The polygon package

E E Barton*and I Buchanan

The polygon package is a set of procedures which manipulate geometric objects in the 2D plane. The operations that can be performed on these objects, regarded as polygons, include intersection, union, asymmetric difference, inflation and deflation. Polygons may have both straight and circular arc edges and include holes within their boundaries. Using a binary tree of bounding boxes as a sorting aid in the centrally important procedure for calculating edge intersections results in a fast, efficient package.

A polygon package may be used in any program that requires operations such as union, intersection, difference, inflation and deflation. Examples of these operations are shown in Figure 1. Many applications of such a package exist in the fields of, computer graphics, architecture, engineering and geographical data processing. The following examples illustrate two of the areas in which the package may be used.

One of the computationally most expensive operations in computer graphics is the removal of hidden lines or surfaces in 3D images. The polygon package may be used to remove hidden regions provided the surfaces do not intersect[1,2]. Relevant surfaces are projected onto the viewplane where they may be treated as polygons. By sorting these polygons on their distance from the view point and using the difference operator hidden regions may be removed.

In VLSI circuit fabrication, the masks used to expose selective portions of resist on the silicon substrate are subject to certain dimensional design rules. These rules state that certain regions in the mask may not be closer than a defined distance, while other may not be less than a minimum size. Manual checking for violations of these rules is a time consuming, expensive and unreliable procedure. If the mask, or set of masks, is passed to the polygon package as a collection of polygons, the proximity of a set of regions to each other may be quickly and accurately tested by inflating those regions by half the test distance and calculating the intersection. If such intersecting areas are found then the design rule has been violated at precisely those locations.

Similar techniques are applicable in architecture and various engineering disciplines where structures may be subject to minimum and maximum size requirements.

DEFINITIONS

The generality and correctness of the definition of a polygon is the foundation on which the package builds the set of manipulation procedures. The set of polygons, corresponding to the definition below, must be closed under the operations union, intersection, difference, inflation and deflation. Briefly, a *polygon* consists of a number of *sheets*, each of which may have a positive or negative sense[3]. A sheet is a sequence of directed *edges*, which may be straight lines or circular arcs, connecting at *vertices*. Every point in the plane has an associated *wrap number* which is a measure of its level of enclosure by a set of sheets (Figure 2).

Detailed definitions follow:

- An *edge* is a straight line or a circular arc connecting two vertices. The edge possesses direction from its tail vertex to its head vertex. The tail vertex is omitted from the set of points on the edge. Circular arc edges are represented by their centre point and radius. The radius is positive if the arc is described anti-clockwise from tail to head and negative if it is described clockwise.
- A *vertex* is a point on the 2D coordinate plane at which two edges connect. It is the head vertex of its preceding edge and the tail vertex of its succeeding edge.

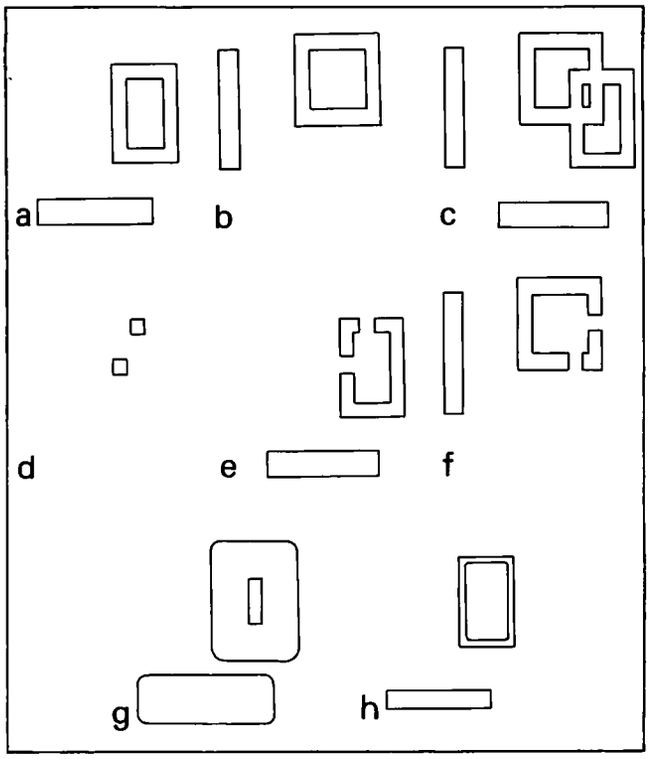

*Figure 1. (a) polygon A (b) polygon B (c) union, A+B (d) intersection, A*B (e) difference, A–B (f) difference, B–A (g) inflation A (h) deflation A.*

Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ.

*Now at Department of Computer Science, California Institute of Technology, USA.

 0010–4485/80/010003–09 $02.00

- A *sheet* is a set of points enclosed by an ordered ring of alternating vertices and edges. The edges may not exhibit any self intersections. The sense of the sheet is either clockwise (negative) or anti-clockwise (positive) depending on the sense of the cyclic order of the edges and vertices.
- The *wrap number* of a point is a measure of the number of times the point is encircled by a set of sheets. This number is positive if the point is enclosed by a net number of anti-clockwise sheets. It is negative if the point is enclosed by a net number of clockwise sheets. If the point lies on the edge of a negative sheet then it is considered to be outside the sheet and the sheet makes no contribution to the wrap number of the point. If the point lies on the edge of a positive sheet then the point is considered to be inside the sheet and the wrap number is incremented.
- A *polygon* is a set of associated sheets such that none of the sheets exhibit edge intersections with any of the others and, furthermore, relative to this set of sheets all points in the plane are either inside the polygon and have wrap number +1 or are outside the polygon and have wrap number 0, ie no sheet can be enclosed by another of the same sense and no negative sheet can exist outside a positive sheet.

All well-formed polygons conform to the above definition.

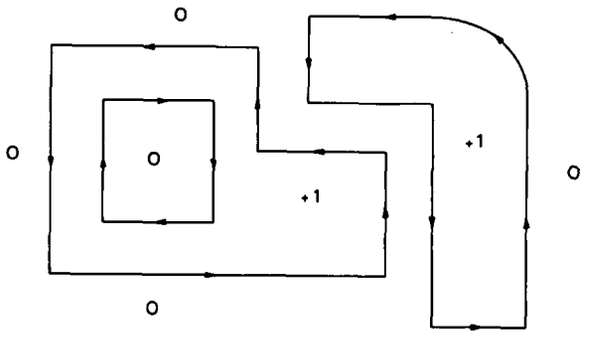

Figure 2. Wrap numbers are associated with each point in the plane

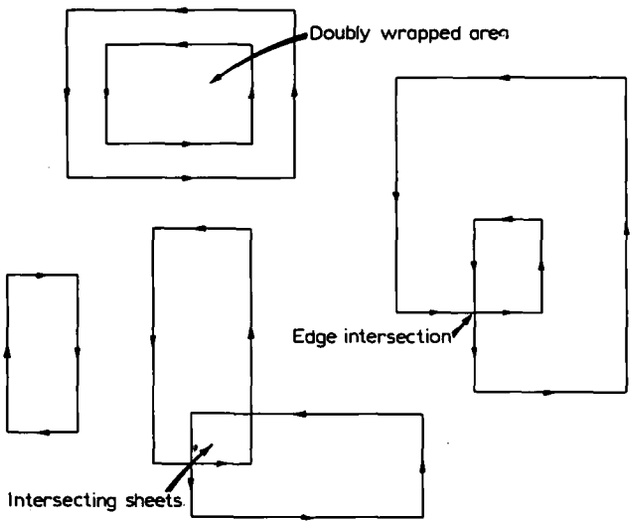

Figure 3. Malformed polygons

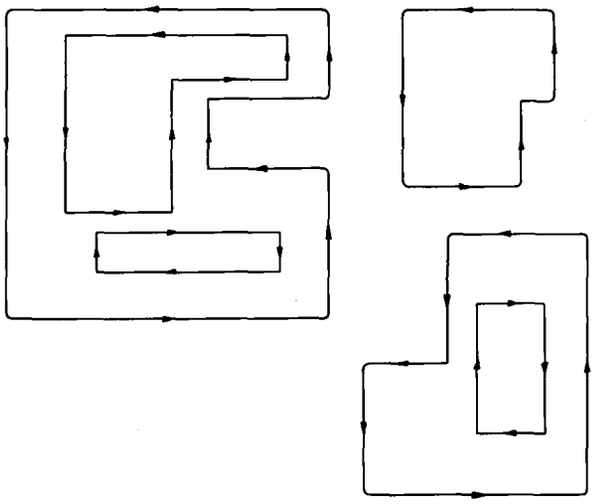

Figure 4. Well-formed polygons

Polygons which violate any of the rules are termed malformed and cannot be reliably manipulated by the union, intersection, difference, inflation and deflation procedures unless they are first processed by the package to remove the irregularities. Examples of well-formed and malformed polygons are shown in Figures 3 and 4.

PRIMITIVE OPERATIONS

High-level polygon manipulation procedures depend on several lower level operations. Most basic of these is the edge intersection routine. On this may be built the calculation of the wrap number which is the basis for deducing the topology of any polygonal interaction. Use of a sorting aid increases the speed of these primitive operations by reducing the number of calculations to be performed. The technique used in this package is a bounding box based binary tree[4].

Bounding boxes

Manipulating polygons ultimately reduces to calculations on pairs of edges to discover the existence and locations of any intersections. Testing each edge in one polygon against every edge in another yields an algorithm of $O(n^2)$ complexity. For large polygons with few intersections this is clearly unacceptable. It is necessary to develop some method of disregarding unproductive pairs of edges. The technique used in this package is based on the paraxial bounding box (Figure 5), a simply described rectangle which encloses all the points on an edge and within a surrounding rectangular region.

To achieve an algorithm with the desired performance it is necessary to merge the primitive bounding boxes so that sets of edges may be compared (Figure 6).

The necessary hierarchy is realised in a binary tree structure in which the root is the bounding box of the complete polygon and the leaves are the bounding boxes of the individual edges. The tree is built up by merging the primitive bounding boxes according to the cyclic order of the edges in the sheets. Sheets are sorted by proximity and merged accordingly until the root of the tree, the polygon, is reached and is enclosed in a single bounding box. Below sheet level the tree building is performed by

an algorithm of $O(n)$ complexity, in fact in $(2n-1)$ operations. Above that level the sorting of sheets imposes an $O(n \log n)$ complexity.

EDGE INTERSECTIONS

The bounding box hierarchy is used to concentrate on only those areas where intersections are likely to occur. At each call of the edge intersection procedure two bounding boxes are referenced and compared for a possible intersection. Initially, these are the bounding boxes of the two polygons. No intersection between bounding boxes implies no intersection between any of the edges they contain and therefore the search down these branches of the two trees can be terminated. If an intersection does occur then one of the bounding boxes, that containing the greater number of edges, is divided into its two successors. The algorithm is then applied recursively to each new pair of bounding boxes. Once two primitive bounding boxes have been reached an edge intersection is possible and the appropriate calculation is performed. Figure 7 shows a pair of intersecting polygons and the bounding boxes examined by the searching algorithm.

The number of bounding boxes tested by the algorithm is greatest near the edge intersection points, thus demonstrating the efficiency of the search method.

Distinction must be made between three types of interactions by the edge intersection procedure:

- straight edge and straight edge
- straight edge and circular arc
- circular arc and circular arc.

Calculations involving circular arc edges are more complex than their straight edge counterparts. However, edge intersection calculations occur only when strictly necessary, so this makes no appreciable difference to the overall performance of the package while considerably enhancing its functionality.

Wrap number

The wrap number of a point determines its level of nesting within a set of sheets. A line is constructed from the test point rightwards for an infinite distance. The number of intersections between this line and the edges of the polygon are counted and the sense of the intersection determines whether the count is incremented or decremented. Again, the bounding box hierarchy produces great savings in computation by restricting the number of edge intersection calculations to be performed. The procedure calls itself recursively and at each level it references a bounding box and the test point. Bounding boxes are trivially

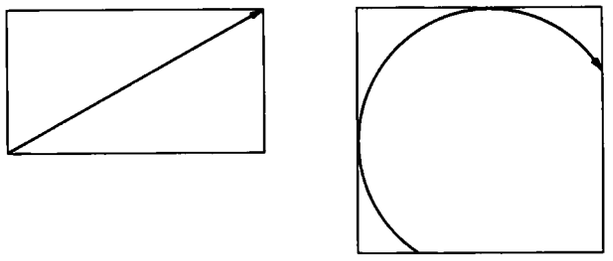

Figure 5. Paraxial bounding boxes

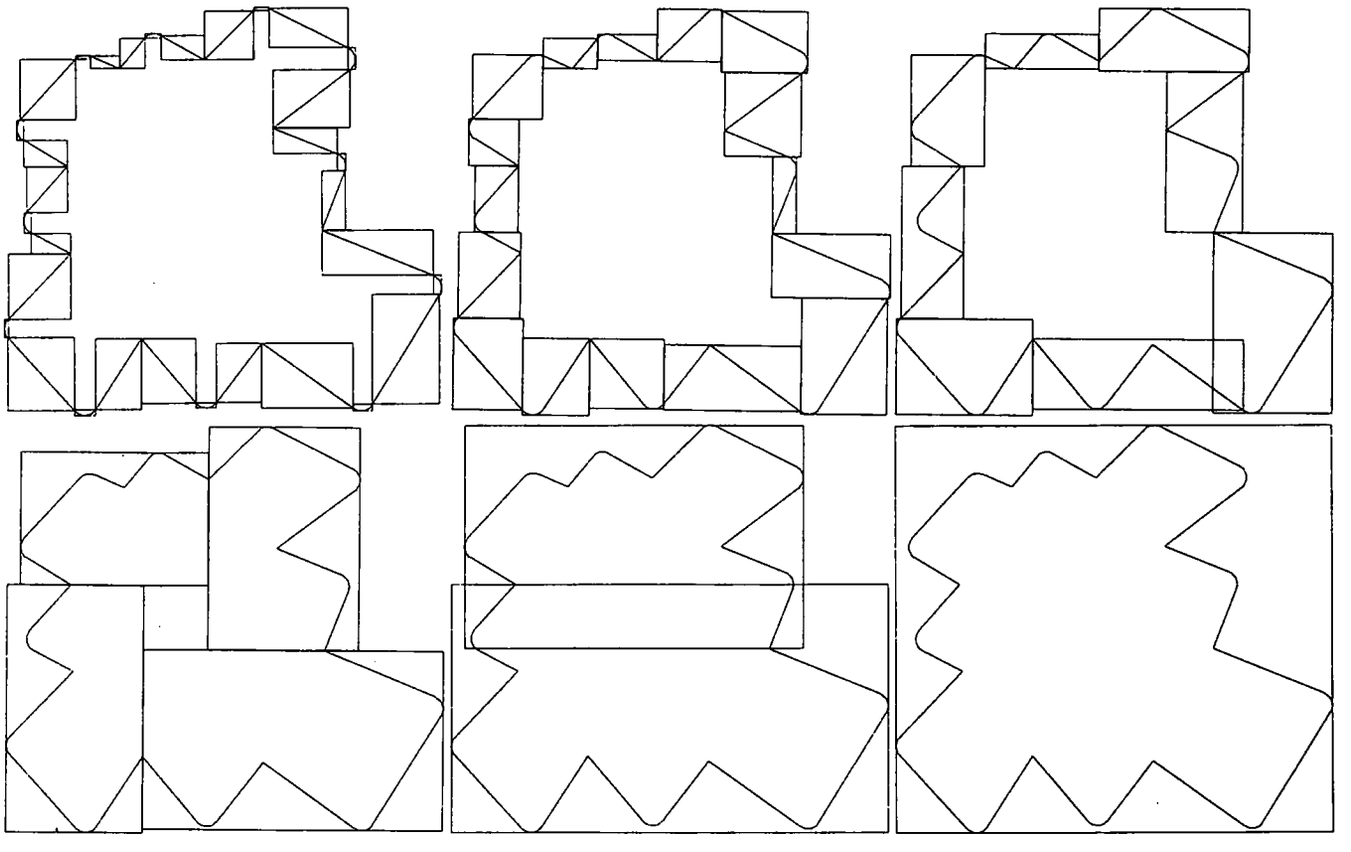

Figure 6. Merging of bounding boxes

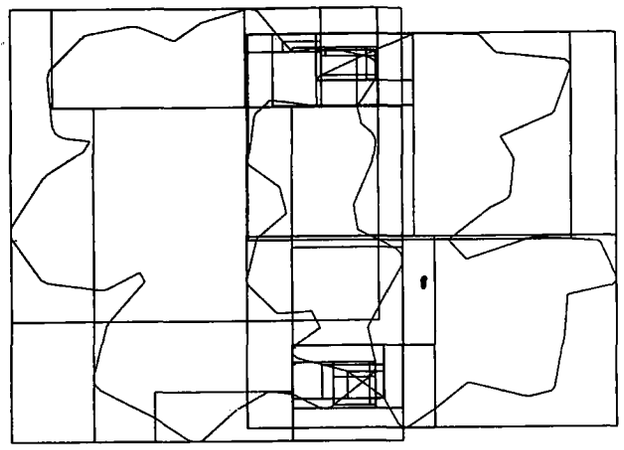

Figure 7. Intersecting pair of polygons with bounding boxes

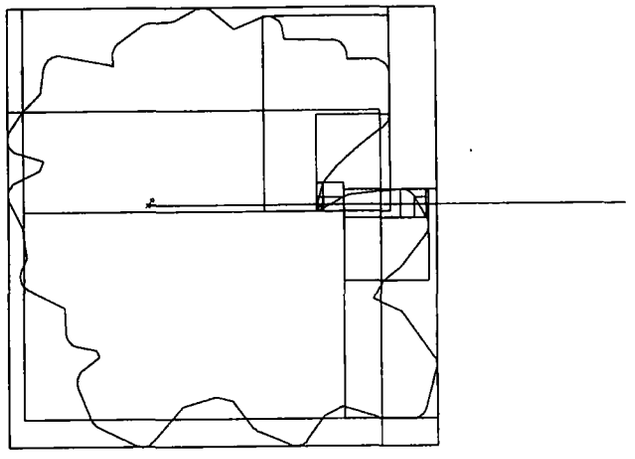

Figure 8. Bounding boxes used in wrap number calculation

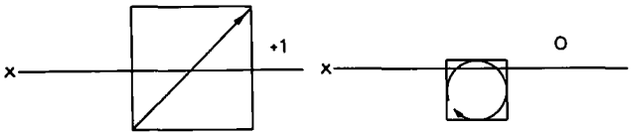

Figure 9. Contributions to wrap numbers

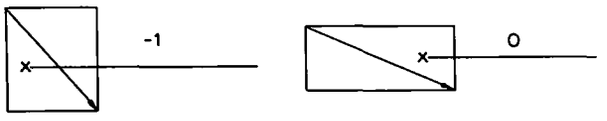

Figure 10. Testing of point position

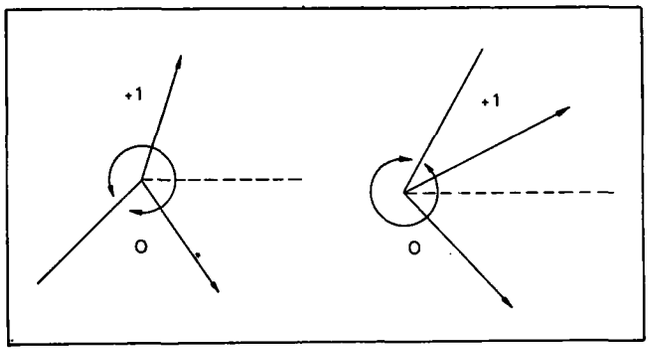

Figure 11. Affect of next edge direction wrap number

rejected if they lie completely to the left, above or below the constructed test line. A bounding box intersected by the line is divided into its two successors and the algorithm is applied to both, recursively, until a primitive bounding box is reached (Figure 8).

At this stage the normal edge intersection calculation is performed and several alternative cases are recognised:

- The test point lies to the left of the bounding box.
  The test line crosses the edge, if and only if one of its vertices lies on or above the line and the other lies below the line (Figure 9). Circular arcs therefore make no contribution to the wrap number if both their vertices lie to one side of the line, as their crossings are in opposite directions and cancel each other out (Figure 9). The wrap number is incremented for edges crossing in an anti-clockwise direction, ie the tail vertex lies below the head vertex, and decremented for edges crossing in a clockwise direction.

- The test point lies within the bounding box.

  o The edge within the bounding box is a straight line.
  The position of the point relative to this edge is tested. If the point lies to the left of the edge, the wrap number is incremented or decremented as in Figure 10. If the point lies to the right of the edge, no crossing of the test line has occurred and no contribution to the wrap number is made (Figure 10).

  If the point lies on the edge then either the edge points upwards or it points downwards; horizontal edges are classed as pointing upwards if they point to the right and downwards if they point to the left. The wrap number is incremented for all points on upward pointing edges, excluding the tail and head vertices. A point on the head of an upward pointing edge increments the wrap number if the next edge points upwards or to the left of the edge (Figure 11). The wrap number is not affected by downward pointing edges unless the point is on the head vertex and the next edge points upwards and to the right of the edge (Figure 11).

  o The edge within the bounding box is a circular arc.
  Tests are made to discover whether or not the point lies on the arc and also how many times the test line crosses the arc. If the arc is crossed twice no net change occurs to the wrap number as the directions of the crossings are opposite (Figure 12). If the line crosses the arc only once then the wrap number is incremented or decremented according to the direction of the arc at the crossing point (Figure 12).

  Once again, particular attention must be paid to points on the edge and its two endpoints. If the test point lies on the arc but not at a vertex then the wrap number is incremented only for arcs of positive radius (Figure 13). If the test point lies on the tail vertex (Figure 13) then no change is made to the wrap number for arcs of positive or negative radius unless the test line intersects with another portion of the same arc.

  If the test point lies on the head vertex then a similar series of further tests is required as for head vertices of straight edges. If the tangent to the arc at the head vertex points upwards and the next edge

points upwards or to the left of the arc (Figure 14), or the tangent to the arc at the head vertex points downwards and the next edge points upwards and to

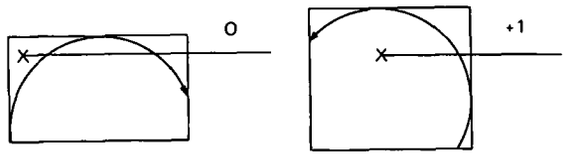

Figure 12. Arc direction at crossing point

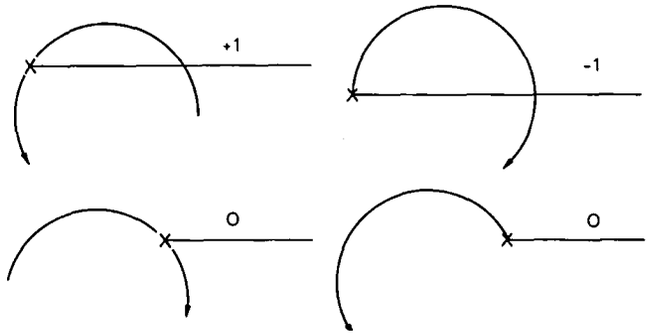

Figure 13. Position of test point and wrap number

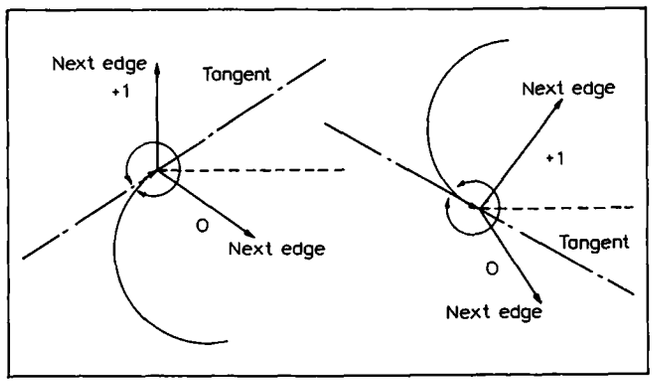

Figure 14. Affect of next edge direction on wrap number

the right of the arc (Figure 14) then the wrap number is incremented.

This set of tests is consistent for all well-formed polygons. More tests are performed when the test point lies on a vertex as more information is needed to decide whether the point forms an upper or lower extremity and, if so, in a clockwise or anti-clockwise sheet.

POLYGON OPERATIONS

Polygon manipulation operations that the package provides fall into two categories; unary (inflation and deflation) and binary (intersection, union and difference). All operations result in a single well-formed polygon. In addition three other unary operations (self-union, overlap and underlap) are supplied to discover and repair malformed polygons. Self union trims off any malformed regions of the polygon, overlap extracts all areas where the polygon intersects itself and underlap finds all the regions that are encircled a net negative number of times by the edges of the polygon.

Binary operations

Let wrap (r,R) be a function which returns the wrap number of a point r with respect to a polygon R. Then for a given point p in polygon P, wrap (p,P)=1. Thus for polygons P and Q, p is in union P+Q if, and only if wrap (p,P)=1 *or* wrap (p,Q)=1. Similarly p is in intersection P*Q if, and only if wrap (p,P)=1 *and* wrap(p,Q)=0. Figure 15 shows the results of these operations on an interaction between two polygons, one with two curved sheets and the other with a single straight-edged sheet. In each case the result of the operation has been slightly inflated or deflated to make its position obvious.

Clearly it is necessary to consider only points on the edges of sheets when manipulating the polygons within the package. Points of interest can be limited even further to the edge intersections. Accordingly, an algorithm has been developed which constructs a new polygon by tracing all the edges with the appropriate wrap number.

To calculate the result of a set operation on two polygons the algorithm starts by finding all edge intersections. Not all of these intersections will be useful to the tracing routines. The following tests select only those intersections which are relevant and will form decision points during the tracing of the output polygon.

- The intersection lies on the tail vertex of one or both of the two edges.
  The intersection is always ignored as the tail vertex of an edge is excluded from the edge for the purpose of finding edge intersections since the tail vertex of one edge forms the head vertex of the preceding edge.

- The intersection lies on the head vertices of both the edges.

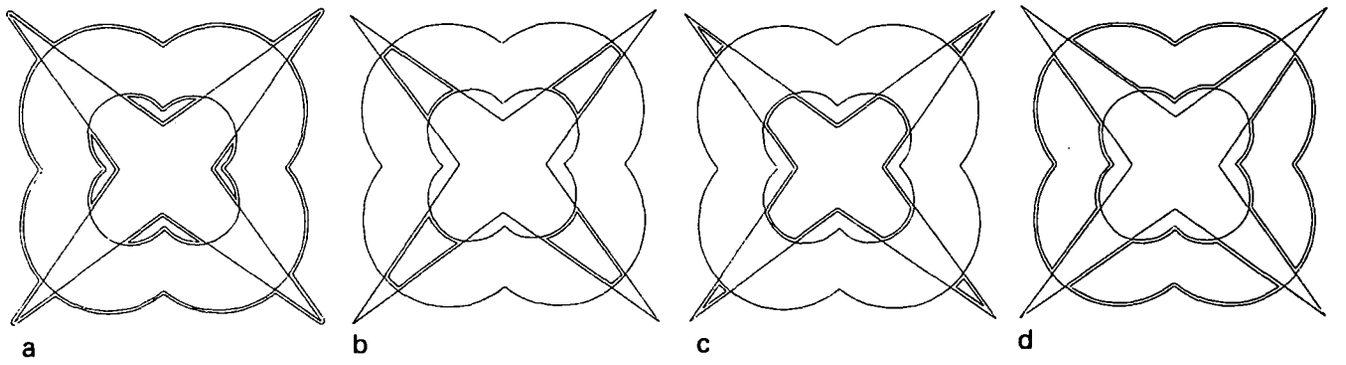

*Figure 15. (a) union P+Q (b) intersection P*Q (c) difference P—Q (d) difference Q—P*

The intersection is ignored if either the two edges are parallel and their successors are parallel, or if both edges are antiparallel to the other edge's successor (Figure 16).

- The intersection occurs on the head of one edge. If the successor of the edge with the intersecting head is parallel to the other edge it is ignored (Figure 16).

In effect, parallel edges, either singly or in groups, will yield a single intersection where they diverge.

Edge intersection routines build the data structure required by the tracing routines. The tracing routines must now examine the valid intersections and decide which paths to follow round the initial polygons to produce the desired binary operation result. The same procedure, parameterised by the appropriate wrap number, is used to produce the union, intersection or difference. The process begins by passing the first valid edge intersection to the tracing routine as the initial vertex of a sheet.

The step that selects which edge to follow makes the decision with reference to a number of conditions. If nothing is known about the current edge intersection, eg on first entering the procedure, then a test must be made of the wrap number of a pair of points on the edges about to be traced (Figure 17).

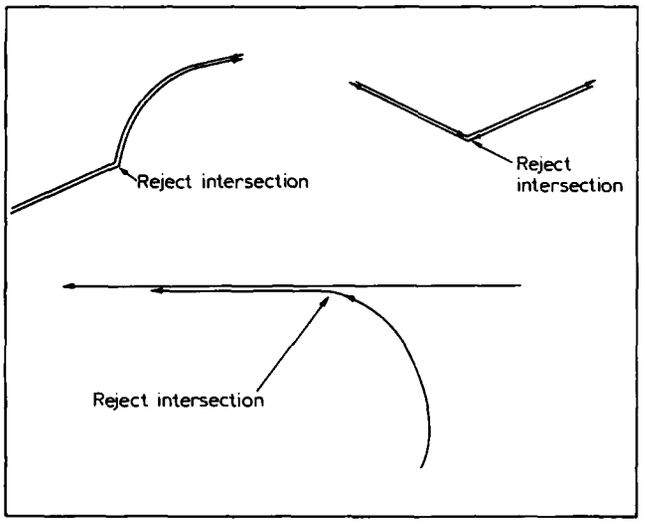

Figure 16. Cases where intersection ignored

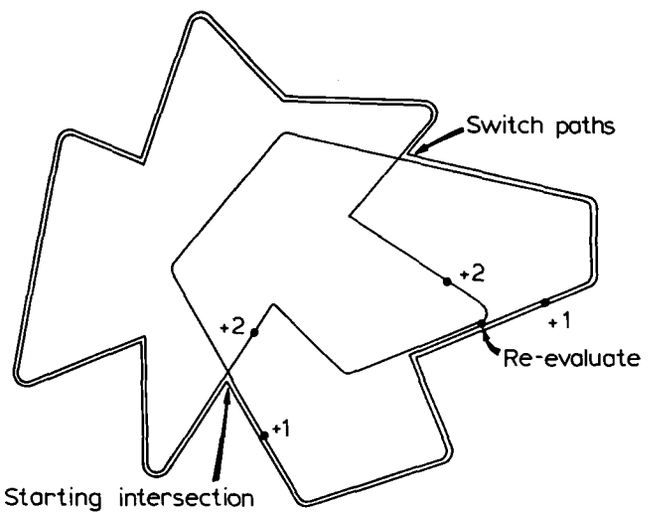

Figure 17. Test of wrap number

These wrap numbers are compared with the desired w number with the following possible outcomes.

- Neither wrap number equals the parameter. The intersection point must lie on some malformed region of one or both of the two polygons. Such a configuration should never arise if the polygons passe to the package are well-formed. Any edges added to the output polygon by the tracing procedure up to this point are dissembled and the point is marked as having been analysed.

- One of the points has a wrap number equal to the parameter. The edge with the matching wrap number is selected as the path on which tracing will be continued. The intersection is marked as having been analysed.

- Both points have wrap numbers equal to the paramet This case occurs when two edges which will appear in the output polygon touch, but do not intersect, one another. The point is marked to indicate the edge on which tracing will be continued so that the next time this point is reached the other edge will be selected immediately without recalculation of the wrap numbers. If the procedure is generating the intersection of two polygons then tracing will be continued on the current path. If the union or difference is being traced then the path is switched.

The procedure continues by copying edges and vertices the output sheet from the selected path until another intersection point is reached. If the current and intersecting edges are continuous over the intersection point there is no need to repeat the above tests and the tracin path is switched to the other polygon (Figure 17). The intersection point is then marked as having been analyse

The procedure continues to add edges to the output sheet until the initial intersection is rediscovered. If onl a single valid path exists from this point then the output sheet is closed. However, if both paths from the intersection were valid and only one has been traced, then the current sheet may be continued. Tracing is continued for the following reasons.

- Intersection is being constructed and the starting poi is not approached on the path that was initially selec (Figure 18).

- Union or difference is being constructed and the star point is approached on the initially selected path (Figure 18).

In all other cases the output sheet is closed (Figure 19).

When a new sheet has just been closed the next inter section that has either not been analysed or possesses a remaining untraced path is passed to the tracing procedure as an initial vertex. Sheets are added to the outp polygon until all intersections have been analysed.

Any sheets in the two initial polygons which did not exhibit edge intersections are now tested. They are incl in the output polygon if the wrap number of a point on one of their edges, calculated relative to both polygons, equals the desired wrap number. This completes the cor struction of the output polygon.

Inflation and deflation

Inflating a polygon expands all the anticlockwise sheets and contracts all the clockwise sheets by a given distance. Conversely, deflating a polygon contracts all the anticlockwise sheets and expands the clockwise sheets by a given distance. These two operations are implemented in the same procedure which takes a positive or negative parameter for inflation and deflation respectively (Figure 20).

A strict definition of the output polygon places the edges of the output polygon exactly the given distance from the edges of the input polygon. However, some algorithms have been content merely to shift the edges of the input polygon outwards or inwards by the desired distance without regard to the discrepancies that can occur at the vertices of the output polygon. This course of action has been followed by packages which do not include edges modelled by circular arcs. Some simplifications in code and data can be obtained by this view of the problem but at a cost in accuracy and functionality.

Expansion of a vertex on an anticlockwise sheet, whose edges form an internal angle of less than 180 degrees, should produce a circular arc centred on the original vertex with radius equal to the inflation distance (Figure 21). An algorithm that acts only on straight lines approximates this by extending the two edges until they intersect. If the edges are almost parallel but in opposite directions their endpoints will be extended by a distance many times greater than the inflation distance (Figure 21). Such inaccuracies are unacceptable in many applications.

The polygon package uses a single algorithm to produce both inflation and deflation of sheets. Every edge in the input sheet is shifted outwards or inwards by the appropriate distance. Straight edges are translated along their unit normal vertors and circular arcs have their radii increased or decreased by the given distance. Each vertex in the input sheet is the source of a new circular arc in the output sheet centred on that vertex. This process produces a curlicue at each vertex (Figure 22). The curlicues may cause malformed regions in the output sheet if the edges become intersecting. Such regions are removed by repair

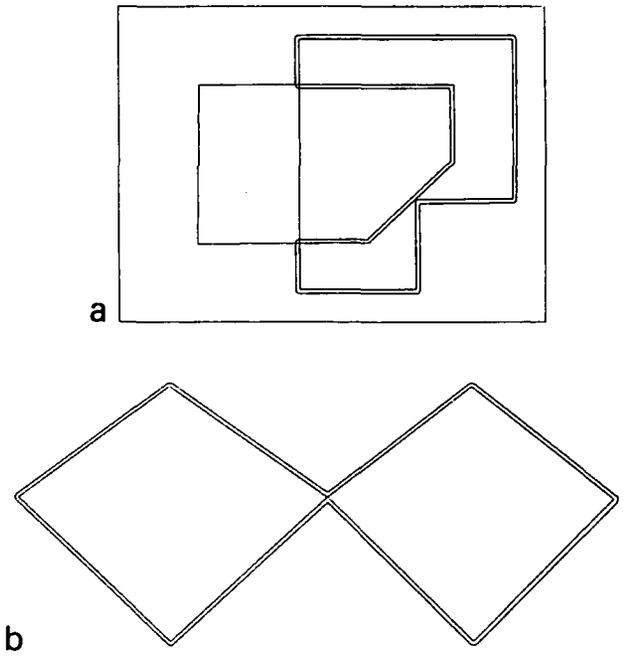

Figure 18. (a) starting point not *approached on initial path, (b) starting point approached on initial path*

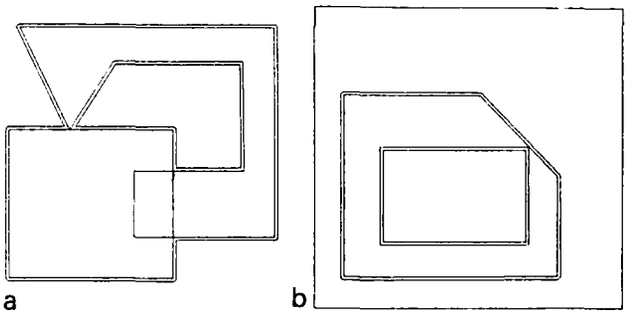

Figure 19. (a) union, (b) intersection

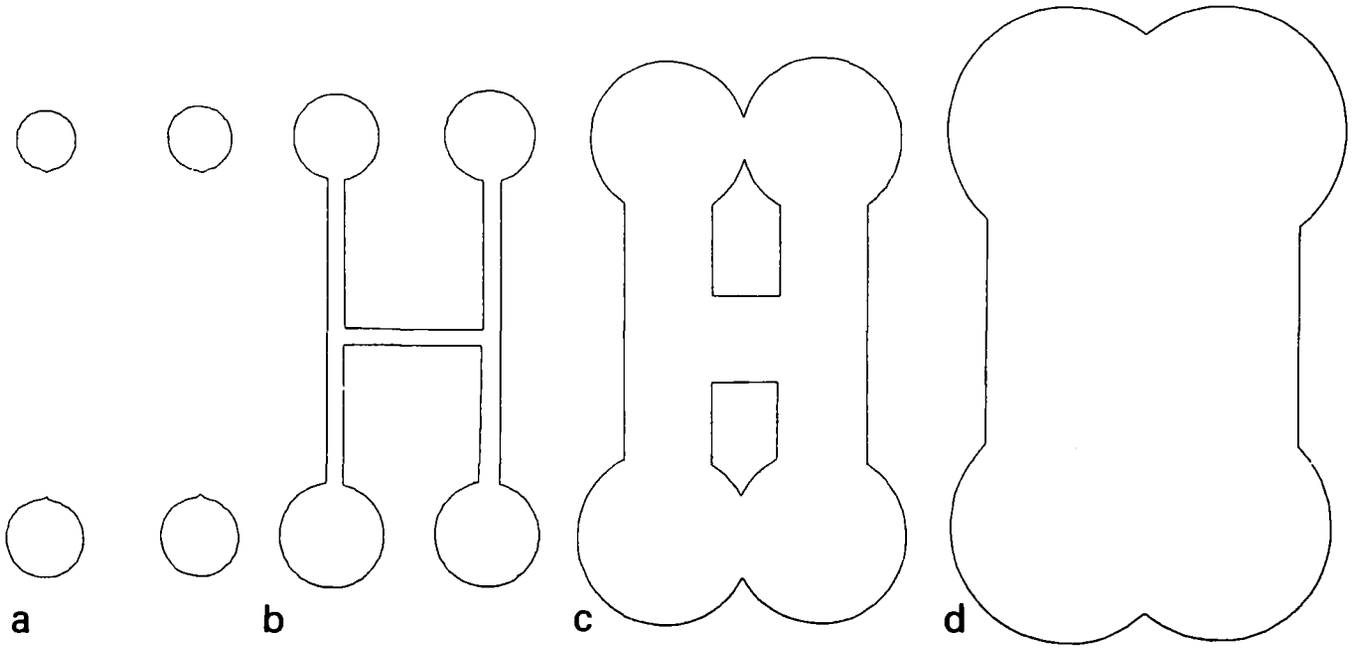

Figure 20. (a) deflated, (b) original, (c) inflated, (d) inflated

procedures included in the package. Output includes only those edges in the output sheet which have wrap numbers equal to that of the edges in the input sheet. Sheets which have been transformed so far as to reverse the sense of the sheet are discarded (Figure 22).

While the individual sheets of the output polygon are well-formed the polygon itself may be malformed. This occurs when two adjacent sheets have been inflated to such an extent that their boundaries now cross. A well-formed polygon may be produced in the usual way by the repair procedures but this is not done automatically since the user may require the data in the pre-repair format, possibly so that some check on overlap areas can be performed.

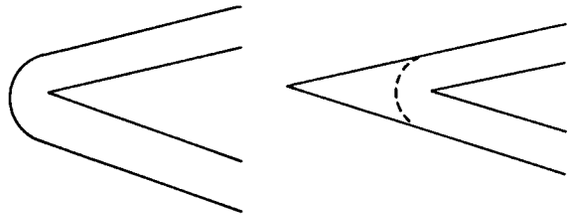

Figure 21. (a) radius equal to inflation distance, (b) radius much greater than inflation distance

Unary operations

The definition of the polygon has so far provided a set of rules within which the operations of union, intersection, difference, inflation and deflation have been implemented. These operations must be guaranteed to produce correct output polygons from well-formed input data. Polygons passed to the package could be malformed and it is necessary to provide a checking procedure so that a user may have confidence in the package. Additionally, such checks are of use internally, eg following an inflation which may cause self-overlap. Malformations are detected by the presence of edge intersections combined with topological tests. A number of configurations can cause the polygon to be malformed and a number of these are shown in Figure 23. Any of these conditions will cause the result *false* to be returned by the checking procedure. Self-union and self-intersection procedures repair the malformed polygons by tracing the singly and doubly wrapped regions respectively. Any holes in free space are discovered by following edges with wrap number 0.

Implementation of the malformation detector and the repair procedures rely heavily on the intersection and tracing routines already described. In the binary operations the tracing routine is used to discover all the sheets possessing edges with the required wrap number. Exactly this operation is performed on single polygons in the unary oper-

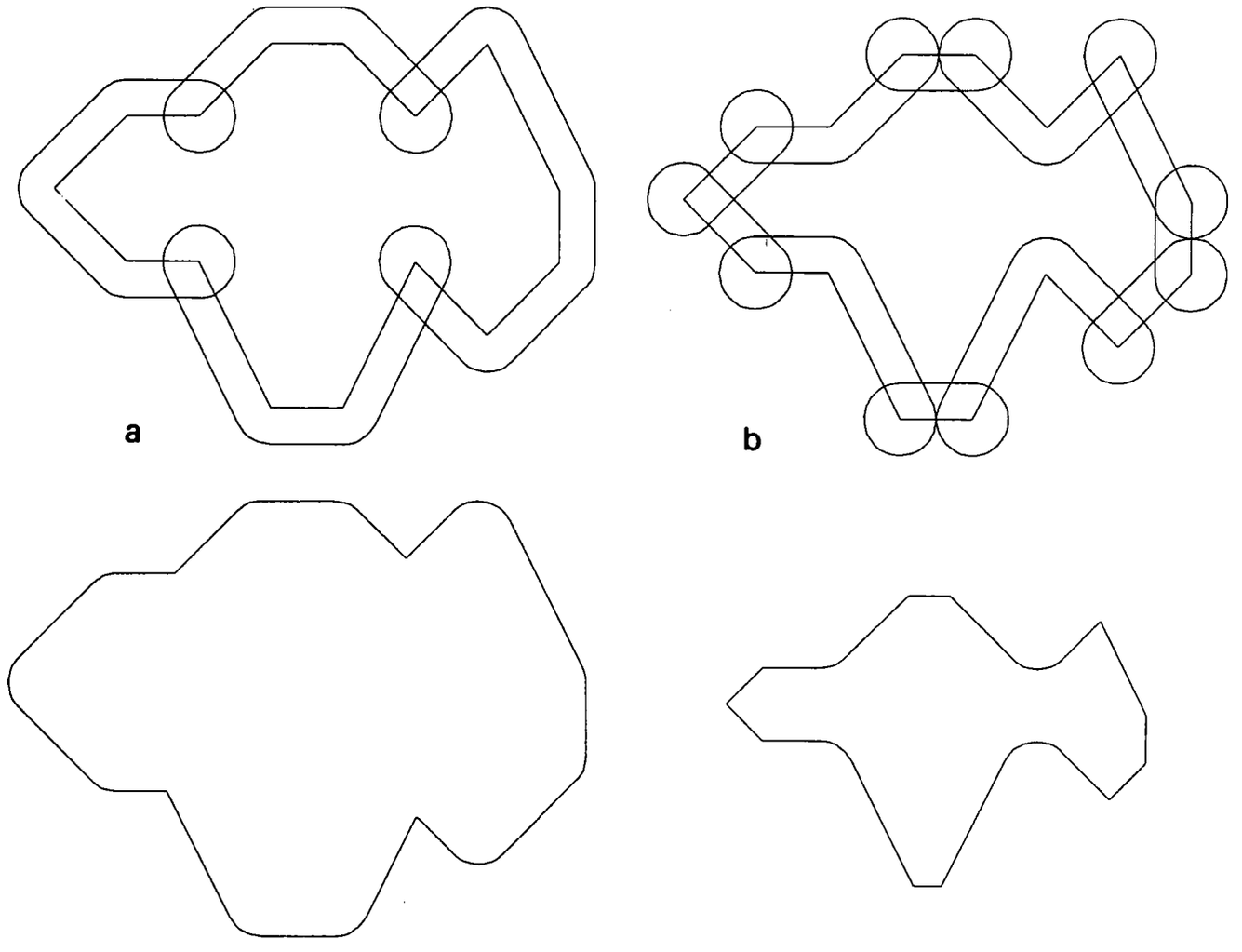

Figure 22. (a) inflation, (b) deflation

ations. The difference lies in constructing the intersection lists which the tracing procedure uses.

All the edge intersections of the input polygon must be found before tracing can commence. The usual edge intersection procedure is used to find these intersections but it is only efficient when finding the intersections of groups of edges. A second procedure is required to present a series of groups of edges to the intersection routine in such a manner that, eventually, all the edges have been compared for intersections with all other edges. The bounding box trees consist of groups of edges. Every edge in the polygon will be implicitly compared with all other edges if the intersection procedure is called with the two sons of every element in the tree, excluding the leaves. This has been implemented in a recursive procedure which is initialized with the bounding box of the whole polygon. It calls the intersection routine with the two sons of the current bounding box and then recurses on these. The routine returns immediately if the bounding box is primitive. If there are n edges in the polygon there will be $(2n-1)$ elements in the tree and this procedure will perform $(n-1)$ calls of the edge intersection procedure. Thereafter the tracing procedure will be called upon to retrieve the negatively, singly and doubly wrapped regions with a wrap number parameter of 0, 1 and 2 respectively (Figure 23). The output polygon is well-formed and can be used in further polygon package operations.

Area and perimeter

Two further procedures are provided, area and perimeter, both of which return real number results. The area of a polygon is the total area of all its sheets, remembering that negative sheets have negative area. A polygon may not however, have a negative area since a negative sheet may not exist outside a positive sheet.

The perimeter of a well-formed polygon is simply the total absolute length of all its edges, ignoring direction and sheet nesting depth.

CONCLUSIONS

The polygon package provides a powerful and efficient tool which may be employed in a variety of application areas. Within the stated definitions, the set of polygons is closed under the operations of intersection, union, difference, inflation and deflation supplied by the polygon package. The bounding box hierarchy enables groups of edges to be compared for intersection, and is effective in reducing the average number of edge intersection calculations below the upper bound of $0(n^2)$ complexity. The inclusion of

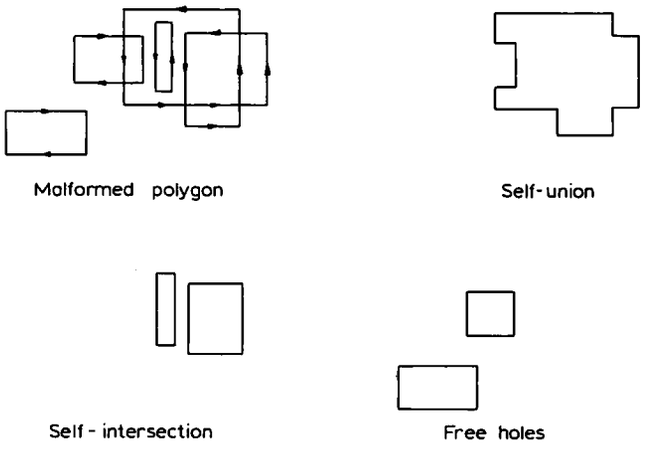

Figure 23. (a) malformed polygon, (b) self-union, (c) self-intersection, (d) free holes

circular arcs considerably extends the application area of the package beyond that available to its straight line counterpart. Additional procedures for detecting and repairing malformed polygons enable input data to be verified so that polygons resulting from any of the operations supplied by the package will be well-formed. The resulting package is therefore both functionally complete and efficient in performance.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the major contribution to this work made by Ivan Sutherland, Professor of Computer Science at the California Institute of Technology. I Buchanan was a graduate student at Caltech during 1978. This implementation was undertaken by E Barton as a final year undergraduate project.

REFERENCES

1 **Newman, W M and Sproull R F** *Principles of interactive computer graphics* Second Edition, McGraw-Hill (1979)

2 **Sutherland, I E, Sproull, R F and Schumacker, R A** 'Characterisation of ten hidden surface algorithms' *Comput. Surv.* Vol 6 No 1 (March 1974)

3 **Sutherland, I E** *The polygon package* California Institute of Technology (1978)

4 **Burton, W** 'Representation of many-sided polygons and polygonal lines for rapid processing' *Commun. ACM* Vol 20, No 3 (March 1977)