VECTOR-BASED COMPUTER GRAPHICS IN AUTOMATED MAP COMPILATION

C F Scheepers, Scientist
Centre for Advanced Computing and Decision Support
CSIR, P O Box 395, Pretoria 0001
Republic of South Africa

ABSTRACT

This paper describes a computer-assisted mechanism for constructing line and area symbols on maps. Vector-based computer graphics techniques such as hatching and clipping are extended for this purpose. In an effort to enhance visually the perception levels of line and area symbols, filling and following algorithms are used to place basic symbols on line and area symbols. Filling and following densities can be modified easily by changing input parameters, affording the cartographer freedom in symbol design and use.

INTRODUCTION

The application of computers and related technology to cartography has revolutionized the art and science of cartography. The digital geographic database has proved to be supreme as a storage medium for spatial information, resulting in the utilization of maps primarily for the communication of preselected spatial information. Moreover, substantial changes to the process of map compilation have also taken place. Through automation and the exploitation of computer graphics techniques, much of the tedium of manual map compilation has been relieved. Specifically, the cartographer could be assisted with the selection of cartographic features to be placed, with the design of a map symbolism, and with the creation and placement of additional features such as titles, legends, north indicators, grids, and feature name labels.

The function of maps in cartography is closely related to the general purpose of computer graphics. Whether two-dimensional or three-dimensional, or even vector or raster-based, the ultimate goal in computer graphics is to convey information graphically. Maps convey *spatial* information in a similar way. Spatial information pertaining to cartographic features of different dimensionality (point, line and area type features) is represented by means of symbols. These symbols should be designed not only to give an indication of the types of features they represent, but also to reflect characteristic properties of the features. In figure 1, examples of point, line and area symbols are illustrated. Note that *follow* and *fill* symbols are often included with line and area symbols to enhance the ability of the latter to convey information.

A SPECIFICATION FOR MAP SYMBOLISM

Before the computer-assisted mechanism for the construction of line and area symbols is presented, it is necessary to describe a method by which the map symbolism may be specified.

About line segments and pixels

The difference between vector-based and raster-based graphics is characterized by the primitive elements used for building pictures. In vector-based graphics the primitive element is the line segment, whereas the picture element (or pixel) is used in raster-based graphics.

Since point and line symbols (and the borders of area symbols) are more naturally described using a vector-based format, and since vector-to-raster conversion routines now frequently reside in hardware, the choice to describe symbols in a vector-based rather than a raster-based fashion is the least limiting. More specifically, raster technology could still be used even though symbols are described in a vector-based format.

A formal specification

The description of symbols is now formally specified in extended Backus-Naur form (Scheepers 1987b). Note that four conceptual primitives are defined to supply graphic building blocks at a somewhat higher level:

- *dots*: line segments with zero length;
- *segments*: visible or invisible line segments;
- *arcs*: a limited number of line segments approximating an arc;
- *text*: characters compiled from sets of line segments.

```
  Key :
             <...>    Concept
             [...]    Optional
             {...}+   Set of 1 or more
              ::=     Consists of / is defined as
               |      Choice
           terminals  Without brackets

<point symbol>  ::= <std_symbol> <placement>
<placement>     ::= centered | standing

<line symbol>   ::= <line> [<follow symbol>] | [<line>] <follow symbol>
<line>          ::= single <attribute> | double <attribute>
<follow symbol> ::= <std_symbol> <follow type> <orientation> <overlap>
<follow type>   ::= true | visual
<orientation>   ::= perpendicular | upright | parallel
<overlap>       ::= permitted | forbidden

<area symbol>   ::= <area> [<border>] | [<area>] <border>
<area>          ::= <fill symbol> | <hatched>
<fill symbol>   ::= <std_symbol> <clipping>
<hatched>       ::= hatch | crosshatch
<clipping>      ::= clip | don't clip
<border>        ::= <line symbol>

<std_symbol>      ::= {<picture element>}+
<picture element> ::= <primitive> <attribute>
<primitive>       ::= dot | segment | arc | text
<attribute>       ::= colour linetype linewidth textsize
```

Terminals are defined graphically in figure 2.

Parameters for the specification

Additional information would be needed during the actual construction of symbols. This information may be imported using the following parameters (see figure 3):

- *point symbols*:
  - o The orientation and relative size of point symbols.

- *line symbols*:
  - The distance between double lines.
  - The distance between individual follow symbols.
  - The size of follow symbols relative to basic symbols.
- *area symbols*:
  - The orientation of hatch lines and the distance between them.
  - The spacing parameters for regular fill patterns.
  - The orientation and relative size of fill symbols.

THE CONSTRUCTION OF LINE SYMBOLS

From the specification in the previous section, a line symbol may be defined in two ways. Firstly, available attributes may be used to construct single or double lines and, secondly, a set of follow symbols may be used to construct the line symbol. Note that usage of the one method does not preclude the use of the other.

If the second method is used, the follow type (*true* or *visual*) and the orientation of follow symbols (*perpendicular*, *upright* or *parallel*) should be considered. Furthermore, if both methods are used simultaneously, the overlap specification (*permitted* or *forbidden*) should also be catered for.

Single and double lines

The construction of single lines is trivial. Double lines, on the other hand, require careful consideration. Apart from using the available attributes to distinguish between different double line symbols, a distance parameter should also be incorporated. Suppose that the cartographic line feature to be represented is described by n line segments with coordinates $(x_0, y_0) \ldots (x_n, y_n)$ as illustrated in figure 4. To determine the coordinates $(x'_0, y'_0) \ldots (x'_n, y'_n)$ representing one line of the double line symbol, two cases are considered:
- Determining the start point (x'_0, y'_0) and end point (x'_n, y'_n).
- Determining (x'_i, y'_i) for $i = 1, 2, \ldots, n-1$, the rest of the coordinates.

To determine the coordinates representing the second line of the double line symbol, a similar approach is taken.

The start and end points. The coordinate of the start point (x'_0, y'_0) in figure 4 is given by the following formulas:

$$\begin{aligned} x'_0 &= d\cos\alpha & \qquad\qquad y'_0 &= b\sin\alpha \\ &= d\cos(\tfrac{\pi}{2} - \theta_1) & &= d\sin(\tfrac{\pi}{2} - \theta_1) \\ &= d\sin\theta_1 & \textit{and} \qquad &= d\cos\theta_1 \\ &= d(y_1 - y_0)/s & &= d(x_1 - x_0)/s \end{aligned}$$

where d is half the perpendicular distance between the double lines and

$$s = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}.$$

The end point (x'_n, y'_n) is determined in a similar fashion by replacing (x_0, y_0) and (x_1, y_1) with (x_{n-1}, y_{n-1}) and (x_n, y_n).

The other coordinates. To simplify the discussion which follows, assume that the coordinate (x_i, y_i) is placed at the origin of a Cartesian coordinate system. Then, the coordinate of (x'_i, y'_i) is easily determined by intersecting the two lines A $(y\cos\alpha_1 - x\sin\alpha_1 - d = 0)$ and B $(y\cos\alpha_2 - x\sin\alpha_2 - d = 0)$ in figure 4.

Let the distances from the origin to (x_{i+1}, y_{i+1}) and (x_{i-1}, y_{i-1}) be $s_1 = \sqrt{x_{i+1}^2 + y_{i+1}^2}$ and $s_2 = \sqrt{x_{i-1}^2 + y_{i-1}^2}$ respectively, then since $\cos\alpha_1 = x_{i+1}/s_1$, $\cos\alpha_2 = x_{i-1}/s_2$,

$\sin \alpha_1 = y_{i+1}/s_1$, and $\sin \alpha_2 = y_{i-1}/s_2$, the equations above may be rewritten as

$$\begin{aligned} yx_{i+1} - xy_{i+1} - ds_1 &= 0, \\ \Longrightarrow y &= \frac{xy_{i+1} + ds_1}{x_{i+1}} \end{aligned} \tag{1}$$

and

$$\begin{aligned} yx_{i-1} - xy_{i-1} - ds_2 &= 0, \\ \Longrightarrow y &= \frac{xy_{i-1} + ds_2}{x_{i-1}}. \end{aligned} \tag{2}$$

Using (1) and (2), the intersection point (x_i', y_i') is determined as follows:

$$\begin{aligned} \frac{xy_{i+1} + ds_1}{x_{i+1}} &= \frac{xy_{i-1} + ds_2}{x_{i-1}}, \\ \Rightarrow (xy_{i+1} + ds_1)x_{i-1} &= (xy_{i-1} + ds_2)x_{i+1} \\ \Rightarrow xx_{i-1}y_{i+1} + dx_{i-1}s_1 &= xx_{i+1}y_{i-1} + dx_{i+1}s_2 \\ \Rightarrow x &= \frac{d(x_{i+1}s_2 - x_{i-1}s_1)}{x_{i-1}y_{i+1} - x_{i+1}y_{i-1}}, \end{aligned}$$

provided that $x_{i-1}y_{i+1} - x_{i+1}y_{i-1} \neq 0$. Similarly, $y = d(y_{i+1}s_2 - y_{i-1}s_1)/(x_{i-1}y_{i+1} - x_{i+1}y_{i-1})$ provided that $x_{i-1}y_{i+1} - x_{i+1}y_{i-1} \neq 0$.

If $x_{i-1}y_{i+1} - x_{i+1}y_{i-1}$ is indeed equal to zero, then the lines A and B are collinear, and the intersection is not calculated.

Line following

Follow symbols are placed on line symbols with a predetermined distance between consecutive symbols. This distance may be interpreted in two ways. With a *true* distance measurement, the distance between consecutive symbols is measured along the curvature of the line. On the other hand, if the distance is representative of the radius of a circle placed on one follow symbol, with the next follow symbol being placed at the intersection of this circle with the curvature of the line, the distance is called a *visual* distance measurement.

True follow distance. The placing of follow symbols according to the true distance measurement presents few problems. Line segment lengths are merely accumulated until a length greater than the true follow distance is found. The follow symbol should then be placed on the last line segment used during the accumulation process. Note that if a single line segment is longer than the required distance, more than one follow symbol might have to be placed on that particular segment.

Visual follow distance. With the placing of visual type follow symbols the distances between a previously placed follow symbol and the endpoints of consecutive line segments are compared to the required visual follow distance. If any of these distances exceed the visual follow distance, a new follow symbol should be placed. Once again, the symbol is placed on the last line segment under consideration, or more specifically, at the intersection of the circle and the line segment as illustrated in figure 5. A formula for determining (x, y) may be derived as follows (Scheepers 1987b):

Line equation:

$$\begin{aligned} y &= mx + c \\ &= x \tan\theta + c && \text{if } \cos\theta \neq 0 \\ &= x\frac{\sin\theta}{\cos\theta} + c \\ &= \frac{x\sin\theta + z}{\cos\theta}, \end{aligned} \tag{3}$$

with

$$\begin{aligned} z &= c\cos\theta \\ &= (y - x\tan\theta)\cos\theta \\ &= y\cos\theta - x\sin\theta. \end{aligned} \tag{4}$$

Circle equation:

$$y^2 = r^2 - x^2. \tag{5}$$

$(3)^2$ - (4):

$$\begin{aligned} 0 &= (x\frac{\sin\theta}{\cos\theta} + \frac{z}{\cos\theta})^2 - (r^2 - x^2) \\ &= x^2\frac{\sin^2\theta}{\cos^2\theta} + x\frac{2z\sin\theta}{\cos^2\theta} + \frac{z^2}{\cos^2\theta} - r^2 + x^2 \\ &= x^2\sin^2\theta + x^2\cos^2\theta + 2xz\sin\theta + z^2 - r^2\cos^2\theta \\ &= x^2 + 2z\sin\theta x + z^2 - r^2\cos^2\theta. \end{aligned}$$

Thus:

$$\begin{aligned} x &= \frac{-2z\sin\theta \pm \sqrt{4z^2\sin^2\theta - 4(z^2 - r^2\cos^2\theta)}}{2} \\ &= -z\sin\theta \pm \sqrt{z^2\sin^2\theta - z^2 + r^2\cos^2\theta} \\ &= -z\sin\theta \pm \sqrt{z^2(\sin^2\theta - 1) + r^2\cos^2\theta} \\ &= -z\sin\theta \pm \sqrt{-z^2\cos^2\theta + r^2\cos^2\theta} \\ &= -z\sin\theta \pm \cos\theta\sqrt{r^2 - z^2}. \end{aligned} \tag{6}$$

(6) into (3) gives:

$$\begin{aligned} y &= \frac{(-z\sin\theta \pm \cos\theta\sqrt{r^2 - z^2})\sin\theta + z}{\cos\theta} \\ &= \frac{-z\sin^2\theta}{\cos\theta} \pm \sin\theta\sqrt{r^2 - z^2} + \frac{z}{\cos\theta}. \end{aligned}$$

Let $s = \sqrt{r^2 - z^2}$, then, from (4):

$$\begin{aligned} y &= \frac{-(y_1\cos\theta - x_1\sin\theta)\sin^2\theta}{\cos\theta} \pm s\sin\theta + \frac{y_1\cos\theta - x_1\sin\theta}{\cos\theta} \\ &= x_1\frac{\sin^3\theta}{\cos\theta} - y_1\sin^2\theta \pm s\sin\theta + y_1 - x_1\frac{\sin\theta}{\cos\theta} \\ &= x_1\frac{\sin\theta}{\cos\theta}(\sin^2\theta - 1) - y_1(\sin^2\theta - 1) \pm s\sin\theta \\ &= \frac{-x_1\sin\theta\cos^2\theta}{\cos\theta} + y_1\cos^2\theta \pm s\sin\theta \\ &= y_1\cos^2\theta - x_1\sin\theta\cos\theta \pm s\sin\theta. \end{aligned}$$

Continuing with (6):

$$\begin{aligned} x &= -z\sin\theta \pm s\cos\theta \\ &= -(y_1\cos\theta - x_1\sin\theta)\sin\theta \pm s\cos\theta \\ &= x_1\sin^2\theta - y_1\sin\theta\cos\theta \pm s\cos\theta, \end{aligned}$$

where

$$\begin{aligned} s &= \sqrt{r^2 - z^2} \\ &= \sqrt{r^2 - (y_1 \cos\theta - x_1 \sin\theta)^2} \\ &= \sqrt{r^2 - (y_1^2 \cos^2\theta - 2x_1 y_1 \sin\theta\cos\theta + x_1^2 \sin^2\theta)}. \end{aligned}$$

The orientation of follow symbols

The orientation of a particular follow symbol is easily determined. If the required orientation is *upright*, the symbol is merely placed in position. If, on the other hand, the orientation is either *parallel* or *perpendicular*, the orientation of the line segment on which the symbol is to be placed determines the orientation of the follow symbol.

Overlap

The overlap specification is used only if both methods for defining a line symbol is used simultaneously. If overlap is *forbidden*, the single or double line needs to be interrupted in the vicinity of each follow symbol such that no overlap occurs (indeed a form of clipping). One possible solution to this problem is to determine a bounding circle around each follow symbol, and to intersect this circle with the appropriate line segments of the single or double line. For this purpose an extension of the formula for placing visual type follow symbols may be used.

THE CONSTRUCTION OF AREA SYMBOLS

The construction of vector-based area symbols presents some very interesting problems. The areal extent of these symbols requires the use of hatching (or filling) algorithms. Furthermore, matters are complicated by the fact that cartographic regions might include 'holes' or *islands*.

From the specification for map symbolism presented earlier, an area symbol may be defined in two ways. Firstly, the borders of the area symbol could be constructed as if these borders were line symbols, and secondly, the inside of the closed region could be filled with hatch lines or with fill symbols. The type of hatch pattern (*hatch* or *crosshatch*) should be considered. Note that using the one method does not preclude the use of the other.

If both methods are used simultaneously clipping (*clip* or *don't clip*) will also have to be considered.

Hatching and filling

Traditionally, multiple simply-closed polygons with non-intersecting edges are used to approximate geographical regions by a single primitive. The lists of vertices representing the polygons are generally ordered in such a way that the *inside* of the region is implicitly defined (for example, outer boundary clockwise, all inner boundaries counter-clockwise). The methodology propagated here is to subdivide or partition regions into more manageable areas prior to hatching or filling.

The *PMP* partitioning algorithm. If a polygon P is considered topologically, three types of vertices can be identified with respect to the y-axis. A vertex V_i of P is called a *peak* if both $y(V_{i-1})$ and $y(V_{i+1})$ are less than $y(V_i)$, and it is called a *pit* if both $y(V_{i-1})$ and $y(V_{i+1})$ are greater than $y(V_i)$ (Cromley 1984). Vertices that are neither peaks nor pits are refered to as *regular* vertices. A polygon containing only one peak and one pit is called *monotone* (Lee 1981).

Following this terminology, define a *peak of type-1* to be a peak of the inside and a *peak of type-2* to be a peak of the outside of the area of interest as indicated in figure 6. Similarly, let a pit of the inner region be called a *type-1 pit* and a pit of the outer region be a *type-2 pit.* Furthermore, define a *pseudo-monotone polygon* (*pmp*) to be a polygon that contains exactly two non-crossing, non-descending routes from its minimum to its maximum vertex.

If the region of interest in figure 6 is now considered to be a piece of paper, possibly with holes cut into it, then using a pair of scissors, it would be simple to 'cut away' *pmp*'s by cutting from every type-2 peak or pit in a horizontal direction until the edge of the paper in that direction is reached. The result of this cutting would yield six pieces of paper without any holes in them, all *pmp*'s, or in the general case:

$$n(pmp) = n(\text{type-2 peaks} + \text{type-2 pits}) + 1 - n(\text{islands})$$

Note that the actual cutting direction is arbitrary as a change of direction would result in the same number of *pmp*'s. Also note that the restriction placed by identifying peaks and pits *with respect to the y-axis* and cutting in a *horizontal direction* is also arbitrary, since the polygons representing a region to be partitioned can always be rotated if required.

Filling with hatch lines. Consider figure 7, where a hypothetical left hatch limit is illustrated. Instead of intersecting each hatch line with this limit to determine the hatch line end points, an incremental displacement along each edge is calculated (see also Brassel 1979, Cromley 1984 and Scheepers 1987a).

Let L_i denote the left end points of hatch lines. Then, by calculating $x(L_1)$ and $y(L_1)$ once for each hatch limit, and by repeatedly adding d_x and d_y, it is fast and simple to determine the other values of L_i. Assume that apart from being parallel to the x-axis of a Cartesian coordinate system, the hatch lines also have integer y-axis values.[1] Let $\lfloor$ and $\lceil$ represent floor and ceiling functions, then from this assumption it follows that $y(L_1) = \lceil(y(A))$. Let $DX = x(B) - x(A)$ and $DY = y(B) - y(A)$. Then

$$\frac{x(L_1) - x(A)}{y(L_1) - y(A)} = \frac{DX}{DY}$$

and therefore

$$x(L_1) = x(A) + (y(L_1) - y(A))\frac{DX}{DY}.$$

Similarly, $x(L_2) = x(A) + (y(L_2) - y(A))\frac{DX}{DY}$ and from figure 8 and the assumption, it follows that $d_y = y(L_2) - y(L_1) = 1$ and $d_x = x(L_2) - x(L_1) = \frac{DX}{DY}$.

Hence, the left end points of hatch lines are:

$$x(L_i) = x(L_{i-1}) + d_x$$

$$y(L_i) = y(L_{i-1}) + 1$$

for all $i = 2 \ldots \mu$, $\mu = \lfloor(y(B)) - \lceil(y(A)) + 1$.

The same method is used to determine the coordinates of the right end points of hatch lines.

Filling with symbols. A similar approach to the one discussed above is taken to construct a fill pattern. First, *conceptual hatch lines* are determined. These lines are used to position fill symbols incrementally to form a regular pattern.

[1]This assumption can be made without a loss of generality, since the vertices of the polygons representing the region can always be scaled accordingly.

Let s represent the spacing between consecutive fill symbols measured along conceptual hatch lines, and define an indentation factor i/o in terms of s, where i represents the number of intervals in s, and o the *indentation* (in terms of i) between consecutive hatch lines (see figure 7).

If P_j are the points on a conceptual hatch line where fill symbols should be placed, the following procedure may be used to calculate the coordinates of P_j. (Assume that P_j have integer x-axis values, since if this was not the case, scaling by s could easily be executed. Then by determining the coordinates of P_1 once for each conceptual hatch line, and repeatedly adding $d_x = x(P_2) - x(P_1)$, it is simple and fast to determine the other values of P_j.)

Procedure:

- Calculate $\beta = y(A) \bmod i$ and $\delta = \lceil(x(A) + \frac{\beta}{i})$.
- Calculate the *indentation function* $\Psi(i,o)$:

$$\Psi(i,o) = \begin{cases} \frac{\beta}{i} & \text{for } \lceil(x(A)) = \delta \\ -\frac{(\beta \times o) \bmod i}{i} & \text{otherwise.} \end{cases}$$

- Let $DX = x(B) - x(A)$ and determine μ, where:

$$\mu = \frac{\lceil(x(A)) + \Psi(i,o) - x(A)}{DX}$$

- Calculate ν, the relationship between one spacing increment and distance AB:

$$\nu = \frac{x(P_2) - x(P_1)}{DX}.$$

- Use μ and ν to determine the first point:

$$x(P_1) = x(A) + \mu DX$$

$$y(P_1) = y(A) + \mu DY$$

  where $DX = x(B) - x(A)$ and $DY = y(B) - y(A)$.
- Determine the rest of the points:

$$x(P_j) = x(P_{j-1}) + \nu DX$$

$$y(P_j) = y(P_{j-1}) + \nu DY$$

  for all $j = 2 \ldots \gamma$, where $\gamma = \lceil(x(B) + \frac{\beta}{i}) - \lceil(x(A) + \frac{\beta}{i})$.

Clipping fill symbols

The clipping specification is used only if both border and inner area symbolization are used in the definition of an area symbol. If clipping is required (*clip*), the approach taken depends on whether follow symbols are used in the border symbolization. If follow symbols are not used, an algorithm for polygonal clipping of polygons should be used (Matthew 1985). If, on the other hand, follow symbols are used along the borders, no clear-cut solution exists. One possibility would be to construct a conceptual double line symbol on the border of the area symbol, and to use the inner line of this double line to clip against.

CONCLUDING REMARKS

This paper has presented a computer-assisted mechanism for constructing vector-based line and area symbols on maps. A specification for map symbolism has also been described. This specification was used as a reference basis for the aforementioned presentation.

The author wishes to thank Werner Strydom for preparing the illustrations.

REFERENCES

BRASSEL K E, FEGEAS R (1979). "An algorithm for shading of regions on vector display devices", *Computer Graphics,* **13** : 2, 126–133.

CROMLEY R G (1984). "The peak-pit-pass polygon line-shading procedure", *The American Cartographer,* **11** : 1, 70–79.

LEE D T (1981). "Shading of regions on vector display devices", *Computer Graphics,* **15** : 3, 37–44.

MATTHEW A J (1985). "Polygonal clipping of polylines", *Computer Graphics Forum,* 4, **407–414.** SCHEEPERS C F (1987A). "Polygon shading on vector type devices", *Quaestiones Informaticae,* **5** : 2, 46–55.

SCHEEPERS C F (1987B). "'n Vektorbenadering tot grafiese voorstelling in rekenaar-ondersteunde kartografie", *CSIR CACDS Technical Report TWISK 571,* Pretoria, November 1987, 168pp, (MSc dissertation in Afrikaans).

SCHEEPERS C F (1988). "Computer-assisted map symbolism", *Proceedings: 1988 ACSM-ASPRS Annual Convention,* **2** : Cartography, 47–56.

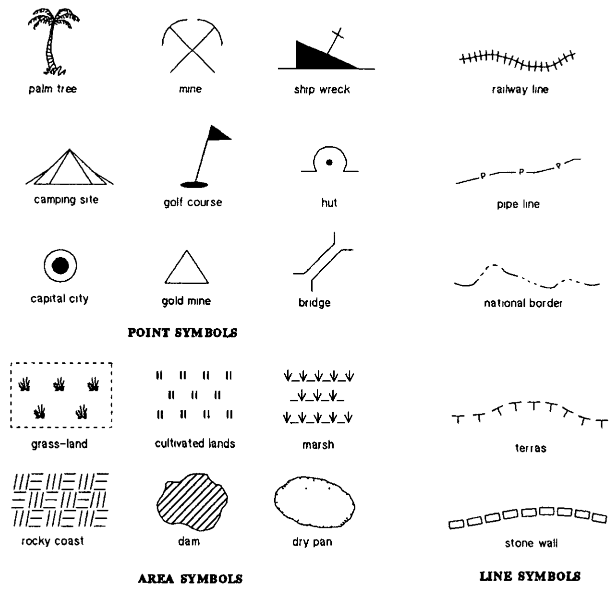


Figure 1: Examples of symbols

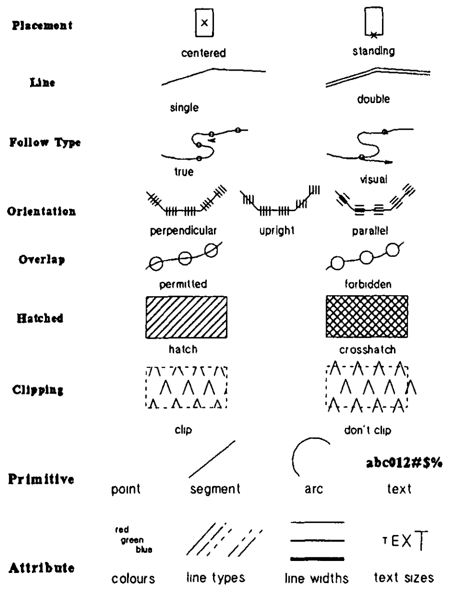


Figure 2: Terminals

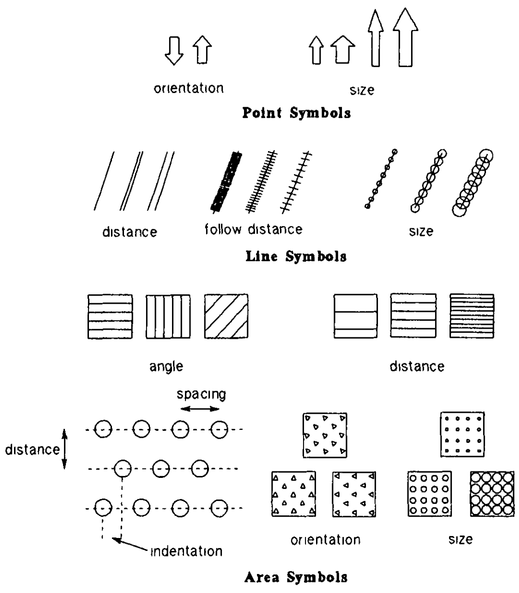


Figure 3: Parameters

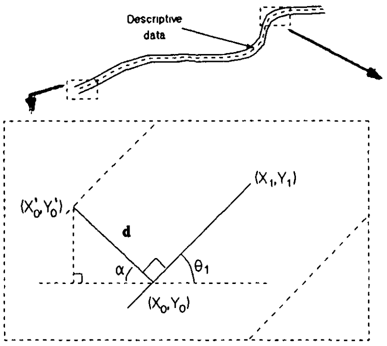
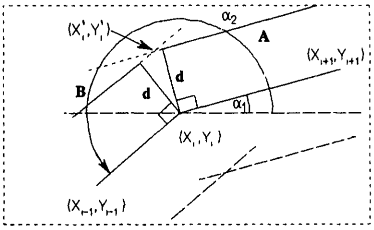


Figure 4: Double lines



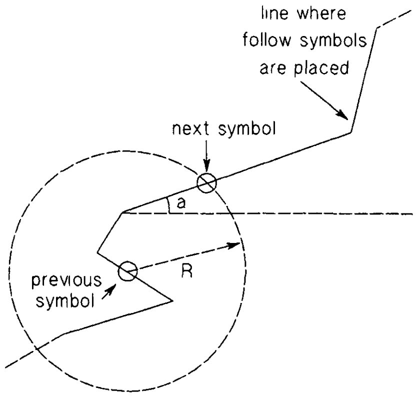


Figure 5: Visual follow symbol placement

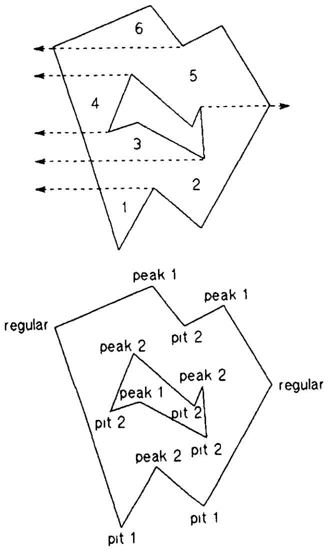
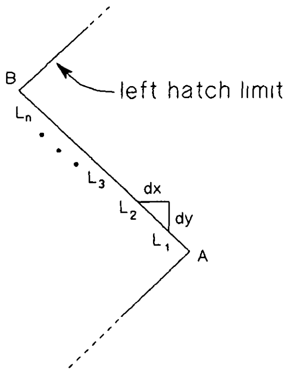
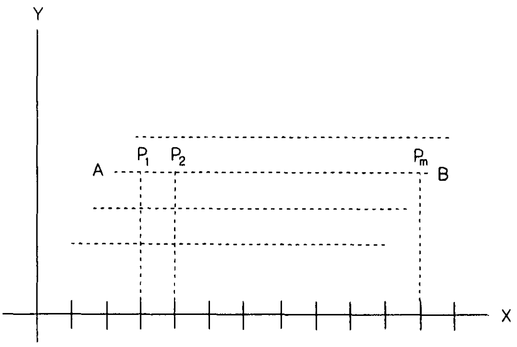


Figure 6: Polygon partitioning





Hatching **Filling**

Figure 7: Hatching and filling