Computer Graphics

Lecture 10

Attributes of output primitives

In general, any parameter that affects the way a primitive is to be displayed is referred to as an attribute parameter. Some attribute parameters, such as colour and size, determine the fundamental characteristics of a primitive. Others specify how the primitive is to be displayed under special conditions. Examples of attributes in this class include depth information for three-dimensional viewing and visibility or detectability options for interactive object-selection programs. These special-condition attributes will be considered in later classes. Here, we consider only those attributes that control the basic display properties of primitives, without regard for special situations. For example, lines can be dotted or dashed, fat or thin, and blue or orange. Areas might be filled with one colour or with a multicolour pattern. Text can appear reading from left to right, slanted diagonally across the screen, or in vertical columns. Individual characters can be displayed in different fonts, colours, and sizes. And we can apply intensity variations at the edges of objects to smooth out the raster stair step effect.

One way to incorporate attribute options into a graphics package is to extend the parameter list associated with each output primitive function to include the appropriate attributes. A line drawing function, for example, could contain parameters to set colour, width, and other properties, in addition to endpoint coordinates. Another approach is to maintain a system list of current attribute values. Separate functions are then included in the graphics package for setting the current values in the attribute list. To generate an output primitive, the system checks the relevant attributes and invokes the display routine for that primitive using the current attribute settings. Some packages provide users with a combination of attribute functions and attribute parameters in the output primitive commands. With the GKS and PHIGS standards, attribute settings are accomplished with separate functions that update a system attribute list

LINE ATTRIBUTES

Basic attributes of a straight line segment are its type, its width, and its colour. In the following sections, we consider how line drawing routines can be modified to accommodate various attribute specifications

Line Type: Possible selections for the line-type attribute include solid lines, dashed lines, and dotted lines. We modify a line drawing algorithm to generate such lines by setting the length and spacing of displayed solid sections along the line path. A dashed line could be displayed by generating an inter-dash spacing that is equal to the length of the solid sections. Both the length of the dashes and the inter-dash spacing are often specified as user options. A dotted line can be displayed by generating very short dashes with the spacing equal to or greater than the dash size. Similar methods are used to produce other line-type variations.

To set line type attributes in a PHIGS application program, a user invokes the function

setLinetype(lt)

where parameter lt is assigned a positive integer value of 1,2,3, or 4 to generate lines that are, respectively, solid, dashed, dotted, or dash-dotted. Other values for the line-type parameter It could be used to display variations in the dot dash patterns. Once the line-type parameter has been set in a PHIGS application program, all subsequent line-drawing commands produce lines with this line type.

Raster line algorithms display line-type attributes by plotting pixel spans. For the various dashed, dotted, and dot-dashed pattern, the line-drawing procedure outputs sections of contiguous pixels along the line path, skipping over a number of intervening pixels between the solid spans. Pixel counts for the span length and inter-span spacing can be specified in a pixel **mask**, which is a string containing the digits 1 and 0 to indicate which positions to plot along the line path. The mask 1111000, for instance, could be used to display a dashed line with a dash length of four pixels and an inter-dash spacing of three pixels. On a bi-level system, the mask gives the bit values that should be loaded into the frame buffer along the line path to display the selected line type.

Line Width: Implementation of line- width options depends on the capabilities of the output device. A heavy line on a monitor could be displayed as adjacent parallel lines, while a pen plotter might require pen changes. As with other PHIGS attributes, a line-width command is used to set the current line-width value in the attribute list. This value is then used by line-drawing algorithms to control the thickness of lines generated with subsequent output primitive commands.

We set the line-width attribute with the command

setLinewidthScaleFactor(lw)

Line-width parameter lw is assigned a positive number to indicate the relative width of the line to be displayed. A value of 1 specifies a standard-width line. On a pen plotter, for instance, a user could set lw to a value of 0.5 to plot a line whose width is half that of the standard line. Values greater than 1 produce lines thicker than the standard.

For raster implementation, a standard-width line is generated with single pixels at each sample position, as in the Bresenham algorithm. Other width link are displayed as positive integer multiples of the standard line by plotting additional pixels along adjacent parallel line paths.

Pen and Brush Options: With some packages, lines can be displayed with pen or brush selections. Options in this category include shape, size, and pattern. Some possible pen or brush shapes are given in Fig. 4-7. These shapes can be stored in a pixel mask that identifies the array of pixel positions that are to be set along the line path. For example, a rectangular pen can be implemented with the mask shown in Fig 4-8 by moving the centre (or one corner) of the mask along the line path. To avoid setting pixels more than once in the frame buffer, we can simply accumulate the horizontal spans generated at each position of the mask and keep track of the beginning and ending x positions for the spans across each scan line.

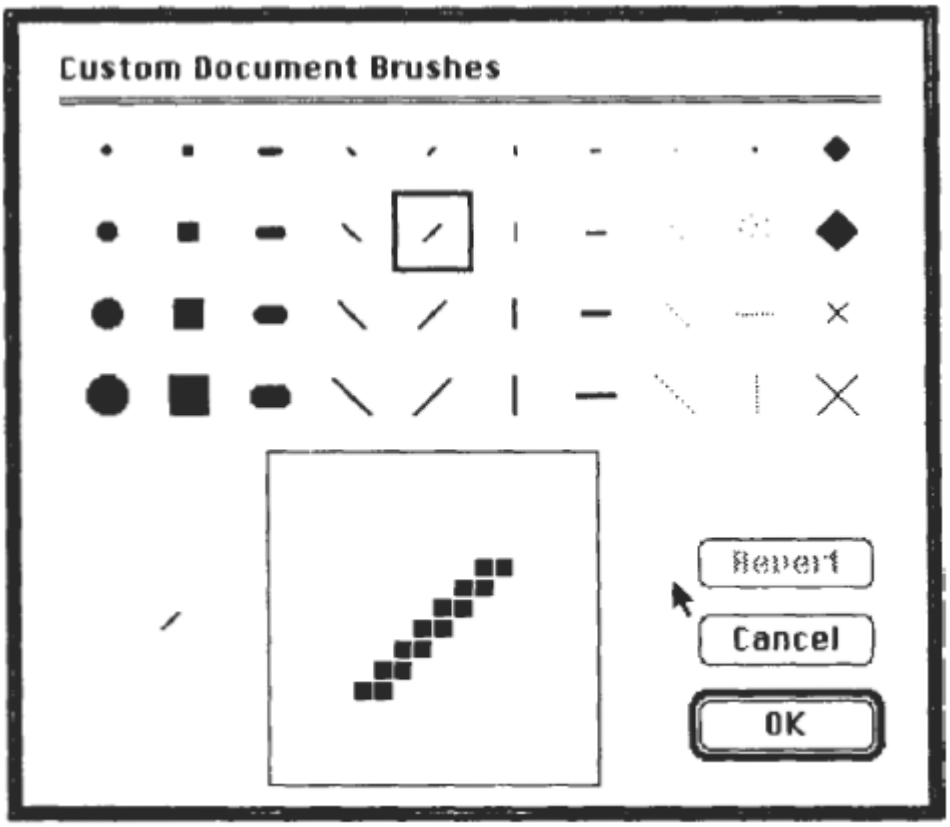

Fig 4-7 Pen and brush shapes for line display

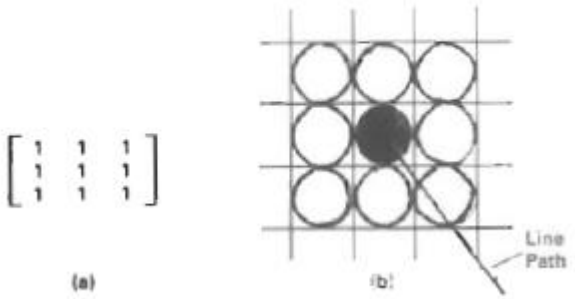

Figure 4-8

(a) ***A*** pixel **`mask`** for a rectangular pen, and **(b)** the associated array of pixels displayed by centring the mask over a specified pixel position.

Lines generated with pen (or brush) shapes can be displayed in various widths by changing the size of the mask

Line Colour: When a system provides colour (or intensity) options, a parameter giving the current colour index is included in the list of system-attribute values. A polyline routine displays a line in the current colour by setting this colour value in the frame buffer at pixel locations along the line path using the setPixel procedure. The number of colour choices depends on the number of bits available per pixel in the frame buffer.
We set the line colour value in PHlCS with the function

setPolylineColourIndex(lc)

Nonnegative integer values, corresponding to allowed colour choices, are assigned to the line colour parameter lc. A line drawn in the background colour is invisible, and a user can erase a previously displayed line by re-specifying it in the background colour (assuming the line does not overlap more than one background colour area).