horizontal X, Y and a vertical Z coordinates where the Z coordinate is normal to the map plane [20]. The scale of a map indicates the relationship between two end points on the map [23].

A contour map is also called a topographic line map, and it represents an object's surface with a vector representation [22]. Figure 1.2 shows an example of a

contour map [23]. The popularity of the contour map shows that it is an efficient way to give geometrical information to users [13]. The complicated construction of a contour map can be performed by a computer more efficiently and faster than a human. One of the most widely used examples is the tract evaluation model used in the U.S. Geological Survey to fix the fair market value of offshore leases. Another example would be the program used by many oil companies for prospect evaluation [6].

In this study, we develop a Vector File Generator (VFG) which converts ASCII format data to vector file format HPGL. This program reads ASCII format data and display a bitmap graph, and saves the information as both bitmap and HPGL file format according to the user's selection. This program is written in Microsoft Visual Basic 4.0 to provide a Graphical User Interface.

The remainder of this thesis is organized as follows: We first discuss related works such as Graphical User Interface (GUI), Event-Driven programming, and Computer-Aided Design (CAD). The subsequent chapter (chapter III) reviews Hewlett-Packard Graphics Language (HPGL). Conversion method from ASCII file format to Vector file format and user interface are described in chapter IV. Performance comparison of HPGL with bitmap is given in chapter V. This thesis ends with conclusion in chapter VI.

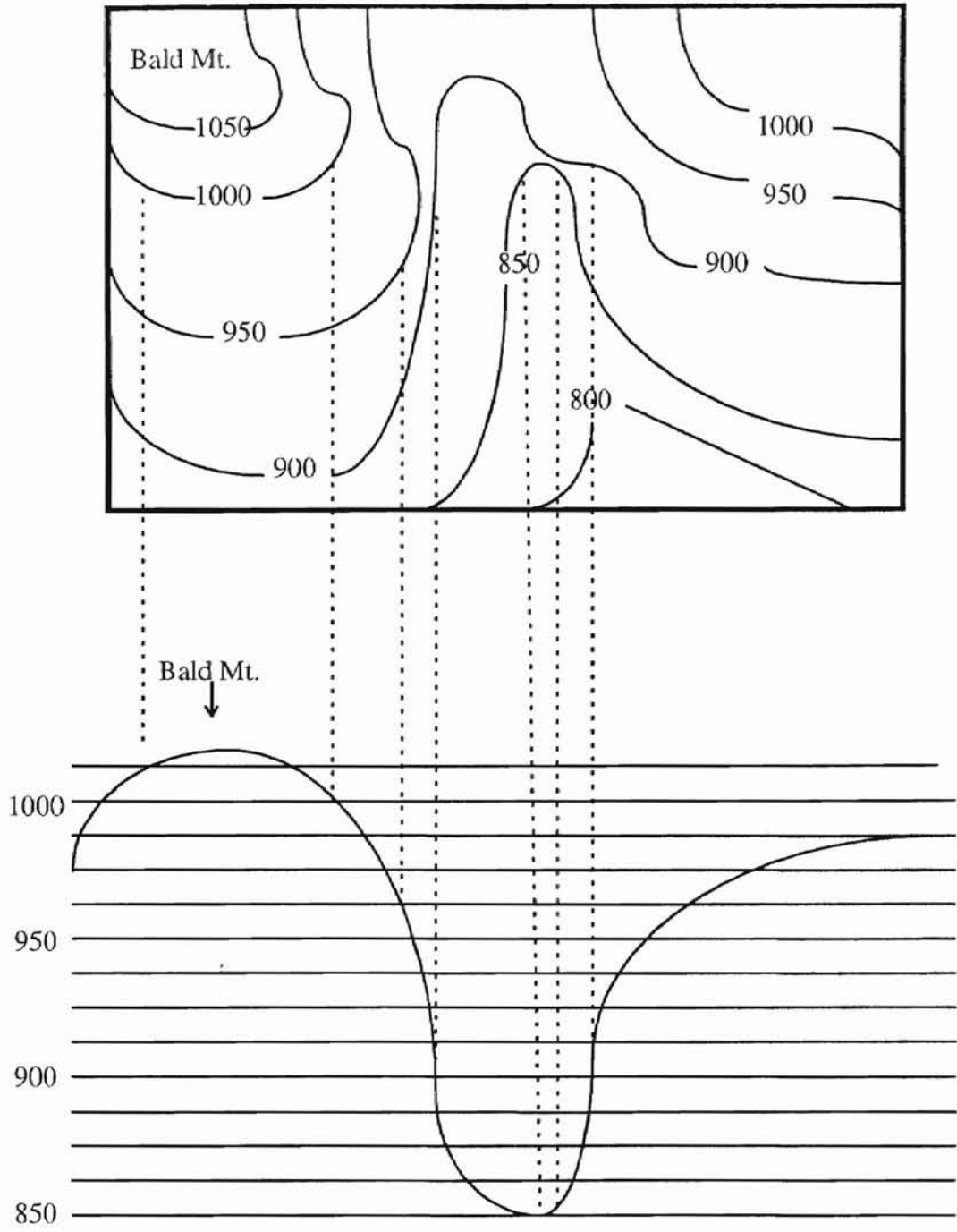


Figure 1.2. Example of a contour map

CHAPTER II

Related work

2.1 Graphical User Interface (GUI)

A user interface is a means of communication between an application and the users. The user interface determines the effective user acceptance of an application [1]. A GUI is a user interface that is well organized and visually represented, such as Windows, OS/2 Presentation Manager, Motif, and Open Look . A GUI helps people save time and effort by having similar type of interface organization and visual elements for several applications[21].

A GUI can also reduce the user's information load by displaying selective menus to let users focus on what they want, and organizing information in a meaningful way to make the decision-making process easier [1].

2.2 Traditional programming and Event-Driven programming

In traditional linear programming, the program itself, rather than an event, directly controls the execution

process [14]. The problem with this approach is that it grows exponentially in size when its complexity grows linearly. The biggest reason for this exponential growth is that the program must manage the interaction with users in various ways as well as the task it was designed for. This interaction management includes managing windows, keystrokes, mouse movements, window placement and drawing [11].

In an event-driven program, a system event triggers the execution of corresponding event procedure [14]. Thus events control program execution. The Event Manager handles each event such as a mouse movement and a key stroke. An event-driven program is waiting for an event to occur at any time, while a traditional linear program is not. This makes it much easier to develop an interface application with event-driven methods [11].

2.3 Comparison of bitmap with vector format

In the raster format, each pixel position is recorded in an array of intensity values[8]. Since a bitmap format can directly record any complicated image by breaking it up into a grid, it is widely used to simplify programming. However, a bitmap image has several disadvantages such as large image size and restricted flexibility.

The image size could be very large, even several megabytes, if the image uses high-resolution color.

Compression is used to manage size [19]. Therefore the compression method for storing bitmap data and transferring bitmap data over a network is one of the big issues in using a bit mapping method. However, since that issue is not part of this study, that problem will not be discussed in detail.

Flexibility is another problem with bitmap presentation. For example, if a part of a bitmap file is enlarged, one raster pixel may be duplicated in a larger area, possibly in rectangular shape, creating a problem known as *aliasing* or *staircasing* [7].

A vector format has more flexibility. For example, a line segment can be specified by only its two endpoints. It is easier to manipulate vector format than bitmap. For example, recoloring part of the picture or an object, such as a line or a polygon is easy to do in vector format as compared to bitmap format. However the vector format has more limitations in describing complicated graphics such as photographs than bitmap has.

For those reasons, a vector format is usually used for line drawing or designing a building, or circuit, etc., while a bitmap format is used for drawing complicated objects and various pictures, such as photographs.

2.4 Computer-Aided Design (CAD)

The CAD system was developed to help human interact with computers in designing and displaying objects. The general CAD system assists people for mechanical engineering design, building design, structural engineering design, electronic circuit design, animation and graphic design [3]. Drawing is an important tool in these areas. A freehand drawing could cause a misunderstanding between an engineer and a draftsman, and additional time to clarify the misunderstanding could be required [16]. One of the greatest advantages of a CAD system is that time can be dramatically saved at the modification stage, such as erasing and copying parts of the graphics [17]. A CAD system consists of three major parts[10] :

- 1) hardware - computer terminal, I/O devices, etc. -
- 2) software, and
- 3) designer.

AutoCAD is the most widely used CAD application program. AutoCAD drawings are used for mechanical engineering, software engineering, technical illustrating, architectural design and graphic design [18]. Vector format is used in AutoCAD.

CHAPTER III

Hewlett-Packard Graphics Language (HPGL)

HPGL is the command language of Hewlett-Packard pen plotters [7]. It was developed to send graphical commands to plotters and other graphical output devices [5]. Originally, it was implemented in 1976 for Hewlett-Packard's 7400 series pen plotters. HPGL has the advantages that it is easy to produce, independent of paper size, and widely used.

HPGL has two modes of drawing. They are *pen-up* mode and *pen-down* mode. In the *pen-up* mode, the pen moves but does not draw anything. In the *pen-down* mode, the pen draws actual line segments along its path. HPGL draws characters using simple stroke fonts, so they can be scaled, rotated, and slanted. In this chapter a list of HPGL commands and their meanings are provided.

All HPGL commands start with two uppercase letters which are acronyms of the commands, and these are followed by some number of arguments separated by spaces or commas. Commands are terminated by a semicolon.

For example:

```
PA 100, 3000;
```

is a command to move the pen to position (100,3000). PA stands for Plot Absolute. The PA command moves the pen to the given point. Drawing the line depends on whether the mode is *pen-up* or *pen-down*.

Most of the basic HPGL commands are given in Tables 3.1- 3.5. Complete references can be found in [5][7].

Table 3.1 HPGL commands to draw primitives

Instruction	Function
AA (Arc Absolute) X, Y, Angle	Draw an arc of the circle centered at (X, Y)
AR (Arc Relative) X, Y, Angle	Draw an arc of the circle centered at distance (X, Y) from the current position.
CI (Circle) Radius (, Angle)	Draw a circle whose radius is given.
EA (Edge rectangle Absolute) X, Y	Outline the rectangle defined by (X, Y) and the current position.
EP (Edge polygon)	Outline the current polygon.
EW (Edge wedge) Radius, Start, Sweep	Outline a pie-shaped wedge.
FP (Fill Polygon)	Fill the current polygon.
PA (Plot Absolute) X1, Y1 (, X2, Y2 ...)	Move pen to (X1, Y1) and so forth. If the pen is down, draw a line.
PD (Pen down) X1, Y1 (, X2, Y2 ...)	Pen down on the paper, and move the pen to (X1, Y1) and so forth.
PR (Plot Relative) X1, Y1 (, X2, Y2 ...)	Plot relative from current position.
PU (Pen Up) X1, Y1 (, X2, Y2 ...)	Pen up
RA (Rectangle Absolute) X, Y	Fill the rectangle defined by current position and (X, Y).
RR (Rectangle Relative) X, Y	Fill the rectangle defined by the current position and the current position plus (X, Y).
XT	Draw ticks on X-axis.
YT	Draw ticks on Y-axis.

Table 3.2 HPGL commands to display text

Instruction	Function
CA set	Designate character set.
CP (Character Position) X, Y	Move to the point (X, Y) character spaces from the current position.
CS set	Designate alternate character set.
DI (Absolute Text Direction) X,Y	Set absolute label direction.
DR (Relative Text Direction) X,Y	Set relative label direction relative to the current position.
DT (Define Terminator) C	Set the new label terminator to the character immediately after the command DT.
ES (Extra Space) W, H	Leave extra space of W units horizontally and H units vertically between characters.
LB (Label) String	Draw the string at the current position.
LO PositionNo	Set label origin.
SA (Alternate Font)	Select alternate character set.
SI (Absolute Character Size) X,Y	Set the size of characters displayed in centimeters.
SL (Slant) Tan	Set the slant of characters displayed.
SM (Symbol Mode) Character	Draw the given character at the end of each line. This command doesn't depend on PU or PD.
SR (Relative Character Size) X,Y	Set the size of characters displayed as a percent of the page size.
SS (Standard Font)	Switch between two internal plotter fonts.

Table 3.3 HPGL commands to set attributes

Instruction	Function
FT (Fill Type) Type, A, B	Set the fill type. If the fill type is 1 or 2, then fill type is solid color, and A and B are ignored. If the fill type is 3, then the fill type is hatched. If the fill type is 4, then the fill type is crosshatched. In the cases of fill types 3 and 4, A is the spacing of the lines and B is the angle of the lines in degrees.
LT (Line Type) Type	Set the type of line drawn. The default type is a solid line.
PC	Pen color
PM (Polygon Mode) Flag	If the Flag is 0, then start a new polygon. If the Flag is 1, then close the current polygon by adding an edge to the first vertex, and start new polygon. If the Flag is 2, then close the current polygon.
SG (Select Group) GroupNo	Select pen group.
SP (Select Pen) PenNo	Select the given pen for subsequent drawing.
UF type1(..., type20)	User-defined fill type.

Table 3.4 HPGL commands to control coordinate system

Instruction	Function
IP (Input Points) X1, Y1, X2, Y2	Change the scaling points.
IW (Input Window) X0, Y0, X1, Y1	Set clipping limits to the rectangle defined by (X0, Y0) and (X1, Y1).
RO (Rotate) Angle	Rotate the coordinate system by the given number of degrees counterclockwise.
SC (Scale) X0, X1, Y0, Y1	Set the scale of the plot. The lower left corner is set to (X0, Y0), and the upper right corner is set to (X1, Y1).

Table 3.5 HPGL commands to control devices

Instruction	Function
AF	Advance full page.
AH	Advance half page.
DF (Defaults)	Set default values.
EC	Enable cutter.
FR	Advance frame.
GC Counter	Set count number.
IN (Initialize Plotter)	Set all parameters to defaults. Move the pen to the origin.
NR	Not ready.
PG (Page)	Advance page.
PS Size	Set paper size.
PT thickness	Set pen thickness.
VS Velocity	Set pen's velocity.

CHAPTER IV

Data Conversion from ASCII to HPGL

In this chapter we describe the software developed in this research project. It is called Vector File Generator (VFG). VFG reads graphics data from an ASCII file, and converts it to HPGL file format. The general architecture of the VFG is as shown in Figure 4. As can be seen from figure 4, it implements distinct functions. They are file I/O, conversion, and user interface. The various problems addressed in conversion are discussed in the subsections that follow.

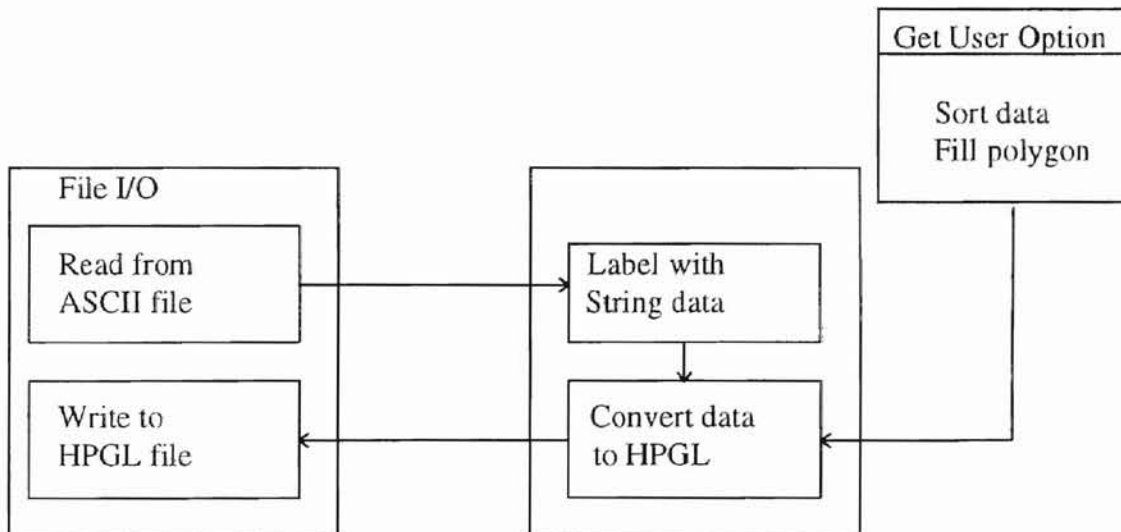


Figure 4. VFG system architecture

4.1 Scaling

To convert ASCII data to HPGL, the first step should be scaling a plot to display it properly on the paper. In scaling, the Hard-clip limit and the number of units along each axis should be specified. Hard-clip limit is a physical boundary for pen movement. This limit should be large enough to enclose the actual graph size specified by Scaling methods.

The Scale instruction in HPGL has three different modes which are Anisotropic, Isotropic, and Point-factor scaling modes [9]. In Anisotropic scaling mode, a user can use a different unit along the X-axis than the unit along the Y-axis. However, in this mode graph could be distorted. For example, a circle could look like an ellipse. On the other hand, Isotropic scaling mode allows the user to use the same size on both the X- and Y-axes as illustrated in Figure 4.1. For this reason, the Isotropic scaling method is used for the VFG. The third mode, Point-factor scaling mode, uses a specific ratio of plotter units and establishes the user-unit coordinates of the first point defined. For example, to scale in millimeters, the instruction should be (SC0, 40, 0, 40, 2) because 1 millimeter is 40 plotter units. Therefore each user unit will be 1 millimeter [9].

4.2 Polygon mode

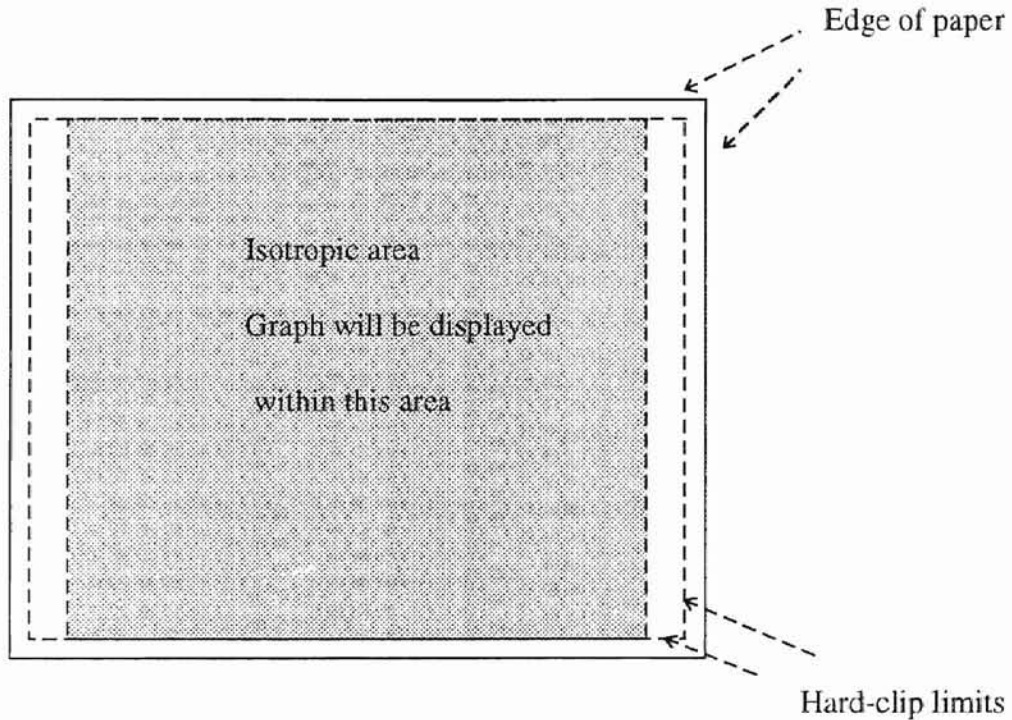


Figure 4.1 Isotropic scaling

To define each line graph in ASCII format as a polygon, the VFG puts each vertex of the graph into a polygon buffer. The polygon buffer is a memory space which keeps all coordinates and instructions of defined polygon. This polygon stays in the polygon buffer until exiting polygon mode or until another polygon replaces it. Polygon mode has three different states. They are [9]:

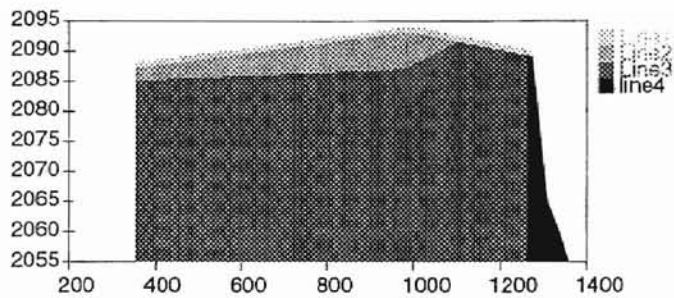
- Polygon mode 0 : Initialize the polygon buffer and enter polygon mode.
- Polygon mode 1 : Close the current polygon or sub-polygon and stay in polygon mode.
- Polygon mode 2 : Close the current polygon or sub-polygon and exit polygon mode.

To fill a polygon, the FP command is used. The FP command uses the current pen and fill type. The VFG uses the 'odd-even' filling algorithm. With the odd-even filling algorithm, FP fills an area if a line crossing the polygon intersects the polygon an odd number of times.

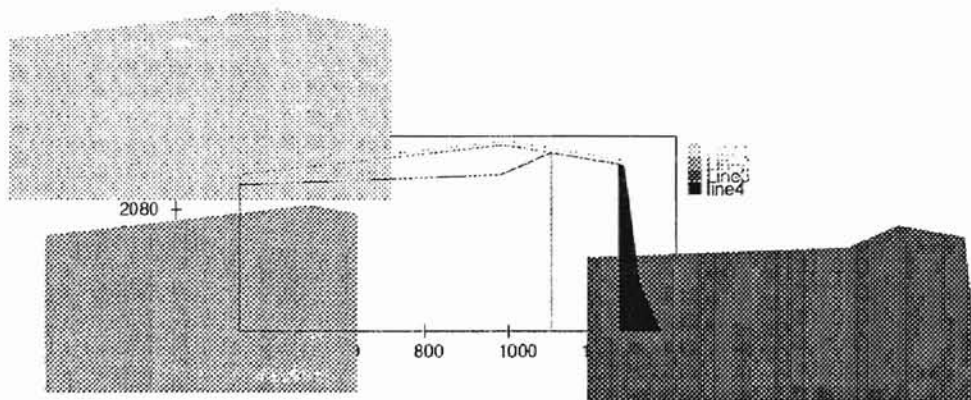
As Figure 4.2 shows, once the data is defined as a polygon by putting all vertices into polygon buffer, it can be easily modified. For example, Figure 4.2 (a) shows the original graph. Each polygon edge and filled area can be separately relocated easily just by dragging a mouse to the desired location as shown in 4.2 (b).

4.3 Labeling

String data from an ASCII file can be imported into HPGL by using the LB command. The LB command uses the currently defined font and position. Therefore, to write a label the pen should be moved into the desired position with *pen-up* mode. The LB command then automatically sets up *pen-down* mode to write the string. The label terminator should be placed



(a) Original graph



(b) After relocating filled area by dragging a mouse

Figure 4.2 Manipulating a polygon

right after the string to exit LB mode, otherwise the entire HPGL instruction will be considered as part of string data and displayed on the paper. The label

terminator can be defined using the DT command. The DT command has two different modes, which specify whether the label terminator is printed or not. The DT command remains in effect until the plotter is initialized or a new DT instruction replaces it.

After moving to the desired position, the program should set the correct starting position. The LO (Label Origin) command specifies the starting position relative to the current pen location. The LO command can specify 19 different positions. The positions specified by 1 to 9 are different from the positions specified by 11 to 19 only in the offset value of the label relative to the pen position. Each specific position is illustrated with an example in Figure 4.3[9]. Each figure used an example string 'LO' with the position number. For example, <LO11;> instruction places the left bottom of the first character of the label at the position $\frac{1}{4}$ of the character height above and $\frac{1}{4}$ of the character width to the right of the current pen position.

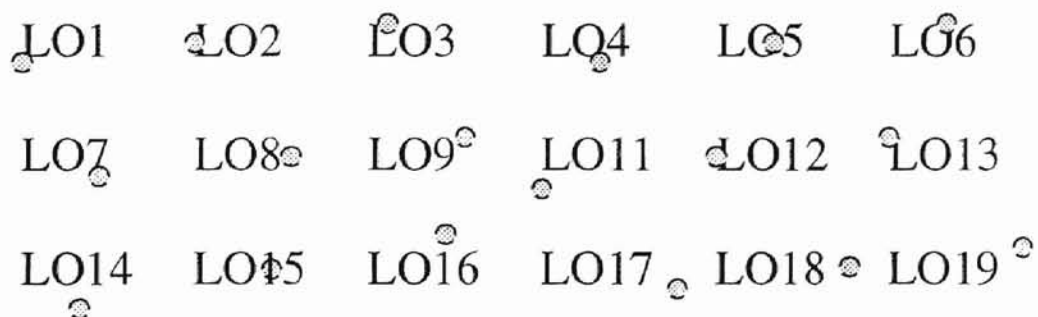


Figure 4.3 Label origin

4.4 Conversion algorithm

The VFG determines the maximum and minimum range of the ASCII input to set Hard-clip limits and scale the X- and Y-axes. After scaling properly, the program enters polygon mode and groups all vertices of a line graph together. Each vertex in the polygon buffer becomes a vertex of the polygon and these vertices can be manipulated as a group with the polygon group instructions such as FP, EP or FT. The ASCII file format used in this thesis is as follows:

```
Graph_Label           `Title
X_Label, Y_Label     `Label on X, Y axes
Line1_Label          `Label on the first data set
X1, Y1              `Point list
.
.
Xn, Yn
#                   `End of first set of data
.
.
Linen_Label
.
.
#
```

The algorithm used to convert ASCII data to HPGL is as follows:

Algorithm: ASCII_to_vector_conversion

```
Reset all plotter functions to default setting
Read ASCII data and find maximum, minimum vertices
Define the Hard-clip limit
Scale
Do While Not End-Of-File {
```

```

Do While Not End-of-line-data (
  Select pen number
  Clear the polygon buffer and enter polygon mode
  Put each vertex of the line into the polygon buffer
  Fill the polygon
  Edge the polygon
  Exit polygon mode
  Label the polygon
)
}

```

To convert the ASCII format data to HPGL format, the VFG first should determine if the data is polygon data or line data. If the data is a polygon data, VFG clears and opens the polygon buffer and puts each vertex into it.

Otherwise, according to user's selection, the VFG closes the line graph data by generating an additional <PDPA> instruction from the last point to the first point and defines the graph as a polygon. The user can choose either format by selecting the option from the menu bar. Detailed algorithm to convert each line graph data is as follows:

Algorithm: Line_and_polygon_drawing

```

For I = number-of-data to 1 step -1
  Select pen color
  If ( first coordinate = last coordinate) then
    PolyFlag = True
  End if
  If polyFlag = True then
    Open polygon mode
    Move the selected pen to the first coordinate
    Put the first coordinate into the polygon buffer
    Set up pen-down mode
    For J = 2 to NumberOfLineData
      move the pen from J-1 to J
      Put the coordinate J into polygon buffer
    Next J
    Close the polygon buffer
    Fill out the polygon buffer with the selected color
  End if

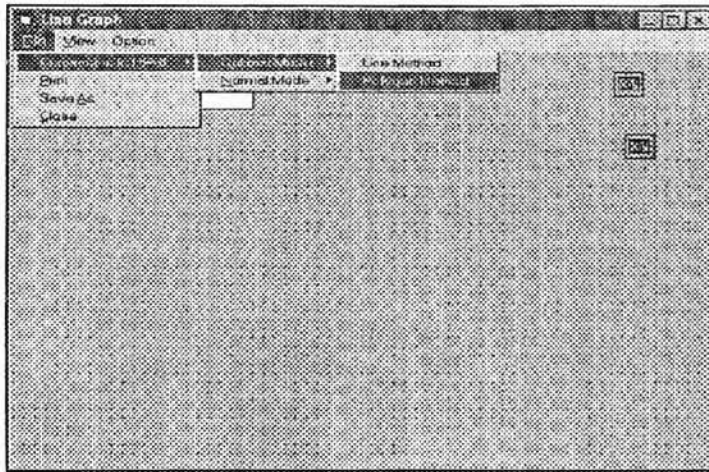
```

```
Else
  Move the selected pen to the first coordinate
  set up pen-down mode
  For J = 2 to NumberOfLineDate
    move the pen from coordinate J-1 to J
  Next J
End If
Next I
```

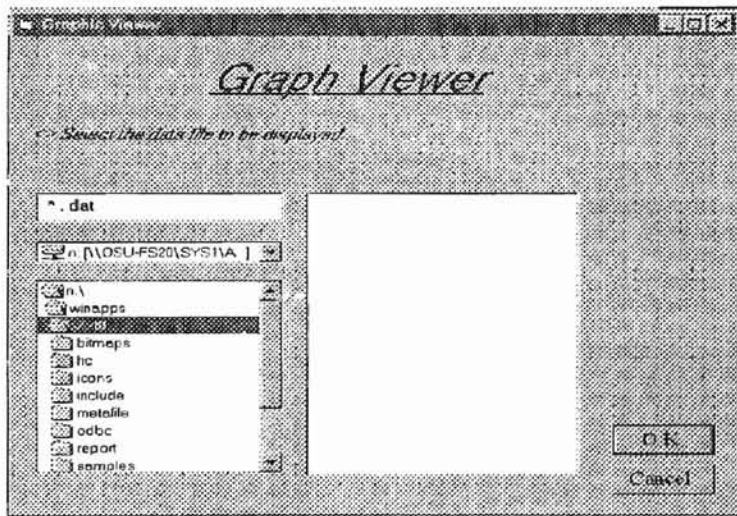
In the next chapter we analyze the effect of VFG.

4.5 Graphical User Interface (GUI)

One of the principal components of VFG is the graphical user interface. The GUI reduces the user's information load by presenting optional menu bar and file I/O dialog boxes and organizing information in a meaningful way to let users focus on essential task [1]. The VFG system is implemented using Visual Basic 4.0. Figure 4.5(a) and 4.5(b) show two examples of the GUI capabilities. In VFG, User's options such as sort and polygon mode setting can be easily accomplished via pop-up menu. These options can be changed by the user at any time during execution. This pop-up menu bar is shown in Figure 4.5(a). Users can also manipulate file I/O such as retrieving input data from the ASCII file, saving bitmap file and HPGL file by simply clicking mouse button on the Open/Save dialog boxes as shown in Figure 4.5(b).



(a) Pop-up menu



(b) Open/Save dialog box

Figure 4.5 GUI in VFG

CHAPTER V

PERFORMANCE COMPARISON WITH BITMAP

This chapter is devoted to comparison of raster representation and HPGL representation of graphs. In this comparison, a windows bitmap file format which has an extension <.bmp> file is used since the windows bitmap file can have the highest possible compatibility between different versions of Windows and hardware[15]. Two criteria which are the quality of the graph and the size of disk storage required to store the information on the graph are used for this comparison. Two case studies are used in this comparison.

Figure 5.1 is a bitmap graph. It takes 787Kbytes to save the information. The bitmap graph is shown as a HPGL graph in Figure 5.2. It uses the same data as Figure 5.1 and it reduces the disk storage required significantly. It takes only 3Kbytes to store the same information.

From Figure 5.2, we observe that the staircasing effect is eliminated when HPGL representation is used.

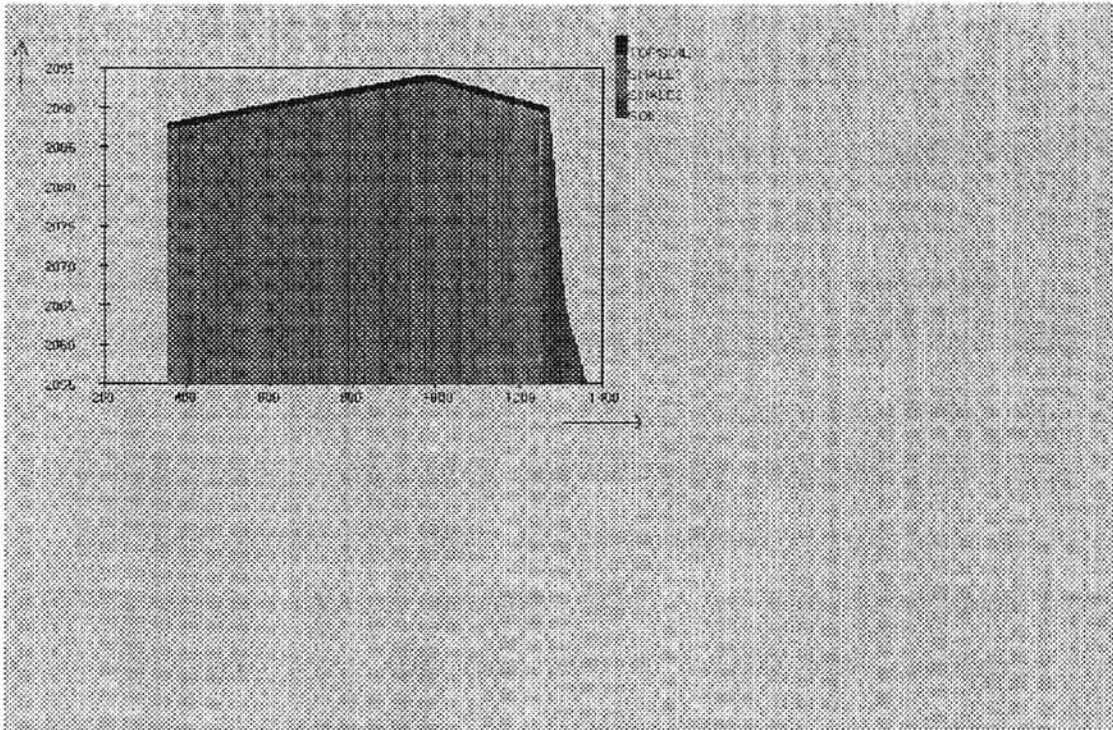


Figure 5.1 A bitmap graph

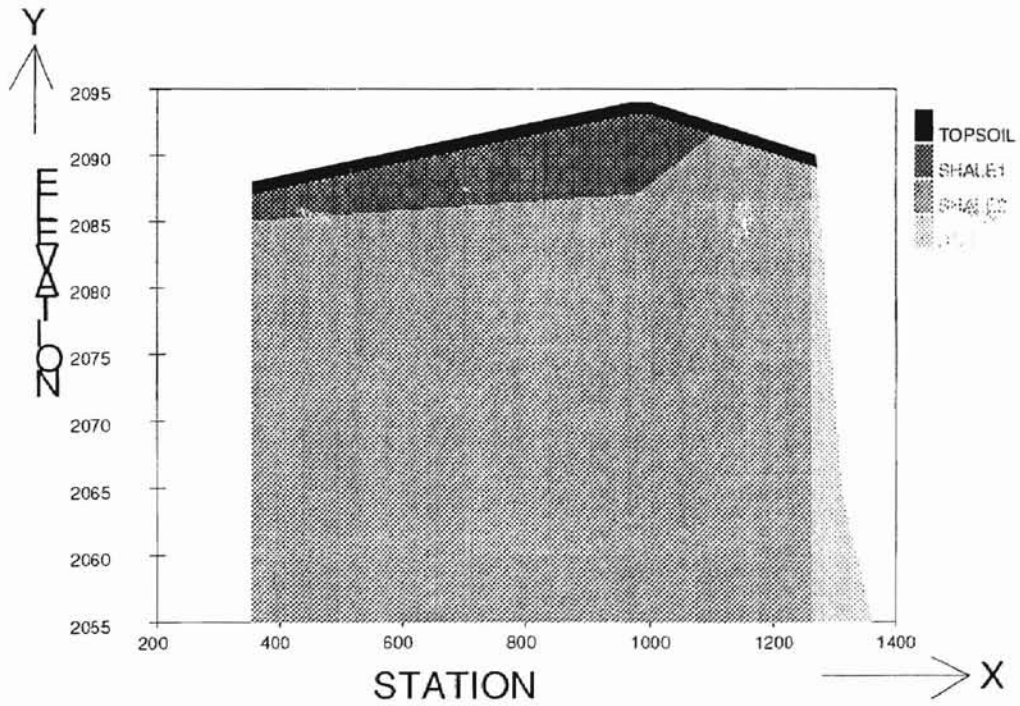


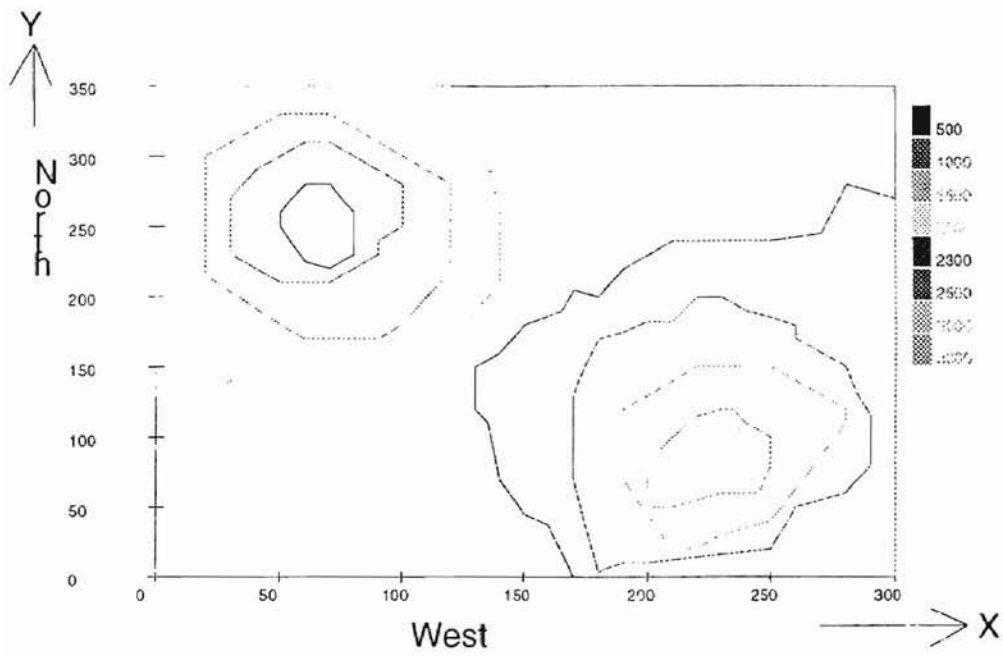
Figure 5.2 A HPGL graph with the same data as Figure 5.1

Figure 5.3 shows a graph in both edge and filling modes of polygon mode. Edge mode is used to outline the polygon, and filling method in polygon mode is used to fill a color in the polygon [9]. In HPGL, filling colors inside the polygon can be accomplished with one 'FP' command without having to program any filling algorithm. In edge mode, this HPGL format takes even less disk storage. In Figure 5.3, edge mode saves 2Kbytes more than polygon mode.

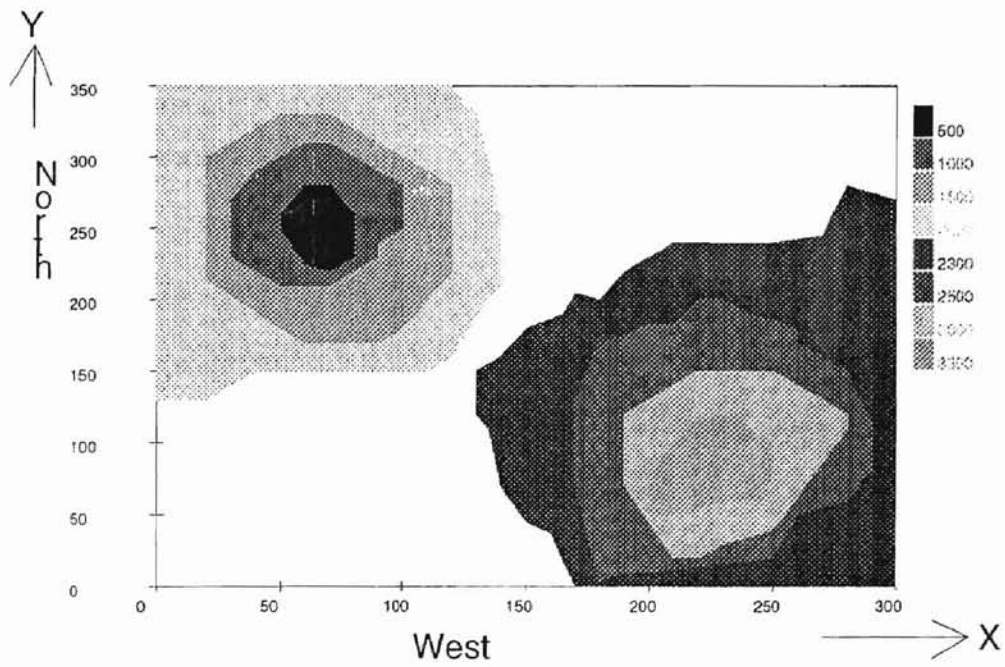
Precise amount of data storage required on each mode is as shown in table 5.1. Data set A is the data used for figures 5.1 and 5.2, and data set B is the one used for figure 5.3. Data used in these experiments are given in appendix A and appendix B.

	Bitmap	HPGL with Edge mode	HPGL with filling mode
Data set A	786,566 (100%)	2,752 (0.35%)	3,001 (0.38%)
Data set B	786,566 (100%)	4,173 (0.53%)	4,301 (0.55%)

Table 5.1. Data storage size in bytes



(a)



(b)

Figure 5.3 Edge/filling method in HPGL

CHAPTER VI

CONCLUSION

In this thesis, we reviewed bitmap and vector format file formats. Each file format has its own advantages over the other one depending on a given situation. A software package, VFG is designed and implemented to convert statistical or geological data to vector file format HPGL.

VFG is implemented with Microsoft Visual Basic 4.0 to provide an efficient Graphical User Interface (GUI). Experiments were conducted with several data files. Geological data files were used to generate graphic objects in bitmap format and HPGL format. The results of the experiments show that HPGL can save data storage significantly and eliminate staircasing which appears in bitmap file. Another advantage is ease of manipulation. For example, it is easier to move and fill polygons in HPGL files than in bitmap files.

VFG reads ASCII format data and converts it to HPGL file. A drawback of the VFG is that the program is currently limited to two dimensional data. Future work will be directed towards an implementation of conversion algorithm for three or higher dimensional data.

BIBLIOGRAPHY

- [1] Aaron Marcus, Nick Smilonich and Lynne Thompson, The Cross-GUI Handbook For Multiplatform User Interface Design, Mass : Addison-Wesley Publishing Co., 1995.
- [2] Andrew U. Frank. "Spatial concepts, Geometric data models, and Geometric data structures", Computers & Geosciences Vol. 18, No. 4, pp. 409-417, 1992
- [3] C. B. Besant, Computer-Aided Design and Manufacture, Second Edition, New York : Ellis Horwood Limited, 1983.
- [4] Cornel Pokorny, Computer Graphics, Oregon : Franklin, Beedle & Associates Incorporated, 1994.
- [5] C. Wayne Brown and Barry J. Shepherd, Graphics File Formats, Greenwich: Manning Publication Co., 1995.
- [6] Daniel F. Merriam, Ed., Computer Application in the Earth Sciences, New York : Plenum Press, 1981.
- [7] David C. Kay and John R. Levine, Graphics File Formats, Second Edition, Windcrest, 1995.
- [8] Donald Hearn and M. Pauline Baker, Computer Graphics, Englewood Cliffs: Prentice Hall, 1986.
- [9] Hewlett Packard, The HP-GL/2 and HP RTL Reference Guide, CA : Addison-Wesley Publishing Co., 1995.

- [10] James H. Earle, Engineering Design Graphics , Fifth Edition, Addison-Wesley Publishing Co., 1987.
- [11] John R. Levine, Ed., Designing GUI Applications For Windows, New York : M&T Books, 1994.
- [12] John R. Rankin, Computer Graphics Software Construction, Australia : Prentice Hall, 1989.
- [13] Michael F. Goodchild. "Geographical Data Modeling", Computers & Geosciences Vol. 18. No. 4, pp.401-408, 1992.
- [14] Microsoft Corporation, Microsoft Visual Basic Programmer's Guide, Washington : Microsoft Corp, 1992.
- [15] Microsoft Corporation, Microsoft Windows User's Guide, Washington : Microsoft Corp, 1991.
- [16] Myron G. Mochel, Fundamentals of Engineering Graphics, Englewood Cliffs : Prentice Hall, 1960.
- [17] Richard G. Budynas, Interactive Graphics In CAD, New York: UNIPUB, 1984.
- [18] Robert M. Thomas, AutoCAD Desktop Companion, CA: SYBEX, 1989.
- [19] Rodney Perkins. "File formats on the Internet", Computers & Geosciences Vol. 21, No. 6. pp. 775-777, 1995
- [20] Simon W. Houlding, 3D Geoscience Modeling, Hong Kong : South Sea Int, 1994.
- [21] Sujatha Samadarsini Neelam. "Design and implementation of an efficient index structure and a GUI for spatial

- databases using Visual C++", Unpublished Master's thesis, Oklahoma State University, 1996
- [22] T. Cronin. "Automated Reasoning with Contour Maps", Computers & Geosciences Vol. 21. No.5. 609-618, 1995
- [23] Terry R. West, Geology Applied to Engineering, Englewood Cliffs: Prentice Hall, 1995.
- [24] T. V. Loudon, Computer Methods In Geology, London: Academic Press, 1979.

APPENDIX A

GEOLOGICAL DATA USED IN FIGURE 5.1 - 5.2

USDA1.dat

STATION, ELEVATION

TOPSOIL

355,2088

970,2094

1000,2094

1270,2090

1272,2089

#

SHALE1

355,2087

970,2093

1000,2093

1101,2091.5

#

SHALE2

355,2085

980,2087

1101,2091.5

1262,2089.15

1300,2055

#

SOIL

1262,2089.15

1272,2089

1310,2065

1360,2055

#

APPENDIX B

SAMPLE DATA USED IN FIGURE 5.3

GeoTest.DAT

West, North

500

50,250

50,260

55,270

60,280

70,280

80,260

80,230

70,220

60,225

50,250

#

1000

30,230

30,270

40,290

50,300

60,310

70,310

80,300

90,290

100,280

100,250

90,240

90,230

80,220

70,210

50,210

30,230

#

1500

20,215

20,300

30,310

40,320

50,330

70,330

80,320

90,310

100,300

110,290

120,280

120,220

110,200

100,180

90,170

60,170

50,180

20,215

#

2000

0,130

0,350
120,350
130,330
140,260
140,210
130,190
120,160
110,150
40,150
30,140
20,130
0,130

2300
170,0
300,0
300,270
280,280
270,245
250,240
210,240
200,230
190,220
180,200
170,205
165,190
150,180

140,160

130,150

130,120

135,110

140,70

150,45

160,37

170,0

#

2500

180,3.5

170,70

170,130

172,140

180,170

190,175

200,182.5

210,182.6

220,200

230,200

240,190

250,185.3

260,179

260,170

280,150

285,130

290,115

290,80

280,60

270,55

260,50

250,20

200,10

190,10

180,3.5

#

3000

210,20

190,70

190,120

200,130

210,140

220,150.5

240,150.5

250,150

260,140

280,120

280,110

250,40

230,30

220,20

210,20

#

3300

200,50

200,80

210,100

220,115

230,120

235,120

240,110

250,100

250,80

245,60

230,60

220,55

210,50

200,50

#

2

VITA

Sejin Choi

Candidate for the Degree of
Master of Science

Thesis: AN ASCII TO HPGL CONVERSION SYSTEM

Major Field: Computer Science

Biographical:

Education: Graduated from Sun-In High School, Incheon, Korea in February 1987; received Bachelor of literature degree in Korean language and literature from Kawndong University, Kangnung, Korea in February 1994. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May 1997.

Experience: Employed by United States Department of Agriculture as a graduate research assistant, February 1997 to December 1997.