GKS

GKS (the Graphical Kernel System) is an ANSI and ISO standard.

GKS **standardizes two-dimensional graphics functionality** at a relatively low level.

The main objective of the Graphical Kernel System, GKS, is the **production and manipulation of pictures** in a computer or graphical **device independent way**.

The primary purposes of the standard are:

To **provide for portability** of graphics application programs.
To **aid in the understanding of graphics methods** by application programmers.
To **provide guidelines for manufacturers** in describing useful graphics capabilities.

It consists of **three basic parts**:

1. An **informal exposition** of the contents of the standard which includes such things as how text is positioned, how polygonal areas are to be filled, and so forth.

2. A **formalization of the expository material** by way of abstracting the ideas into discrete functional descriptions. These functional descriptions **contain such information as descriptions of input and output parameters, precise descriptions of the effect each function should have, references into the expository material, and a description of error conditions**. The functional descriptions in this section are language independent.

3. **Language bindings**. These bindings are an implementation of these abstract functions in a specific computer language such as Fortran or Ada or C.

In GKS, pictures are considered to be constructed from a number of **basic building blocks**. These basic building blocks are of a number of types each of which can be used to describe a different component of a picture.

The five main primitives in GKS are:

Polyline: which draws a sequence of connected line segments.
Polymarker: which marks a sequence of points with the same symbol.
Fill area: which displays a specified area.
Text: which draws a string of characters.
Cell array: which displays an image composed of a variety of colours or grey scales.

Associated with each primitive is a **set of parameters which is used to define particular instances** of that primitive.

For example, the parameters of the **Text** primitive are the **string or characters to be drawn** and the **starting position of that string**. Thus:

TEXT(X, Y, 'ABC') will draw the characters ABC at the position (X, Y).

Although the parameters enable the form of the primitives to be specified, additional data are necessary to describe the actual appearance (or aspects) of the primitives.

For example, GKS needs to know the **height of a character string** and **the angle** at which it is to be drawn. These additional data are known as **attributes**.

GKS Output Primitives

GKS standardizes a reasonably complete set of functions for displaying 2D images. It contains functions for drawing lines, markers, filled areas, text, and a function for representing raster like images in a device-independent manner.

In addition to these basic functions, GKS contains many **functions** for **changing the appearance** of the output primitives, such as changing **colors**, changing line **thicknesses**, changing **marker types** and **sizes** etc. The functions for changing the appearance of the fundamental drawing functions (output primitives) are called **attribute setting functions**.

Polylines

The GKS function for **drawing line segments** is called Polyline. The polyline function **takes an array of X-Y coordinates** and draws line segments connecting them. The attributes that control the appearance of a polyline are:
Linetype, which controls whether the polyline is drawn as a **solid, dashed, dotted,** or **dash-dotted line**.
Linewidth scale factor, which controls **how thick** the line is.
Polyline color index, which controls **what color** the line is.

The main line drawing primitive of GKS is the polyline which is generated by calling the function:
POLYLINE(N, XPTS, YPTS)
where XPTS and YPTS are arrays giving the N points (XPTS(1), YPTS(1)) to (XPTS(N), YPTS(N)). The polyline generated consists of N - 1 line segments joining adjacent points starting with the first point and ending with the last.

Polymarkers

The GKS polymarker function **allows drawing marker symbols** centered at coordinate points specified.
The attributes that control the appearance of polymarkers are:
Marker, which specifies **one of five standardized symmetric characters** to be used for the marker.
The five characters are **dot**, **plus**, **asterisk**, **circle**, and **cross**.

Marker size scale factor, which controls **how large** each marker is (except for the dot marker).

Polymarker color index, which specifies **what color** the marker is.

GKS provides the primitive polymarker marks a set of points, instead of drawing lines through a set of points.

A polymarker is generated by the function:
POLYMARKER(N, XPTS, YPTS)

where the arguments are the **same as for the Polyline function**, namely XPTS and YPTS are arrays giving the N points (XPTS(1), YPTS(1)) to (XPTS(N), YPTS(N)).
Polymarker **places a centered marker** at each point.
GKS recognizes the common use of markers to identify a set of points in addition to marking single points and so the marker function is a polymarker.
POLYMARKER(l9, XDK, YDK)

Text

The GKS text function **allows drawing a text string** at a specified coordinate position.
The attributes that control the appearance of text are:
Text font and **precision**, which specifies **what text font** should be used for the characters and **how precisely** their representation should adhere to the settings of the other text attributes.

Character expansion factor, which **controls the height-to-width ratio** of each plotted character.

Character spacing, which specifies **how much additional white space** should be inserted between characters in a string.

Text color index, which specifies **what color** the text string should be.

Character height, which specifies **how large** the characters should be.

Character up vector, which specifies at **what angle** the text should be drawn.

Text path, which specifies in what direction the text should be written (right, left, up, or down).

Text alignment, which specifies **vertical and horizontal centering** options for the text string.

A **text string** may be generated by invoking the function:

TEXT(X, Y, STRING) where (X, Y) is the text position and STRING is a string of characters.

The **character height attribute** determines the height of the characters in the string. Since a character in a font will have a designed aspect ratio, the character height also determines the character width. The character height is set by the function:
SET CHARACTER HEIGHT(H)
where H is the character height.

The **character up vector** is perhaps the most important text attribute. Its main purpose is to determine the orientation of the characters. However, it also sets a reference direction which is used in the determination of text path and text alignment. The character up vector specifies the up direction of the individual characters. It also specifies the orientation of the character string in that. By default, the characters are placed along the line perpendicular to the character up vector. The function:
SET CHARACTER UP VECTOR(X, Y)

Fill Area

The GKS fill area function **allows specifying a polygonal shape of an area to be filled** with various interior styles.
The attributes that control the appearance of fill areas are:
Fill area interior style, which specifies **how the polygonal area should be filled**: with solid colors or various hatch patterns, or with nothing, that is, a line is drawn to connect the points of the polygon, so to get only a border.
Fill area style index. If the fill area style is hatch, this index specifies **which hatch pattern** is to be used: horizontal lines; vertical lines; left slant lines; right slant lines; horizontal and vertical lines; or left slant and right slant lines.
Fill area color index, which specifies **the color of the fill patterns** or solid areas.

At the same time, there are now many devices which have the concept of an area which may be filled in some way. These vary from intelligent pen plotters which can cross-hatch an area to raster displays which can completely fill an area with a single colour or in some cases fill an area by repeating a pattern.

GKS provides a fill area function to satisfy the application needs which can use the varying device capabilities. Defining an area is a fairly simple extension of defining a polyline. An array of points is specified which defines the boundary of the area. If the area is not closed (i.e. the first point is not the same as the last point), the boundary is the polyline defined by the points but extended to join the last point to the first point. A fill area may be generated by invoking the function:

FILL AREA(N, XPTS, YPTS)
where, as usual XPTS and YPTS are arrays giving the N points (XPTS(1), YPTS(1)) to (XPTS(N), YPTS(N)).

Cell Array

The GKS cell array function **displays raster like images** in a device-independent manner.

The cell array function takes the two corner points of a rectangle specified, a number of divisions (M) in the X direction and a number of divisions (N) in the Y direction. It then partitions the rectangle into M x N subrectangles called cells. Assign each cell a color and create the final cell array by coloring each individual cell with its assigned color.