Appendix A COHEN SUTHERLAND OUTCODE ALGORITHM

A popular method for clipping lines is the COHENSUTHERLAND OUTCODE ALGORITHM. The algorithm quickly removes lines which lie entirely to one side of the clipping region (both end points above, or below, or right or left). The algorithm makes clever use of bit operations(outcodes) to perform this test efficiently. Segment endpoint are each given 4 bit binary codes. The high order bit is set to 2 if the point is above the window; the next bit is set to 1 if the point is below the window; the third and fourth bits indicates right and left of the windows, respectively. The lines which form the window boundary divide the plane into nine regions with the outcodes.

If the line is entirely within the window, then both the endpoints will have outcodes 0000. Segments with this property are accepted. If the line segment lies entirely on one side of the window (say entirely above it), then both endpoints will have a 1 in the outcode bit position
for that side (the first bit will be 1 for both endpoints). We can check to see if the line is entirely on one side of the window by taking the logical AND of the outcodes for the two endpoints. If the result of the AND operation is nonzero, then the line segment may be rejected. Thus one test decides if the line segment is entirely above, or entirely below, or entirely to the right, or entirely to the left of the window.

The following is the brief outline of the algorithm: First, we compute the outcodes for the two endpoints (p1 and p2) of the segment. Next, we enter a loop. Within the loop we check to see if both outcodes are zero; if so we enter the segment into the display file, exit the loop, and return. If the outcodes are not both zero, then we perform the logical AND function and check for a nonzero result. If this test is nonzero, then we reject the line. Exit the loop and return. If neither of this test result is satisfied, we must subdivide the line segment and repeat the loop. If the outcodes for p1 is zero, exchange the points p1 and p2 and also their outcodes. Find a nonzero bit in the outcode of p1. If it is the high-order bit, then find the intersection of the line with the top boundary of the window. If it is the next bit position, then subdivide along the bottom boundary. The other two bits indicates that the right and left boundaries should be used.
Replace the points p1 with the intersection point and calculate its outside. Repeat the loop.

6.6 Algorithm CLIP-LEFT(OP,X,Y) Routine for clipping against the left boundary.

Arguments OP, X, Y the instruction to be transformed.

Global WXL window left boundary

XS, YS arrays containing the last point drawn

NEEDFIRST array of indicators for saving the first command

FIRSTOP, FIRSTX, FIRSTY arrays for saving the first command

CLOSING indicates the stage in polygon.

BEGIN

IF PFLAG AND NEEDFIRST[1] THEN

BEGIN

FIRSTOP[1] <- OP;

FIRSTX[1] <- X;

FIRSTY[1] <- Y;

NEEDFIRST[1] <- FALSE;

END;

Case of drawing from outside in

ELSE IF X >= WXL AND XS[1] < WXL THEN
Algorithm CLIP-RIGHT(OP,X,Y) Routine for clipping against the right boundary.
Arguments  OP, X, Y the instruction to be transformed.

Global  WXH window right boundary

XS, YS arrays containing the last point drawn

NEEDFIRST array of indicators for saving the first command

FIRSTOP, FIRSTX, FIRSTY arrays for saving the first command

CLOSING indicates the stage in polygon.

BEGIN

IF PFLAG AND NEEDFIRST[2] THEN

BEGIN


FIRSTX[2] <- X;

FIRSTY[2] <- Y;

NEEDFIRST[2] <- FALSE;

END;

Case of drawing from outside in

ELSE IF X <= WXH AND XS[2] > WXH THEN

CLIP-BOTTOM(1, WXH, (Y - YS[2])*(WXH - X) / (X -XS[2]) + Y)

ELSE IF X >= WXH AND XS[2] < WXH THEN

IF OP > 0 THEN
Algorithm CLIP-BOTTOM(OP,X,Y) Routine for clipping against the lower boundary.

Arguments OP, X, Y the instruction to be transformed.

Global WYL window lower boundary

XS, YS arrays containing the last point drawn
NEEDFIRST array of indicators for saving the first command

FIRSTOP, FIRSTX, FIRSTY arrays for saving the first command

CLOSING indicates the stage in polygon.

BEGIN

IF PFLAG AND NEEDFIRST[3] THEN

BEGIN

FIRSTOP[3] <- OP;
FIRSTX[3] <- X;
FIRSTY[3] <- Y;
NEEDFIRST[3] <- FALSE;

END;

END;

Case of drawing from outside in

ELSE IF Y >= WYL AND YS[3] < WYL THEN

CLIP-TOP(1, (X - XS[3])*(WYL - Y) / (Y - YS[3]) + X,WYL)

ELSE IF Y <= WYL AND YS[3] > WYL THEN

IF OP > 0 THEN

CLIP-TOP(OP,(X - XS[3])*(WYL - Y)/(Y - YS[3]) + X,WYL)

ELSE
Algorithm CLIP-TOP(OP,X,Y) Routine for clipping against the upper boundary.

Arguments  OP, X, Y the instruction to be transformed.

Global  WYH window upper boundary

XS, YS arrays containing the last point drawn

NEEDFIRST array of indicators for saving the first command
FIRSTOP, FIRSTX, FIRSTY arrays for saving the first command

CLOSING indicates the stage in polygon.

BEGIN

IF PFLAG AND NEEDFIRST[4] THEN

BEGIN

FIRSTX[4] <- X;

END;

Case of drawing from outside in

ELSE IF Y <= WYH AND YS[4] > WYH THEN

SAVE-CLIPPED-POINT(1, (X - XS[4])*(WYH - Y) / (Y - YS[4]) + X, WYH)

ELSE IF Y >= WYH AND YS[4] < WYH THEN

IF OP > 0 THEN

SAVE-CLIPPED-POINT(OP, (X - XS[4])*(WYH - Y) / (Y - YS[4]) + X, WYH)

ELSE

...
SAVE-CLIPPED-POINT(1,(X - XS[4])*(WYH - Y)/
(Y - YS[4]) + X,WYH);

Remember point to serve as one of the endpoints of next line segment

XS[4] <- X;

Case of point inside

IF Y <= WYH AND CLOSING <> 4 THEN SAVE-CLIPPED-POINT(OP,X,Y);

RETURN;

END;

THE SUTHERLAND - HODGMAN ALGORITHM

The Cohen-Sutherland algorithm works well for lines, but we would like a method which may be used with polygons as well. Our clipping routines will be based on a method discovered by Sutherland and Hodgman. The method unbuldles the clipping test to clip against each of the four boundaries individually. The idea behind the algorithm is that we can easily clip a line segment against any one of the window
boundaries. We can then perform the complete clipping by clipping against each of the four boundaries in turn.

To clip at a boundary, we step through the drawing instruction. As we consider each new endpoint, we decide whether it belongs to a line which crosses the boundary. If it does, the point of intersection is determined and is passed on to the next routine. Then each point is examined to see whether it lies within the boundary. If so, it is also passed to the next routine. In this procedure, all line segment endpoints lying within the boundary and all points where lines intersects the boundary are passed on, while points lying outside the boundary are filtered out.

We can think of the process as clipping the entire figure against each window boundary before moving on to the next boundary. However since our clipping process steps sequentially through the figure drawing instructions, it is possible to begin clipping on a second boundary before the clipping of the entire figure against the first boundary is completed. In fact each point may be run through all four clipping routines and entered into the display files before the next point is considered.
Algorithms for clipping a figure against each of the four window boundaries are given below. They all follow the same outline. They first check to see if the new point is the first point of a polygon, and if so, they save it. This is used in closing polygons and is discussed below. They examine the new point and the last point to see whether the line segment with these endpoints crosses the boundary. The algorithms are called for each new point. We can picture as this pen movements. We start with with the pen at some location (the last point which is stored for each clipping boundary in the arrays and ask to move it to some new position. The clipping routine examines this path to see if it encounters the clipping boundary. If it does, the pen is moved only to the boundary; a new command, corresponding to the clipped point, is entered. If the side is drawn from outside the window to inside the window or the command is for character drawing, then we introduce a MOVE command; otherwise the command is the same as the original. This means that if our figure passes outside of the window boundary, the pen will move along the boundary to the point where the figure reenters the window region. The algorithm update the last point to be the current point and check the current point to see whether it is inside the window. If so, this instruction is also entered. When we say a command is entered, we mean that it passed on to the next routine. For the first three clipping algorithm,