**4. ATTRIBUTES OF OUTPUT PRIMITIVES**

Any parameter that affects the way a primitive is to be displayed is referred to as an attribute parameter. Example attribute parameters are color, size etc. A line drawing function for example could contain parameter to set color, width and other properties.

1. Line Attributes
2. Curve Attributes
3. Color and Grayscale Levels
4. Area Fill Attributes
5. Character Attributes
6. Bundled Attributes

**LINE ATTRIBUTES**

Basic attributes of a straight line segment are its type, its width, and its color. In some graphics packages, lines can also be displayed using selected pen or brush options

* Line Type
* Line Width
* Pen and Brush Options
* Line color

**Line type**

Possible selection of line type attribute includes solid lines, dashed lines and dotted lines.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 solid, dashed, dash dotted respectively. Other values for line type parameter it could be used to display variations in dot-dash patterns.

**Line width**

Implementation of line width option depends on the capabilities of the output device to set the line width attributes.

**setLinewidthScaleFactor(lw)**

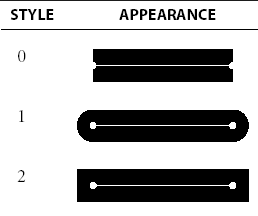
Line width parameter lw is assigned a positive number to indicate the relative width of line to be displayed. A value of 1 specifies a standard width line. A user could set lw to a value of 0.5 to plot a line whose width is half that of the standard line. Values greater than 1 produce lines thicker than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

**Line Cap**

We can adjust the shape of the line ends to give them a better appearance by adding line caps.

There are three types of line cap. They are

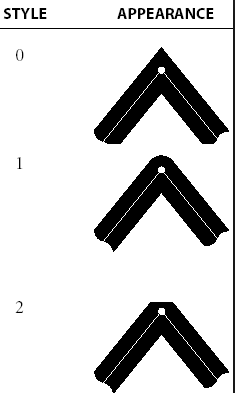
* Butt cap
* Round cap
* Projecting square cap



Butt cap obtained by adjusting the end positions of the component parallel lines so that the thick line is displayed with square ends that are perpendicular to the line path.Round cap obtained by adding a filled semicircle to each butt cap. The circular arcs are centered on the line endpoints and have a diameter equal to the line thicknessProjecting Square cap extend the line and add butt caps that are positioned one half of the line with beyond the secified end points.

Three possible methods for smoothly joining two line segments

* Mitter Join
* Round Join
* Bevel Join



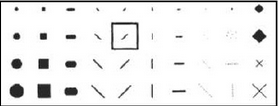
A miter join accomplished by extending the outer boundaries of each of the two lines until they meet.

A round join is produced by capping the connection between the two segments with a circular boundary whose diameter is equal to the width.

A bevel join is generated by displaying the line segment with but caps and filling in tri angular gap where the segments meet.

**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.



A poly line routine displays a line in the current color by setting this color value in the frame buffer at pixel locations along the line path using the set pixel procedure. We set the line color value in PHlGS with the function

**setPolylineColourIndex (lc)**

Nonnegative integer values, corresponding to allowed color choices, are assigned to the line color parameter lc

Example : Various line attribute commands in an applications program is given by the following sequence of statements

setLinetype(2);

setLinewidthScaleFactor(2);

setPolylineColourIndex (5);

polyline(n1,wc points1);

setPolylineColorIindex(6);

poly line (n2, wc points2);

This program segment would display two figures, drawn with double-wide dashed lines. The first is displayed in a color corresponding to code 5, and the second in color 6.

**CURVE ATTRIBUTES**

Parameters for curve attribute are same as those for line segments. Curves displayed with varying colors, widths, dot –dash patterns and available pen or brush options

**COLOR AND GRAYSCALE LEVELS**

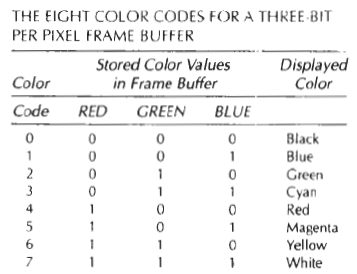
Color scale

Various color and intensity-level options can be made available to a user, depending on the capabilities and design objectives of a particular system. In a color raster system, the number of color choices available depends on the amount of storage provided per pixel in the frame buffer.

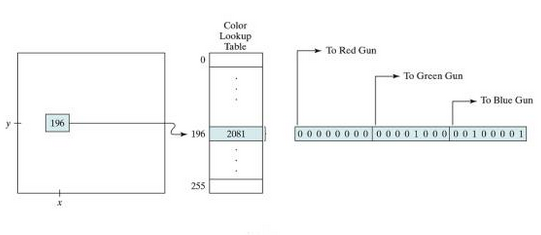
Color-information can be stored in the frame buffer in two ways:

1. We can store color codes directly in the frame buffer
2. We can put the color codes in a separate table and use pixel values as an index into this table

With the direct storage scheme, whenever a particular color code is specified in an application program, the corresponding binary value is placed in the frame buffer for each-component pixel in the output primitives to be displayed in that color.

A minimum number of colors can be provided in this scheme with 3 bits of storage per pixel, as shown in Table

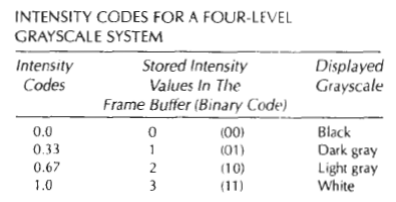
Color tables (Color Lookup Tables) are an alternate means for providing extended color capabilities to a user without requiring large frame buffers.



Grayscale

With monitors that have no color capability, color functions can be used in an application program to set the shades of gray, or grayscale, for displayed primitives. Numeric values over the range from 0 to 1 can be used to specify grayscale levels,

which are then converted to appropriate binary codes for storage in the raster.



**AREA FILL ATTRIBUTES**

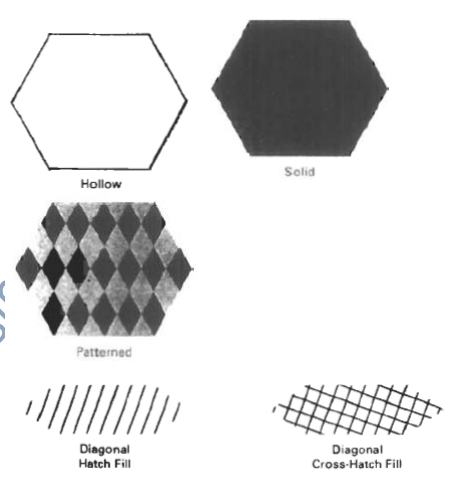
Options for filling a defined region include a choice between a solid color or a pattern fill and choices for particular colors and patterns

**Fill Styles**

Areas are displayed with three basic fill styles: hollow with a color border, filled with a solid color, or filled with a specified pattern or design. A basic fill style is selected in a PHIGS program with the function

**setInteriorStyle(fs)**

Values for the fill-style parameter fs include hollow, solid, and pattern. Another value for fill style is hatch, which is used to fill an area with selected hatching patterns-parallel lines or crossed lines



**Pattern Fill**

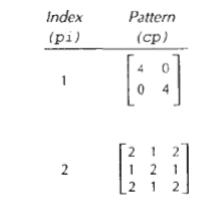
We select fill patterns with setInteriorStyleIndex (pi) where pattern index parameter pi specifies a table position.

For example, the following set of statements would fill the area defined in the fillArea command with the second pattern type stored in the pattern table:

**SetInteriorStyle( pattern)**

**SetInteriorStyleIndex(2);**

Fill area (n, points)

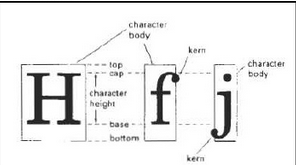


**CHARACTER ATTRIBUTES**

The appearance of displayed character is controlled by attributes such as font, size, color and orientation. Attributes can be set both for entire character strings (text) and for individual characters defined as marker symbols

**Text Attributes**

The choice of font or type face is set of characters with a particular design style as courier, Helvetica, times roman, and various symbol groups. The characters in a selected font also be displayed with styles. (solid,dotted, double) in bold face in italics, and in or shadow styles.



A particular font and associated stvle is selected in a PHIGS program by setting an integer code for the text font parameter tf in the function

**setTextFont(tf)**

Control of text color (or intensity) is managed from an application program with

**setTextColourIndex(tc)**

where text color parameter tc specifies an allowable color code.

Text size can be adjusted without changing the width to height ratio of characters with

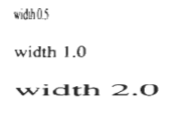
**SetCharacterHeight (ch)**

****

Parameter ch is assigned a real value greater than 0 to set the coordinate height of capital letters. The width only of text can be set with function.

**SetCharacterExpansionFactor(cw)**

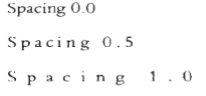
Where the character width parameter cw is set to a positive real value that scales the body width of character.



Spacing between characters is controlled separately with

**setCharacterSpacing(cs)**

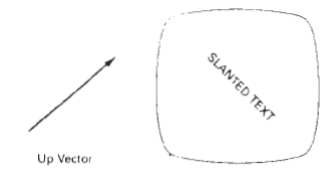
where the character-spacing parameter cs can he assigned any real value



The orientation for a displayed character string is set according to the direction of the character up vector

**setCharacterUpVector(upvect)**

Parameter upvect in this function is assigned two values that specify the x and y vector components. For example, with upvect = (1, 1), the direction of the up vector is 45 degrees and text would be displayed as



To arrange character strings vertically or horizontally

**setTextPath (tp)**

Where the text path parameter tp can be assigned the value: right, left, up, or down

Another handy attribute for character strings is alignment. This attribute specifies how text is to be positioned with respect to the $tart coordinates. Alignment attributes are set with

**setTextAlignment (h,v)**

where parameters h and v control horizontal and vertical alignment. Horizontal alignment is set by assigning h a value of left, center, or right. Vertical alignment is set by assigning v a value of top, cap, half, base or bottom.

A precision specification for text display is given with

**setTextPrecision (tpr)**

tpr is assigned one of values string, char or stroke.

**MARKER ATTRIBUTES**

A marker symbol is a single character that can he displayed in different colors and in different sizes. Marker attributes are implemented by procedures that load the chosen character into the raster at the defined positions with the specified color and

size. We select a particular character to be the marker symbol with

**setMarkerType(mt)**

where marker type parameter mt is set to an integer code. Typical codes for marker type are the integers 1 through 5, specifying, respectively, a dot (.) a vertical cross (+), an asterisk (\*), a circle (o), and a diagonal cross (X).

We set the marker size with

**setMarkerSizeScaleFactor(ms)**

with parameter marker size ms assigned a positive number. This scaling parameter is applied to the nominal size for the particular marker symbol chosen. Values greater than 1 produce character enlargement; values less than 1 reduce the marker

size.

Marker color is specified with

**setPolymarkerColourIndex(mc)**

A selected color code parameter mc is stored in the current attribute list and used to display subsequently specified marker primitives

**BUNDLED ATTRIBUTES**

The procedures considered so far each function reference a single attribute that specifies exactly how a primitive is to be displayed these specifications are called individual attributes.

A particular set of attributes values for a primitive on each output device is chosen by specifying appropriate table index. Attributes specified in this manner are called bundled attributes. The choice between a bundled or an unbundled specification is made by setting a switch called the aspect source flag for each of these attributes

**setIndividualASF( attributeptr, flagptr)**

where parameter attributer ptr points to a list of attributes and parameter flagptr points to the corresponding list of aspect source flags. Each aspect source flag can be assigned a value of individual or bundled.

**Bundled line attributes**Entries in the bundle table for line attributes on a specified workstation are set with the function

**setPolylineRepresentation (ws, li, lt, lw, lc)**

Parameter ws is the workstation identifier and line index parameter li defines the bundle table position. Parameter lt, lw, tc are then bundled and assigned values to set the line type, line width, and line color specifications for designated table index.

Example

**setPolylineRepresentation(1,3,2,0.5,1)**

**setPolylineRepresentation (4,3,1,1,7)**

A poly line that is assigned a table index value of 3 would be displayed using dashed lines at half thickness in a blue color on work station 1; while on workstation 4, this same index generates solid, standard-sized white lines

**Bundle area fill Attributes**

Table entries for bundled area-fill attributes are set with

**setInteriorRepresentation (ws, fi, fs, pi, fc)**

Which defines the attributes list corresponding to fill index fi on workstation ws. Parameter fs, pi and fc are assigned values for the fill style pattern index and fill color.

**Bundled Text Attributes**

**setTextRepresentation (ws, ti, tf, tp, te, ts, tc)**

bundles values for text font, precision expansion factor size an color in a table position for work station ws that is specified by value assigned to text index parameter ti.

**Bundled marker Attributes**

**setPolymarkerRepresentation (ws, mi, mt, ms, mc)**

That defines marker type marker scale factor marker color for index mi on workstation ws.