SOFTFILL, A SOFTWARE FILL PACKAGE

David J. Kennison
NCAR, P.O. Box 3000, Boulder, Colorado 80307-3000
email: kennison@ncar.ucar.edu (128.117.64.4)

SOFTFILL, A SOFTWARE FILL PACKAGE

David J. Kennison
NCAR, P.O. Box 3000, Boulder, Colorado 80307-3000
email: kennison@ncar.ucar.edu (128.117.64.4)

1.0 INTRODUCTION

The graphics package SOFTFILL allows the user to fill a polygonal subset of the plotter frame in any of various ways. It contains the user-callable subroutines SFSGFA, SFWRLD, SFNORM, SFGETC, SFGETI, SFGETP, SFGETR, SFSETC, SFSETI, SFSETP, and SFSETR.

The routine SFSGFA can be made to fill polygons by calling the GKS routine GFA or by calling the lower-level SOFTFILL routines SFWRLD and SFNORM. If all area fills are done using SFSGFA, then the way in which they are done can be changed by modifying the value of a single internal parameter of SOFTFILL.

The routines SFWRLD and SFNORM may be called directly by the user to fill polygons with parallel lines or with dots, polymarkers, or selected characters arrayed in a regular rectangular pattern. By calling these routines more than once for a given polygon, one can create effects like cross-hatching, use combinations of lines and dots, and so on.

Routines with names of the form SFGETx, where “x” stands for “C”, “I”, “P”, or “R”, are used to retrieve the current values of various internal parameters determining how the filling is to be done; routines with names of the form SFSETx are used to give those internal parameters new values.

Using the internal parameters, one may specify the distance between adjacent fill lines and the angle at which the fill lines are to be drawn. One may specify whether the lines are to be solid or dotted. If the lines are to be dotted, one may specify an 8x8 pattern according to which dots are to be included or omitted and one may specify whether each “dot” is to be just a dot or some other symbol.

All of the routines mentioned above are discussed in detail in the next section, named “ROUTINES”.

2.0 ROUTINES

2.1 SFSGFA (XRA,YRA,NRA,DST,NST,IND,NND,ICI)

The subroutine SFSGFA (which stands for “SOFTFILL - Simulate GFA”) is used to fill, in one of various ways, an area of the plotter frame defined by a given set of points; it is intended to provide a way to use the GKS fill-area routine, if it works (as is the case in the version of GKS distributed with NCAR Graphics), or a suitable pattern-fill substitute, otherwise. Doing all area fills with SFSGFA has the advantage that the way in which they are done can then be changed by modifying the value of a single internal parameter of SOFTFILL, named 'TY'.

2.1.1 Usage

The FORTRAN statement

```
CALL SFSGFA (XRA,YRA,NRA,DST,NST,IND,NND,ICI)
```

fills the polygonal area defined by the points

```
(XRA(I),YRA(I)) , for I = 1 to NRA
```

The straight-line segments connecting point 1 to point 2, point 2 to point 3, point 3 to point 4, ..., point NRA-1 to point NRA, and point NRA to point 1 bound the area to be filled. They are allowed to cross each other. The definitions of “inside” and “outside” are as in GKS: A given point is inside the polygon if a straight line emanating from that point intersects the boundary lines an odd number of times; otherwise, it is outside the polygon.

The values of the internal parameter 'TY' (for “TYPE OF FILL”) and the argument ICI determine how the fill should be done, as detailed below (for directions saying how to get a complete set of examples, see the section “EXAMPLES”) :

- When 'TY' has the value 0 (the default), a GKS fill area is generated (by calling GFA). ICI, if zero or greater, specifies the fill area color index. A negative value of ICI specifies that the fill area color index is not to be set before the call to GFA.
- When 'TY' has the value 1, the area is filled with parallel lines (by calling SFWRLD). ICI, if zero or greater, specifies the desired polyline color index. A negative value of ICI specifies that the polyline color index is not to be set before calling SFWRLD.
- When 'TY' has the value 2, the area is filled with parallel lines (by calling SFWRLD) and it is then filled again with parallel lines perpendicular to the first set (by calling SFNORM). ICI, if zero or greater, specifies the desired polyline color index. A negative value of ICI specifies that the polyline color index is not to be set before calling SFWRLD.
- When 'TY' has a negative value between -4 and -1, inclusive, one or more calls to SFWRLD and/or SFNORM are used to fill the area with lines forming a pattern. ICI is used to select that pattern. A zero or negative value of ICI will select a blank pattern. Positive values of ICI will select patterns which increase in density as the value of ICI

increases. The absolute value of 'TY' specifies which set of patterns is to be used; the largest usable value of ICI is approximately 5*ABS('TY'), beyond that, the pattern becomes essentially solid. The fill patterns generated may be described as follows: Let “n” represent the value ABS('TY'). For ICI=0, no fill lines are drawn at all. For each of the next “n” values of ICI, the pattern is elaborated by the addition of fill lines, 32*'SP' units apart, at one of the angles 'AN', 'AN'+180/n, 'AN'+2*180/n, ... For each of the next “n” values of ICI, the pattern is elaborated by halving the distance between fill lines at one of the angles. This final step is then repeated ad infinitum.

Values of 'TY' not specifically mentioned above are reserved for future use. (At the moment, values greater than 2 are treated as equal to 2 and values less than -4 are treated as equal to -4, but this may not be true in the future.)

To fill a simple polygon, just give the coordinates of its vertices, in the order in which they are encountered as the edge of the polygon is traced. To leave an unfilled hole in a filled area, add the vertices of the hole, in the proper order, repeat the first vertex of the hole, and then repeat the vertex which preceded the first vertex of the hole. To create a “negative” (filling what was unfilled, and not filling what was filled), just add the coordinates of the corners of the frame, in the proper order, and then repeat the coordinates of the first frame corner.

Note that, in the case of a complicated polygon (one containing holes), there will be connecting lines between the outer boundary of the polygon and the boundaries of the holes. When 'TY' has a non-zero value, these connecting lines cause no trouble; however, when 'TY' has the value 0, specifying that fill is to be done using calls to GFA, problems may arise. Hardware fill algorithms will frequently display unfortunate edge effects along such lines. Such problems may be minimized by using only horizontal or vertical connecting lines and by ensuring that they do not cross any of the original boundary lines.

2.1.2 Arguments

XRA (an input/output array of type REAL, dimensioned NRA or greater) contains the X coordinates of the points defining the area to be filled, in the user coordinate system. Upon return, the contents of XRA will have been converted to the fractional coordinate system.

YRA (an input/output array of type REAL, dimensioned NRA or greater) contains the Y coordinates of the points defining the area to be filled, in the user coordinate system. Upon return, the contents of YRA will have been converted to the fractional coordinate system.

NRA (an input expression of type INTEGER) is the number of points defining the area to be filled. NRA must be greater than two.

DST (a scratch array of type REAL, dimensioned NST) is for use when fill lines are generated by means of calls to SFWRLD and/or SFNORM.

NST (an input expression of type INTEGER) is the length of the array DST. NST must be greater than or equal to NRA + NIM, where NIM is the largest number of intersection points of any fill line with the boundary lines. To be sure DST is large enough, use NIM = NRA; in practice, NIM rarely needs to be that large. For a convex polygon, for example, NIM = 2.

IND (a scratch array of type INTEGER, dimensioned NND) is for use when fill lines are generated by means of calls to SFWRLD and/or SFNORM.

NND (an input expression of type INTEGER) is the length of the array IND. It must be greater than or equal to NRA + 2 * NIM, where NIM is as defined above.

ICI (an input expression of type INTEGER) is, nominally, the fill-area color index to be used. When the internal parameter 'TY' has a value other than zero, ICI may be used in some other manner (as described above under the heading "Usage").

2.2 SFWRLD or SFNORM (XRA,YRA,NRA,DST,NST,IND,NND)

The subroutines SFWRLD and SFNORM are used to fill, with evenly spaced parallel lines, that portion of the plotter frame inside the area defined by a given polygonal boundary.

2.2.1 Usage

Use one of the two FORTRAN statements

```
CALL SFWRLD (XRA,YRA,NRA,DST,NST,IND,NND)
CALL SFNORM (XRA,YRA,NRA,DST,NST,IND,NND)
```

Either of these statements causes the area defined by the points

```
(XRA(I),YRA(I)) , for I = 1 to NRA
```

to be filled in the manner specified by the current values of the internal parameters. The default values of these parameters yield solid horizontal lines .00125 fractional coordinate units apart.

Use SFWRLD if the coordinates are user coordinates in the range specified by the current window (as defined by arguments 5 through 8 of the last call to the SPPS routine SET or by the last call to the GKS routine GSWN). Use SFNORM if the coordinates are fractional coordinates - real values between 0 and 1.

Note: "User" coordinates and "fractional" coordinates are sometimes referred to as "world" coordinates and "normalized device" coordinates, respectively (hence the names of these routines), but, strictly speaking, "user" coordinates and "world" coordinates are not necessarily the same. The mapping from "user" coordinates to "fractional" coordinates is that specified by a call to the SPPS routine SET; the mapping may be linear or logarithmic in either direction and it may be mirror-imaged in either direction. The mapping from “world” coordinates to “normalized device” coordinates is as defined by the GKS standard: the mapping is linear in both directions and is not mirror-imaged.

The straight-line segments connecting point 1 to point 2, point 2 to point 3, point 3 to point 4, ..., point NRA-1 to point NRA, and point NRA to point 1 bound the area to be filled. They are allowed to cross each other. The boundary lines are not themselves drawn.

If a given fill line intersects the boundary lines at “n” points, the “n” points are ordered in the obvious way along the fill line and every other segment of the fill line (running from an odd-numbered point of intersection to an even-numbered point of intersection) is drawn.

To fill a simple polygon, just give the coordinates of its vertices, in the order in which they are encountered as the edge of the polygon is traced. To leave an unfilled hole in a filled area, add the vertices of the hole, in the proper order, repeat the first vertex of the hole, and then repeat the vertex which preceded the first vertex of the hole. To create a “negative” (filling what was unfilled, and not filling what was filled), just add the coordinates of the frame corners and then repeat the point which was originally last.

Consecutive calls, using different fill angles, may be used to create a cross-hatched effect.

The values of various internal parameters affect the nature of the fill done by these routines. For descriptions of them, see the descriptions of the routines “SFSETx” and the section “INTERNAL PARAMETERS”, below).

2.2.2 Arguments

XRA (an input/output array of type REAL, dimensioned NRA) contains the X coordinates of the points defining the area to be filled, in the user coordinate system (if SFWRLD is called) or in the fractional coordinate system (if SFNORM is called). Upon return from SFWRLD, the contents of XRA will have been converted to the fractional coordinate system.

YRA (an input/output array of type REAL, dimensioned NRA) contains the Y coordinates of the points defining the area to be filled, in the user coordinate system (if SFWRLD is called) or in the fractional coordinate system (if SFNORM is called). Upon return from SFWRLD, the contents of YRA will have been converted to the fractional coordinate system.

NRA (an input expression of type INTEGER) is the number of points defining the area to be filled. NRA must be greater than two.

DST (a scratch array of type REAL, dimensioned NST or greater) is for use by the fill algorithm.

NST (an input expression of type INTEGER) is the length of the array DST. It must be greater than or equal to NRA + NIM, where NIM is the largest number of intersection points of any fill line with the boundary lines. To be sure DST is large enough, use NIM =

NRA; in practice, NIM rarely needs to be that large. For a convex polygon, for example, NIM = 2 suffices.

IND (a scratch array of type INTEGER, dimensioned NND or greater) is for use by the fill algorithm.

NND (an input expression of type INTEGER) is the length of the array IND. It must be greater than or equal to NRA + 2 * NIM, where NIM is as defined above.

2.3 SFGETx (CNP,xVP)

These routines are used to retrieve the current values of the four internal parameters 'AN', 'CH', 'DO', 'SP', and 'TY'.

2.3.1 Usage

Use the FORTRAN statement

```
CALL SFGETx (CNP,xVP)
```

where “x” is a “C”, an “I”, or an “R” (depending on the type of the value involved), to retrieve, in the variable xVP, the value of the parameter specified by CNP.

2.3.2 Arguments

CNP (an input expression of type CHARACTER) is one of the character strings 'AN', 'CH', 'DO', 'SP', or 'TY' (or a longer string beginning with one of these strings), meaning “ANgle”, “CHaracter”, “DOts”, “SPacing”, and “TYpe”, respectively.

xVP (an output variable of type CHARACTER, INTEGER, or REAL, depending on the value of “x”) is the name of a variable into which the value of the parameter specified by CNP is to be retrieved.

- When CNP='AN', xVP is the angle at which fill lines are to be drawn, in degrees. Use “x” = “I” to get an integer value in IVP or use “x” = “R” to get a real value in RVP.
- When CNP='CH', xVP is the character selector, which only becomes important when 'DO' is non-zero, selecting dot fill. An integer 0 means that simple dots are to be used. A positive integer value “n” means that the character “CHAR(n)” is to be used. A negative integer “-n” implies the use of the GKS polymarker of type “n”. Use “x” = “I” to get an integer value in IVP or use “x” = “C” to get a character value in CVP (if the integer value is less than or equal to zero, a single blank will be returned in CVP).
- When CNP='DO', xVP is the dot-fill flag. A zero value means that solid lines are to be used, a non-zero value that dots are to be used (the dots to be arranged according to the pattern specified by the internal parameter array LDP). Use “x” = “I” to get an integer value in IVP.

- When CNP='SP', xVP is the distance between each fill line and the next (and between any pair of adjacent dots along a fill line). The value is given in the fractional coordinate system and represents a fraction of the distance across the screen. Use “x” = “R” to get a real value in RVP.
- When CNP='TY', xVP is the type of fill to be used by the routine SFSGFA. Use “x” = “I” to get an integer value in IVP. See the description of the routine SFSGFA for further information about 'TY'.

2.4 SFGETP (IDP)

This routine is used to retrieve the current dot pattern.

2.4.1 Usage

Use the FORTRAN statement

```
CALL SFGETP (IDP)
```

to retrieve the current contents of the 8x8 internal parameter array LDP, which specifies the dot pattern to be used when dot fill is selected.

2.4.2 Arguments

IDP (an output array of type INTEGER, dimensioned 8x8) is the array in which the dot pattern is to be returned. See the description of the routine SFSETP for further details.

2.5 SFSETx (CNP,xVP)

These routines are used to reset the current values of the four internal parameters 'AN', 'CH', 'DO', 'SP', and 'TY'.

2.5.1 Usage

Use the FORTRAN statement

CALL SFSETx (CNP,xVP)

where “x” is a “C”, an “I”, or an “R” (depending on the type of the value involved), to set the value of the parameter specified by CNP to the value of the variable xVP.

2.5.2 Arguments

CNP (an input expression of type CHARACTER) is one of the character strings 'AN', 'CH', 'DO', 'SP', or 'TY' (or a longer string beginning with one of these strings), meaning “ANgle”, “CHaracter”, “DOts”, “SPacing”, and “TYpe”, respectively.

xVP (an input expression of type CHARACTER, INTEGER, or REAL, depending on the value of “x”) is the desired new value of the parameter specified by CNP. This value will be used until the next call resetting it.

- When CNP='AN', xVP is the angle at which fill lines are to be drawn, in degrees. Use “x” = “I” and give an integer value for IVP or use “x” = “R” and give a real value for RVP. The default value is 0.
- When CNP='CH', xVP is the character selector, which only becomes important when 'DO' is non-zero, selecting dot fill. An integer 0 means that simple dots are to be used. A positive integer value “n” means that the character “CHAR(n)” is to be used. A negative integer “-n” implies the use of the GKS polymarker of type “n”. Use “x” = “I” and give an integer value for IVP or use “x” = “C” and give a desired character for CVP. The default value is 0.
- When CNP='DO', xVP is the dot-fill selector. The value 0 means that solid lines are to be used, the value 1 that dotted lines are to be used (the dots to be arranged according to the pattern specified by an internal array LDP - see the description of the routine SFSETP, below). Use “x” = “I” and give an integer value for IVP. The default value is 0.
- When CNP='SP', xVP is the distance between each fill line and the next (and between any pair of adjacent dots along a fill line). The value is given in the fractional coordinate system and represents a fraction of the distance across the plotter frame, between 0 and 1. Use “x” = “R” and give a real value for RVP. The default value is .00125. A value less than or equal to 0 or greater than or equal to 1 will have the same effect as the default.
- When CNP='TY', xVP is the type of fill to be used by the routine SFSGFA. Use “x” = “I” and give an integer value for IVP. See the description of the routine SFSGFA for further information about 'TY'. The default value is 0.

2.6 SFSETP (IDP)

This routine is used to define a new dot pattern.

2.6.1 Usage

Use the FORTRAN statement

```
CALL SFSETP (IDP)
```

to reset the current contents of the 8x8 internal parameter array LDP, which specifies the dot pattern to be used when dot fill is selected.

2.6.2 Arguments

IDP (an input array of type INTEGER, dimensioned 8x8) contains 0s and 1s specifying the dot pattern to be transferred to the internal array LDP for use by SFWRLD and SFNORM. When dot fill is selected, each fill line is drawn using dots the same distance apart along the line as the lines are from each other. The dots thus fall at the intersection points of a square grid. Each dot is associated with a particular element of LDP. If that element is a 1, the dot is drawn; if it is a 0, the dot is not drawn. The association is done in such a way as to replicate the pattern specified by LDP across the entire filled area.

If the fill angle is 0 (the default), then incrementing the first subscript of LDP corresponds to a horizontal motion across the plot from left to right and incrementing the second subscript of LDP corresponds to a vertical motion across the plot from top to bottom. This allows the contents of IDP to be declared in a DATA statement which creates a “picture” of the pattern. For example, the FORTRAN statements

```
      DIMENSION IDP(8,8)
      ...
      DATA IDP / 0,0,0,0,0,0,0,0,
     +           0,1,1,1,1,1,1,0,
     +           0,1,0,0,0,0,0,1,
     +           0,1,0,0,0,0,0,1,
     +           0,1,1,1,1,1,1,0,
     +           0,1,0,0,0,1,0,0,
     +           0,1,0,0,0,0,1,0,
     +           0,1,0,0,0,0,0,1/
```

creates the letter “R”, with the correct orientation.

The contents of the array IDP are transferred to the internal array LDP. The new dot pattern is used until a subsequent call to SFSETP changes the contents of LDP again.

The default dot pattern consists of all 1s.

3.0 INTERNAL PARAMETERS

The internal parameters of SOFTFILL are summarized in the following table. For details, see the descriptions of the routines which are used to set and retrieve them.

Name	Type	Description
'AN'	I or R	Angle of fill lines, in degrees counterclockwise from horizontal. The default value is zero.
'CH'	C or I	Character selector (-n => GKS polymarker n; 0 => dots; +n => CHAR(n)). The default value is 0.

Name	Type	Description
'DO'	I	Dot fill selector (0 => line fill; 1 => dot fill). The default value is 0, which implies line fill.
'SP'	R	Spacing of fill lines, in the fractional coordinate system. The default value is .00125.
'TY'	I	Type of fill to be used by SFSGFA (-1 to -4 => patterns using lines in 1 to 4 different directions and varying in density as the argument ICI varies; 0 => color fill, by calling GFA, with ICI specifying the color index; 1 => simulated color fill, using lines in one direction, with ICI specifying the color index; 2 => simulated color fill, using lines in two directions, with ICI specifying the color index). The default value is 0.
"LDP"	I	An 8x8 array of type INTEGER, containing nothing but 0s and 1s, describing the dot pattern to be used when 'DO' is set non-zero to select dot fill, rather than line fill. This internal parameter, unlike the others, is accessed using the routines SFGETP and SFSETP.

The examples described in the section "EXAMPLES", below, show the effect of all of the internal parameters; in particular, the effect of 'TY' is detailed in the second example.

4.0 ERROR HANDLING

When a SOFTFILL routine detects an error condition, it calls the routine SETER, which is the principal routine in the error-handling package for NCAR Graphics. (There is a programmer document describing SETER and associated routines; see that document for complete information about error handling in NCAR Graphics.)

By default, SETER prints a line and STOPs. The line printed will look something like this:

```
ERROR    3 IN SFGETR - PARAMETER NAME NOT KNOWN - X
```

The error number ("3", in the example) may be of use to a consultant (to determine exactly where the error occurred), but is not otherwise meaningful. The actual error message consists of the name of the routine in which the error occurred ("SFGETR", in the example), a blank, a minus sign, another blank, and, lastly, a short description of the error.

All errors are "recoverable" in the sense that, if the user program puts SETER into "recovery mode", control will be returned to the caller of the SOFTFILL routine in which the error occurred. In some cases, it is then possible to take remedial action to get around whatever problem has occurred; in any case, the error flag can be cleared and execution of the user's program can continue.

When SETER is in recovery mode (and, occasionally, even when it is not), error messages may have a somewhat more complicated form, like this:

```
SFGETI/SFGETR - PARAMETER NAME NOT KNOWN - X
```

What this particular error message says is that SFGETI called SFGETR, which detected an error condition (a bad parameter name) and called SETER. Upon getting control back from SFGETR, SFGETI detected the fact that SFGETR had logged an error. It augmented the error message by prepending its own name, followed by a slash, and then passed control back to the user. Of course, there can be more than two such levels of routine calls indicated in the error message: in a few cases, seven or eight routine names may be listed, each separated from the next by a slash.

The various error conditions in SOFTFILL are described in the list below. Each bulleted item includes an error message and a thumb-nail description of the error. The items in the list are arranged in alphabetical order. If you get an error message with one or more prefixed subroutine names, as described above, omit them and look up the result in this list. Note that, since SOFTFILL routines sometimes call other routines, elsewhere in NCAR Graphics, that can detect error conditions and call SETER, the error message you get by calling a SOFTFILL routine may not be listed here, but in the programmer document for some other package.

- **SFGETC - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to SFGETC is not the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFGETC - PARAMETER NAME TOO SHORT - X**

  The first argument in a call to SFGETC is too short to be the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFGETC - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFGETC was called, there was an unrecovered outstanding error. In this case, SFGETC cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFGETI - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFGETI was called, there was an unrecovered outstanding error. In this case, SFGETI cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFGETP - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFGETP was called, there was an unrecovered outstanding error. In this case, SFGETP cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFGETR - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to SFGETR is not the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFGETR - PARAMETER NAME TOO SHORT - X**

  The first argument in a call to SFGETR is too short to be the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFGETR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFGETR was called, there was an unrecovered outstanding error. In this case, SFGETR cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFNORM - ARRAY DST IS TOO SMALL**

  The indicated real scratch array is too smalL.

- **SFNORM - ARRAY IND IS TOO SMALL**

  The indicated integer scratch array is too small.

- **SFNORM - COORDINATE ARRAYS TOO SMALL**

  Thie error only occurs if the number of points defining a polygon to be filled is less than or equal to 2 - a totally avoidable error.

- **SFNORM - LOGIC ERROR - SEE SPECIALIST**

  This error should never occur; if it does, it indicates a compiler problem or some sort of tampering with the code of the package.

- **SFNORM - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFNORM was called, there was an unrecovered outstanding error. In this case, SFNORM cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFSETC - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to SFSETC is not the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFSETC - PARAMETER NAME TOO SHORT - X**

  The first argument in a call to SFSETC is too short to be the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFSETC - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFSETC was called, there was an unrecovered outstanding error. In this case, SFSETC cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFSETI - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFSETI was called, there was an unrecovered outstanding error. In this case, SFSETI cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFSETP - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFSETP was called, there was an unrecovered outstanding error. In this case, SFSETP cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFSETR - PARAMETER NAME NOT KNOWN - X**

The first argument in a call to SFSETR is not the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFSETR - PARAMETER NAME TOO SHORT - X**

  The first argument in a call to SFSETR is too short to be the name of one of the internal parameters of SOFTFILL. “X” is the offending name.

- **SFSETR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFSETR was called, there was an unrecovered outstanding error. In this case, SFSETR cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFSGFA - ERROR EXIT FROM GQFACI**

  The GKS routine GQFACI, which is called to get the current value of the fill area color index, has returned a non-zero error code.

- **SFSGFA - ERROR EXIT FROM GQPLCI**

  The GKS routine GQPLCI, which is called to get the current value of the polyline color index, has returned a non-zero error code.

- **SFSGFA - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFSGFA was called, there was an unrecovered outstanding error. In this case, SFSGFA cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **SFWRLD - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time SFWRLD was called, there was an unrecovered outstanding error. In this case, SFWRLD cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

5.0 ALGORITHM

The given points are sorted in order of increasing distance from a “baseline” having the desired fill angle. Then, a “fill line” is passed over the figure, starting at the minimum distance found and stopping at the maximum distance found. Each time the “fill line” passes one of the points, a list of the boundary lines which the fill line intersects is updated. At each desired drawing position, a list of intersection points is generated, sorted in the order of their projection on the x axis or the y axis (depending on the fill angle) and the segments of the fill line which fall inside the figure to be filled are drawn, using solid lines or dotted lines. In the latter case, individual dots are drawn or not drawn as specified by the current contents of an 8x8 array representing a square pixel; each “dot” may be just a dot or a selected symbol.

6.0 HISTORY

This code was written by Dave Kennison in April, 1982, using an algorithm of his invention rather like that used by Richard Helgason in a program for continent-coloring, called Sherwin-Williams, about 1969. It was greatly modified to become a part of NCAR Graphics in January, 1989.

7.0 TIMING

Roughly speaking, the timing of SFWRLD and SFNORM varies linearly with the number of line segments or dots produced. It is little affected by the number of points defining the area to be filled (except insofar as this results in an increase in the number of lines drawn).

On a Cray X-MP, filling a “circle” defined by 10 points and having a radius of .25 fractional coordinate units with lines at default spacing required .031 CPU seconds. Using dots to fill the circle boosted the time to .571 CPU seconds. Reducing the radius of the circle by a factor of 2 reduced the times for line fill and dot fill to .018 and .144 CPU seconds, respectively. Defining the circle with 100 points boosted the corresponding times to .036, .608, .019, and .155 CPU seconds.

8.0 MISCELLANY

All routines in the package are written in FORTRAN 77. Block names used (including subroutine names and labelled COMMON block names) are SFBLDA, SFCOMN, SFGETC, SFGETI, SFGETP, SFGETR, SFNORM, SFSETC, SFSETI, SFSETP, SFSETR, SFSGFA, SFSORT, and SFWRLD. The only I/O is the graphic output generated by calls to the SPPS routine PLOTIF. The FORTRAN library routines SIN and COS, NCAR Graphics routines CUFX, CUFY, GETSET, PLOTIF, POINTS, SET, and SETER, and GKS routines GFA, GQFACI, GQPLCI, GSFACI, and GSPLCI are called.

9.0 EXAMPLES

On a Unix system on which NCAR Graphics has been installed, the command “ncargex” may be used to acquire the code for and run two different examples for SOFTFILL. The names of the examples are “sfex01” and “sfex02”. The first of these shows how to use the basic routines SFWRLD and SFNORM and the parameter-access routines to produce various kinds of fill patterns, and the second shows how to use the routine SFSGFA.