10 Simple Hidden Line and Hidden Surface Algorithms

Having drawn a cube and other wire objects we soon become irritated by the lack of solidity in the figures. We would like to consider solid objects, in which case the facets at the front of the object will obviously restrict the view of the facets (and boundary lines) at the back. In order to draw pictures of such objects we have to introduce a hidden surface algorithm; or a hidden line algorithm if we wish to draw all, but only, the visible lines on the object. There are many, many such algorithms – some elementary for specially restricted situations, others very sophisticated for viewing general complicated scenes. The time and store limitations of microcomputers bar us from implementing the very complex algorithms. Nevertheless, by limiting the types and number of objects in the scenes it is possible to get most acceptable pictures. In chapter 12 we discuss a relatively complex algorithm, but here we consider two special types of scene – we use the properties implicit in these special configurations to minimise the work needed to discover which surfaces and lines are hidden. Later in this chapter we shall give a simple method for drawing mathematically defined three-dimensional surfaces, but to start we consider an algorithm for drawing a single solid convex body in three-dimensional space.

For our work on hidden line and surface algorithms we choose to define a scene by storing the NOV vertices of objects in the scene (in the OBSERVED position) in arrays X, Y and Z as usual. However we shall now use facet rather than line information. These data may be implicit in the program or the NOF facets are explicitly stored in an array FACET. The integer code for the colour of the facet will usually be implicit, but if necessary we can store it in an array COL. Should we store the facet data then we also need the array SIZE for the number of edges on each polygonal facet: and to save space we insist that no polygonal facet has more than six edges. Should we need more edges, then the facet must be broken down into a set of smaller polygons. In order to make the hidden surface algorithm easier we impose a restriction on the order ofvertices within the array FACET. The vertices must be stored (or are understood to be) in the order in which they occur around the edge of the facet, and when viewed from the outside ofan object they must be in an anticlockwise orientation. Naturally from the inside the vertices taken in this same order would appear clockwise. We shall also assume that all lines are the junction of two facets. Individual lines not related to facets must be added as trivial two-sided facets.

The Orientation of a Three-dimensional Triangle

Once we have planned our object in terms of vertices and facets, how do we check that the facets are actually anticlockwise? Simp]y write a program! The orientation of any convex polygon can be calculated from any three of its vertices taken in order, and so we need consider only an ordered triangle of vertices from the facet. In chapter 7 we saw a method for calculating the orientation of a two-dimensional triangle. Our problem is solved if we can reduce the three-dimensional situation down into two dimensions.

For simplicity we shall assume that all objects are SETUP about and contain the origin. We also insist that the infinite planes that contain the facets on the surface of an object do not pass through the origin. Then we rotate space so that one of the vertices of the triangle in question lies on the negative z-axis (compare with procedure ‘look3', listing 9.1). Since we assume that the origin is inside the object and the eye is outside, all we need do is project the transformed triangle back on to the x/y plane (that is, ignore the z-coordinates) and treat it like a two-dimensional triangle (in fact one of the three vertices will be (0, 0)). Listing 10.1 is our solution of the problem.

Exercise 10.1
Rewrite the wire-figure procedures of the last chapter by assuming that the data are given as vertices and anticlockwise polygonal facets, and not as lines. Check your facet data with the above program. The line information is still there of course, implicit in the facet data – they are the edges of the facet considered as pairs of vertices. Within this information each line occurs twice, once on each of two neighbouring facets. We do not want to waste time drawing lines twice because of the anticlockwise manner of constructing the figures we note that if a line joins vertex I to vertex J on one facet then the equivalent line on the neighbouring facet joins vertex J to I. So for wire figures stored as facets we shall draw lines from vertex I to vertcx J if and only if I < J.

A Hidden Surface Algorithm for a single closed convex body

A finite convex body is one in which any line segment that joins two points inside the body lies totally within the body – a direct extension of the definition in two-dimensioiaal space. It is automatically closed, and thus it is impossible to get inside the body without crossing through its surface. We orthographically project all the vertices of the object on to the view plane, while noting that a projection of a convex polygon with n sides in three-dimensional space is an n-sided convex polygon (or degenerates to a line) in the view plane. By taking the projected vertices of any facet in the same order as the original, we find that either the new two-dimensional polygon is an anticlockwise orientation, in which case we are looking at the outside of the facet, or the new vertices are clockwise

Listing 10.1

```
100 REM Orientation of 3-D triangle
110 DIM X(3),Y(3),Z(3)
120 DIM A(4,4),B(4,4),R(4,4)
129 REM input and print vertex data
130 CLS : PRINT TAB(0,1) "Orientation of 3-D triangle"
140 PRINT TAB(0,3) "The triangle has vertices"
150 ROW=2
160 FOR I%=1 TO 3
170 ROW=ROW+3
180 PRINT TAB(0,17)"                                          "
190 PRINT TAB(0,16) "Type coordinates of vertex ";I%
200 INPUT X(I%),Y(I%),Z(I%)
210 PRINT TAB(0,ROW) "Vertex "; I%
220 PRINT TAB(0,ROW+1);"(";X(I%);",";Y(I%);",";Z(I%);")"
230 NEXT I%
240 PRINT TAB(0,14) "is ";
249 REM matrix R places first vertex on negative z-axis
250 PROCidR3 : THETA=-FNangle(X(1),Y(1))
260 PROCrot3(THETA,3) : PROCmult3
270 THETA=PI-FNangle(Z(1),SQR(X(1)*X(1)+Y(1)*Y(1)))
280 PROCrot3(THETA,2) : PROCmult3
289 REM position 3 vertices with R
290 FOR I%=1 TO 3
300 XX=X(I%) : YY=Y(I%) : ZZ=Z(I%)
310 X(I%)=R(1,1)*XX+R(1,2)*YY+R(1,3)*ZZ+R(1,4)
320 Y(I%)=R(2,1)*XX+R(2,2)*YY+R(2,3)*ZZ+R(2,4)
330 NEXT I%
339 REM (DX1,DY1,0) and (DX2,DY2,0) ordered directional vectors
340 DX1=X(2)-X(1) : DY1=Y(2)-Y(1)
350 DX2=X(3)-X(2) : DY2=Y(3)-Y(2)
359 REM find z-value of dot product negative = clockwise,
      positive = anticlockwise
360 IF DX1*DY2-DX2*DY1 >0 THEN PRINT"ANTI-";
370 PRINT "CLOCKWISE"
380 PRINT TAB(0,16)"if the eye and origin are on opposite sides
of the facet"
390 PRINT TAB(0,21) : STOP
```

and we are looking at the underside. Since the object is closed we are able to see only the outside of facets, the view of their under-side being blocked by the bulk of the object. Therefore we need draw only the anticlockwise polygonal facets – a very simple algorithm, which can be implemented in either construction or ‘drawit’ procedures.

For example, an adjusted constmction procedure ‘cube’ for eliminating the hidden surfaces from an orthographic picture of a cube is given as listing 10.2. Here we do not store the facets, but instead READ the information from DATA and draw the visible facets immediately. The facet data, induding colour, are implied in the program listing. This program was used to produce figure 10.1 , a hidden surface version of figure 9.1d. We take the procedures that were used in the last chapter to draw figure 9.1d, except of course for the construction procedure which sets up the data as vertices and facets, and draws the object (listing 10.2 replaces listing 9.4 in the program for drawing figure 9.1d).

Naturally we use the same data that were used for figure 9.1d. Note that if we colour in the facets with the same logical colour as the background then we get a hidden linee as well as a hidden surface algorithm.

Listing 10.2

```
6500 REM cube / not stored, no hidden surfaces
6510 DEF PROCcube
6520 LOCAL I%,XX,YY,ZZ,F1,F2,F3,F4,DX1,DY1,DX2,DY2
6530 DATA 1,1,1, 1,1,-1, 1,-1,-1, 1,-1,1,
         -1,1,1, -1,1,-1, -1,-1,-1, -1,-1,1
6540 DATA 1,2,3,4, 5,8,7,6, 1,5,6,2, 2,6,7,3, 3,7,8,4, 4,8,5,1
6550 RESTORE
6559 REM READ and position vertices
6560 FOR I%=1 TO 8
6570 READ XX,YY,ZZ
6580 X(I%)=R(1,1)*XX+R(1,2)*YY+R(1,3)*ZZ+R(1,4)
6590 Y(I%)=R(2,1)*XX+R(2,2)*YY+R(2,3)*ZZ+R(2,4)
6600 Z(I%)=R(3,1)*XX+R(3,2)*YY+R(3,3)*ZZ+R(3,4)
6610 NEXT I%
6620 FOR I%=1 TO 6
6629 REM READ facet data
6630 READ F1,F2,F3,F4
6639 REM check orientation
6640 DX1=X(F2)-X(F1) : DY1=Y(F2)-Y(F1)
6650 DX2=X(F3)-X(F2) : DY2=Y(F3)-Y(F2)
6660 IF DX1*DY2-DX2*DY1 < 0 THEN 6750
6669 REM if anticlockwise draw facet
6670 PROCtriangle(X(F2),Y(F2),X(F1),Y(F1),X(F3),Y(F3),1,-2)
6680 PROCtriangle(X(F4),Y(F4),X(F1),Y(F1),X(F3),Y(F3),1,-2)
6690 GCOL 0,0
6699 REM draw edge lines of facet
6700 PROCmoveto(X(F1),Y(F1))
6710 PROClineto(X(F2),Y(F2))
6720 PROClineto(X(F3),Y(F3))
6730 PROClineto(X(F4),Y(F4))
6740 PROClineto(X(F1),Y(F1))
6750 NEXT I%
6760 ENDPROC
```

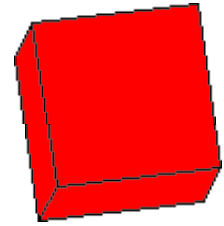

Figure 10.1

If we had stored the colours of the facets in the array COL then naturally we would call the 'triangle' procedure for the I%th facet with colour parameter COL(I%) or we can explicitly fill in the polygon in the program.

We now give a hidden surface construction procedure for an icosahedron (listing 10.3). Change line 6020 to hold 12 vertices.

Listing 10.3

```
6500 REM icosahedron
6510 DEF PROCicosahedron
6520 LOCAL I%,D,XX,YY,ZZ,F1,F2,F3,DX1,DY1,DX2,DY2
6530 D=(1+SQR(5))/2
6540 DATA 0,1,D, D,0,1, 1,D,0, 0,-1,D, D,0,-1, -1,D,0, 0,1,-D, -
D,0,1, 1,-D,0, 0,-1,-D, -D,0,-1, -1,-D,0
6550 DATA 1,3,2, 1,2,4, 1,4,8, 1,8,6, 1,6,3, 2,3,5, 2,9,4,
4,12,8, 8,11,6, 3,6,7, 2,5,9, 4,9,12, 8,12,11, 6,11,7, 3,7,5,
5,10,9, 9,10,12, 12,10,11, 11,10,7, 7,10,5
6560 RESTORE
6570 FOR I%=1 TO 12
6580 READ XX,YY,ZZ
6590 X(I%)=R(1,1)*XX+R(1,2)*YY+R(1,3)*ZZ+R(1,4)
6600 Y(I%)=R(2,1)*XX+R(2,2)*YY+R(2,3)*ZZ+R(2,4)
6610 Z(I%)=R(3,1)*XX+R(3,2)*YY+R(3,3)*ZZ+R(3,4)
6620 NEXT I%
6630 FOR I%=1 TO 20
6640 READ F1,F2,F3
6650 DX1=X(F2)-X(F1) : DY1=Y(F2)-Y(F1)
6660 DX2=X(F3)-X(F2) : DY2=Y(F3)-Y(F2)
6670 IF DX1*DY2-DX2*DY1 < 0 THEN 6740
6680 PROCtriangle(X(F2),Y(F2),X(F1),Y(F1),X(F3),Y(F3),1,-2)
6690 GCOL 0,0
6700 PROCmoveto(X(F1),Y(F1))
6710 PROClineto(X(F2),Y(F2))
6720 PROClineto(X(F3),Y(F3))
6730 PROClineto(X(F1),Y(F1))
6740 NEXT I%
6750 ENDPROC
```

Exercise 10.2

Change listing 10.2 so that it can draw a rectangular block of length LH, breadth BH and height HT, where LH, BH and HT are input parameters to the procedure. Then draw a hidden line picture of it. Draw hidden line pictures of tetrahedra, pyramids, octahedra etc. Add extra parameters to distort these figures so that they are no longer regular, but are still convex.

Exercise 10.3

Rather than have a one-colour cube with black edges drawn in on a white background, give it three colours (red, yellow and black) where opposite faces have the same colour and the edges are not drawn. The information on the colour of a facet should be stored in DATA alongside the vertices, so the program would contain

READ F1, F2, F3, F4, COLOR

When it comes to colouring in a facet you must call the 'triangle' procedure with colour parameter COLOUR.

Also draw an icosahedron in two colours (red and yellow) with edges in bl

Instead of drawing a hidden surface picture of a cube with facets drawn in one colour we can put patterns on the visible sides. Listing 10.4 can be used in conjunction with 'lib1' and 'lib3' to draw a flag, in a similar way to that of the two-dimensional example 4.1, on the side of a cube (see figure 10.2).

Listing 10.4

```
6000 REM scene3
6010 DEF PROCscene3
6020 DIM X(8),Y(8),Z(8)
6030 DIM FACE(4),XD(12),YD(12)
6040 DIM A(4,4),B(4,4),R(4,4)
6050 PROCidR3 : PROClook3
6060 PROCcube
6070 ENDPROC

6500 REM cube / with flags
6510 DEF PROCcube
6520 LOCAL I%,DX1,DY1,DX2,DY2
6530 DATA 1,1,1, 1,1,-1, 1,-1,-1, 1,-1,1,
         -1,1,1, -1,1,-1, -1,-1,-1, -1,-1,1
6540 DATA 1,2,3,4,  5,8,7,6,  1,5,6,2,  2,6,7,3,  3,7,8,4,
4,8,5,1
6550 RESTORE
6559 REM vertices
6560 FOR I%=1 TO 8
6570 READ XX,YY,ZZ
6580 X(I%)=R(1,1)*XX+R(1,2)*YY+R(1,3)*ZZ+R(1,4)
6590 Y(I%)=R(2,1)*XX+R(2,2)*YY+R(2,3)*ZZ+R(2,4)
6600 Z(I%)=R(3,1)*XX+R(3,2)*YY+R(3,3)*ZZ+R(3,4)
6610 NEXT I%
6619 REM facets
6620 FOR I%=1 TO 6
6630 READ F1,F2,F3,F4
6640 DX1=X(F2)-X(F1) : DY1=Y(F2)-Y(F1)
6650 DX2=X(F3)-X(F2) : DY2=Y(F3)-Y(F2)
6660 IF DX1*DY2-DX2*DY1 < 0 THEN 6680
6669 REM draw flags on visible facets
6670 PROCflag
6680 NEXT I%
6690 ENDPROC

7000 REM flag / on face of cube
7010 DEF PROCflag
7020 LOCAL I%,J%,K% : K%=4
7030 FACE(1)=F1 : FACE(2)=F2 : FACE(3)=F3 : FACE(4)=F4
7039 REM place corners of cube face in (XD(i),YD(i)) where
i=1,2,3,4
7040 FOR I%=1 TO 4
7050 XD(I%)=X(FACE(I%)) : YD(I%)=Y(FACE(I%))
7060 NEXT I%
```

```
7070 REM place vertices of diagonal stripes in (XD(i),YD(i))
         where i=5,6,7,8,9,10,11,12
7080 FOR I%=1 TO 4
7090 J%=(I% MOD 4)+1
7100 K%=K%+1
7110 XD(K%)=0.9*X(FACE(I%))+0.1*X(FACE(J%))
7120 YD(K%)=0.9*Y(FACE(I%))+0.1*Y(FACE(J%))
7130 K%=K%+1
7140 XD(K%)=0.1*X(FACE(I%))+0.9*X(FACE(J%))
7150 YD(K%)=0.1*Y(FACE(I%))+0.9*Y(FACE(J%))
7160 NEXT I%
7169 REM draw red background
7170 PROCquad(1,2,3,4)
7179 REM draw yellow diagonals
7180 PROChex(1,5,8,3,9,12)
7190 PROChex(2,7,10,4,11,6)
7199 REM draw edge of cube face
7200 GCOL 0,0
7210 MOVE FNX(X(F1)),FNY(Y(F1))
7220 DRAW FNX(X(F2)),FNY(Y(F2))
7230 DRAW FNX(X(F3)),FNY(Y(F3))
7240 DRAW FNX(X(F4)),FNY(Y(F4))
7250 DRAW FNX(X(F1)),FNY(Y(F1))
7260 ENDPROC

7300 REM quadrilateral
7310 DEF PROCquad(V1,V2,V3,V4)
7320 GCOL 0,1
7330 MOVE FNX(XD(V2)), FNY(YD(V2))
7340 MOVE FNX(XD(V1)), FNY(YD(V1))
7350 PLOT 85,FNX(XD(V3)), FNY(YD(V3))
7360 PLOT 85,FNX(XD(V4)), FNY(YD(V4))
7370 ENDPROC

7400 REM hexagon
7410 DEF PROChex(V1,V2,V3,V4,V5,V6)
7420 GCOL 0,2
7430 MOVE FNX(XD(V1)), FNY(YD(V1))
7440 MOVE FNX(XD(V2)), FNY(YD(V2))
7450 PLOT 85,FNX(XD(V6)), FNY(YD(V6))
7460 PLOT 85,FNX(XD(V3)), FNY(YD(V3))
7470 PLOT 85,FNX(XD(V5)), FNY(YD(V5))
7480 PLOT 85,FNX(XD(V4)), FNY(YD(V4))
7490 ENDPROC
```

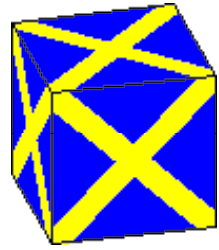

Figure 10.2

Exercise 10.4
Draw a hidden surface picture of a red die with black spots, and with edges drawn also in black. Remember that the values on opposite faces of a die sum to seven.

Bodies of revolution

We can use this anticlockwise versus clockwise method to produce hidden surface pictures of the bodies of revolution that were defined in chapter 9. As we go through the NUMH revolutions we generate NUMV facets with each move. Provided that these quadrilateral (or perhaps degenerate triangular) facets are carefully constructed in an anticlockwise orientation then we may use the same algorithm. Listing 10.5 is such a 'revbod' procedure which produces figure 10.3, a hidden surface version of figure 9.4 (and uses the same input data). Again, because of the modular design of our programs, all the procedures needed to draw figure 10.3, except 'revbod', are the same as those given in chapter 9. Now, however, we must deal solely with convex bodies of revolution.

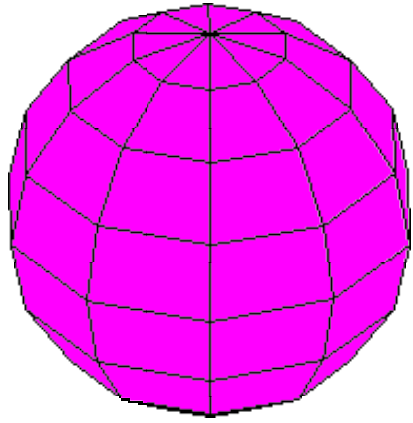

Figure 10.3

As the procedure rotates the definition set of lines about the vertical axis, it stores the vertices of two consecutive vertical sets of lines. These form the vertical edges of one slice of facets. The vertices on these facets are immediately transformed by R (the SETUP to OBSERVED matrix) and stored in arrays X and Y. In such a configuration of pairs of vertical lines the first set of vertices

Listing 10.5

```
6500 REM revbod / hidden surface
6510 DEF PROCrevbod
6520 LOCAL I%,J%,THETA,TD,N1,C,S,XX,YY,ZZ,F1,F2,F3,F4
6530 THETA=PHI : TD=PI*2/NUMH
6539 REM loop through consecutive pairs of fixed-Z lines
6540 N1=NUMV+1 : C=COS(PHI) : S=SIN(PHI)
6549 REM first vertical set
6550 FOR I%=1 TO N1
6559 REM move along polygons formed between these two lines
6560 XX=XD(I%)*C : YY=YD(I%) : ZZ=XD(I%)*S
6570 X(I%)=R(1,1)*XX+R(1,2)*YY+R(1,3)*ZZ+R(1,4)
6580 Y(I%)=R(2,1)*XX+R(2,2)*YY+R(2,3)*ZZ+R(2,4)
6590 NEXT I%
6599 REM loop thru second vertical set
6600 FOR J%=1 TO NUMH
6610 THETA=THETA+TD : C=COS(THETA) : S=SIN(THETA)
6620 FOR I%=1 TO N1
6630 XX=XD(I%)*C : YY=YD(I%) : ZZ=XD(I%)*S
6640 X(I%+N1)=R(1,1)*XX+R(1,2)*YY+R(1,3)*ZZ+R(1,4)
6650 Y(I%+N1)=R(2,1)*XX+R(2,2)*YY+R(2,3)*ZZ+R(2,4)
6660 NEXT I%
6669 REM facet formed by F1,F2,F3 and ( F4 )is SETUP anticlockwise
6670 FOR I%=1 TO NUMV
6680 F1=I% : F2=I%+1
6690 IF I%=NUMV THEN F3=F2+NUMV ELSE F3=F2+N1
6700 DX1=X(F2)-X(F1) : DY1=Y(F2)-Y(F1)
6710 DX2=X(F3)-X(F2) : DY2=Y(F3)-Y(F2)
6719 REM if OBSERVED anticlockwise then then facet visible
6720 IF DX1*DY2-DX2*DY1 < 0 THEN 6810
6729 REM points on fixed-Z = ZV line are put in OBSERVED position
6730 F3=F2+N1 : F4=F3-1
6740 PROCtriangle(X(F1),Y(F1),X(F2),Y(F2),X(F3),Y(F3),1,-1)
6750 PROCtriangle(X(F1),Y(F1),X(F4),Y(F4),X(F3),Y(F3),1,-1)
6760 GCOL0,0 : PROCmoveto(X(F1),Y(F1))
6770 PROClineto(X(F2),Y(F2))
6780 PROClineto(X(F3),Y(F3))
6790 PROClineto(X(F4),Y(F4))
6800 PROClineto(X(F1),Y(F1))
6810 NEXT I%
6820 FOR I%=1 TO N1
6830 X(I%)=X(I%+N1) : Y(I%)=Y(I%+N1)
6840 NEXT I% : NEXT J%
6850 ENDPROC
```

have indices from 1 to NUMV + 1 (= N1), and the second from N1+ 1 to 2 * N1. The I^{th} facet is bounded by four lines, two vertical which join vertex I to I + 1, and I + N1 to I + N1 + 1 , and two horizontal which join I to I + N1, and I + 1 to I + N1 + 1 . Adjustments must be made if one of the original vertices is on the axis of rotation, in which case the quadrilateral degenerates to a triangle. The order of vertices in each facet is carefully chosen so that it is in anticlockwise orientation when viewed from outside the object. This allows us to use our simple algorithm to draw the object with the hidden surfaces suppressed. This technique was also used to draw figure 1.1 in the introduction.

Exercise 10.5
Experiment with this technique. Any initial set of lines will do provided that it starts and ends on the vertical axis and the polygon thus formed in the x/y plane is convex.

The BACK to FRONT Method
The call for pictures of convex solids is limited, so we shall now look at another simple algorithm that can be used with non-convex figures. You will have noticed that when colouring a new area by using option GCOL 0, all the colours previously placed in that section of the screen are obliterated. This furnishes us with a very simple hidden surface algorithm, namely we draw the areas furthest from the eye first and the nearest last. Exactly what we mean by furthest/nearest is not that straightforward. It is not just a matter of comparing *z*-coordinates in a a scene. Imagine a small book and a large tab]e. There are some vertices on the table that have larger *z*-coordinates than those on the book, and some that are smaller. We do not know if the book is on the table or under it. However there are certain situations (for example, as described in the next section, and the stack of cubes of figure 11.5) where this phrase has a very simple meaning and the algorithm is easy to implement. See chapter 12 for a general solution.

Drawing a Special Three-dimensional Surface
We consider the construction of a restricted type of three-dimensional surface in which the *y*-coordinate of each point on the surfice is given by a singlevalued function ‘f’ of the *x*-coordinate and *z*-coordinate of that point. ‘f’ will be included as a procedure in the program – one such example is given in listing 10.6, the function $y = 4 \times SIN(XZ)/XZ$ where $XZ = \sqrt{(x^2 + z^2)}$, which is shown in figure 10.4. The data required were mode = 1 , HORIZ = 32, VERT = 24, NX = NZ = 16, XMIN = ZMIN = -10, XMAX = ZMAX = 10, (EX, EY, EZ) = (3, 2, 1), (DX, DY, DZ) = (0, 0, 0).

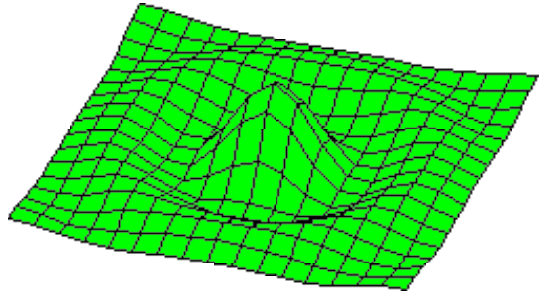

Figure 10.4

Listing 10.6

```
6000 REM scene3 / math. surface
6010 DEF PROCscene3
6020 DIM A(4,4),B(4,4),R(4,4)
6030 CLS
6039 REM INPUT grid data
6040 INPUT"NX,XMIN,XMAX ",NX,XMIN,XMAX
6050 INPUT"NZ,ZMIN,ZMAX ",NZ,ZMIN,ZMAX
6060 XD=(XMAX-XMIN)/NX : ZD=(ZMAX-ZMIN)/NZ : ZV=ZMIN : NX1=NX+1
6070 DIM X(2,NX1),Y(2,NX1),Z(2,NX1)
6079 REM view from (EX,EY,EZ) to (0,0,0) : EX and EZ > 0
6080 PROCidR3 : PROClook3
6089 REM draw the surface
6090 PROCsurface
6100 ENDPROC

6500 REM surface
6510 DEF PROCsurface
6520 LOCAL I%,J%,K%
6530 PROCsetpt(1)
6539 REM loop thru consecutive pairs of fixed-Z lines
6540 FOR I%=1 TO NZ
6550 ZV=ZV+ZD : PROCsetpt(2) : K%=1
6559 REM move along polygons formed between these two lines
6560 FOR J%=2 TO NX1
6570 PROCpoly(I%,J%,K%) : K%=J%
6580 NEXT J%
6590 FOR J%=1 TO NX1
6600 X(1,J%)=X(2,J%) : Y(1,J%)=Y(2,J%) : Z(1,J%)=Z(2,J%)
6610 NEXT J%
6620 NEXT I%
6630 ENDPROC

6700 REM setpt
6710 DEF PROCsetpt(M%)
6720 LOCAL I%,XV,YV
6729 REM points on fixed-Z = ZV line are put in OBSERVED position
6730 XV=XMIN
6740 FOR I%=1 TO NX1
6750 YV=FNf(XV,ZV)
6760 X(M%,I%)=R(1,1)*XV+R(1,2)*YV+R(1,3)*ZV+R(1,4)
6770 Y(M%,I%)=R(2,1)*XV+R(2,2)*YV+R(2,3)*ZV+R(2,4)
6780 Z(M%,I%)=R(3,1)*XV+R(3,2)*YV+R(3,3)*ZV+R(3,4)
6790 XV=XV+XD
6800 NEXT I%
6810 ENDPROC

6900 REM FNf / function to be drawn
6910 DEF FNf(XV,ZV)
6920 LOCAL R
6930 R=SQR(XV*XV+ZV*ZV)
6940 IF R<0.00001 THEN=4.0 ELSE =4*SIN(R)/R
7000 REM zvecprod / z-value of vector product
7010 DEF FNzvecprod(I%,J%,K%,L%,M%,N%)
7020 LOCAL DX1,DY1,DX2,DY2
7029 REM check orientation of triangle ((X(I%,J%),Y((I%,J%)) to
       ((X(K%,L%),Y((K%,L%)) to ((X(M%,M%),Y((M%,M%))
7030 DX1=X(K%,L%)-X(I%,J%) : DY1=Y(K%,L%)-Y(I%,J%)
7040 DX2=X(M%,N%)-X(K%,L%) : DY2=Y(M%,N%)-Y(K%,L%)
7050 =DX1*DY2-DX2*DY1
```

```
7100 REM poly
7110 DEF PROCpoly(I%,J%,K%)
7120 LOCAL L%,ZP1,ZP2,A1,B1,A2,B2,A3,B3,A4,B4,A5,B5,
         C11,C12,C21,C22,D1,D2,DET,RMU
7130 ZP1=FNzvecprod(1,K%,1,J%,2,J%)
7140 ZP2=FNzvecprod(2,J%,2,K%,1,K%)
7149 REM grid rectangle transforms to quadrilateral or 2 triangles
7150 IF SGN(ZP1) <> SGN(ZP2) THEN 7250
7159 REM draw quadrilateral
7160 PROCtriangle(X(1,K%),Y(1,K%),X(1,J%),Y(1,J%),X(2,J%),Y(2,J%),1,-1)
7170 PROCtriangle(X(2,J%),Y(2,J%),X(2,K%),Y(2,K%),X(1,K%),Y(1,K%),1,-1)
7180 GCOL 0,0
7189 REM draw outline
7190 PROCmoveto(X(2,J%),Y(2,J%))
7200 PROClineto(X(1,J%),Y(1,J%))
7210 PROClineto(X(1,K%),Y(1,K%))
7220 PROClineto(X(2,K%),Y(2,K%))
7230 PROClineto(X(2,J%),Y(2,J%))
7240 ENDPROC
7249 REM find intersection (A5,B5) of lines from (A1,B1) to (A2,B2)
        and from (A3,B3) to (A4,B4)
7250 A1=X(1,K%) : B1=Y(1,K%) : A2=X(1,J%) : B2=Y(1,J%)
7260 A3=X(2,J%) : B3=Y(2,J%) : A4=X(2,K%) : B4=Y(2,K%)
7270 FOR L%=1 TO 2
7280 C11=A2-A1 : C12=A3-A4 : C21=B2-B1 : C22=B3-B4
7290 D1=A3-A1 : D2=B3-B1 : DET=C11*C22-C21*C12
7300 IF ABS(DET)<0.0000001 THEN 7360
7310 RMU=(D1*C22-D2*C12)/DET : IF RMU<0 OR RMU>1 THEN 7360
7320 A5=A1+RMU*C11 : B5=B1+RMU*C21
7329 REM draw the two triangles (A1,B1) to (A4,B4) to (A5,B5)
        (A2,B2) to (A3,B3) to (A5,B5)
7330 PROCtriangle(A1,B1,A4,B4,A5,B5,1,-1)
7340 PROCtriangle(A2,B2,A3,B3,A5,B5,1,-1)
7350 GOTO 7180
7359 REM no intersection so swap (A2,B2) with (A4,B4)
7360 AA=A2 : A2=A4 : A4=AA
7370 AA=B2 : B2=B4 : B4=AA
7380 NEXT L%
7390 ENDPROC
```

Since it is impossible to draw every point on the surface we have to approximate by considering a subset of these surface points. We choose those points ith *x*/*z* coordinates on a grid, in other words, when orthographically viewed directly frpm above (thus ignoring the *y*-values), the points form a rectangular grid. This grid is composed of NX by NZ rectangles in the *x*/*z* plane. The *x*-coordinates of the vertices are equi-spaced and vary between XMIN and XMAX (XMIN < XMAX) and the equi-spaced *z*-values vary between ZMIN and zMAX (ZMIN < ZMAX). There are thus (NX + 1) × (NZ + 1) vertices (X, Z) in the grid which can be identified by the pair of integers (i, j):

X = XMIN + *i* × XV where 0 ≤ i ≤ NX and XV = (XMAX – XMIN)/NX

Z = ZMIN + *j* × ZV where 0 ≤ j ≤ NZ and ZV = (ZMAX – ZMIN)/NZ

The equivalent point on the surface is (X, Y, Z) where Y = f(X, Z). Every one of the (NX + 1) × (NZ + 1) points generated in this way is joined to its four

immediate neighbours along the grid (that is, those with equal *x*-values or equal *z*-values), unless it lies on the edge, in which case it is joined to three or, in the case of corners, to two neighbours.

The approximation to the surface that is formed in this way may undulate so not all the facets need be visible from a given view point – in fact some may even be partially visible. We devise a very simple method to eliminate the hidden surfaces by working from the back of the surface to the front. To simplify the algorithm we assume that the eye is always in the positive quadrant, that is, EX > 0 and EZ > 0, and that the eye is always looking at the origin (DX = DY = DZ = 0). If the function is non-symmetrical and we wish to view it from another quadrant then we simply change the sign of *x* and/or *z* in the function. We can then transform the surface into the OBSERVED position.

We start by looping through the set of NX facets that are generated from the consecutive fixed-x grid lines x = XMIN + i × XV and x = XMIN + (i + 1) × XV from the back (i = 0) to the front (i = NX – 1) (naturally the term back to front is used in the sense of the final OBSERVED position). Within each set we loop through the individual facets that are generated by the intersection of the fixed-x lines with the fixed-*z* grid lines starting at z = ZMUN and z = ZMIN + ZV, and working through to z = ZMAX – ZV and z = ZMAX. Wc may label the four grid points created in this way (1, K) and (1, K + 1) – on the fixed *x*-line with smaller *x*-value – and (2, K) and (2, K + 1) – on the fixed-*x* line with larger value (K + 1 is called J in the program , and in the explanation below). 'f' is used to form four points on the surface from these grid points, which when transformed on to the view plane form either a quadrilateral or two triangles. We distinguish beiween the two possibilities by finding the orientation of the two triangles that are formed by the grid points (1, K), (1, J) and (2, J), and (2, J), (2, K) and (1 , K). If they have the same orientation (both clockwise or both anticlockwise) then we have a quadrilateral, otherwise two triangles. The extra vertex of the two triangles is found from either the intersection of the lines joining (1 , K) to (1, J) and (2 , J) to (2, K), or from (1 , K) to (2, K) and (1 , J) to (2, J): other combinations are topologically impossible. After having found the quadrilateral or two triangles they are coloured in and their edges also drawn (procedure 'poly'), and the back to front construction (because EX and EZ are positive) means that we get a correct hidden surface picture.

Exercise 10.6

Change the functions 'f' used by this program. For example use f = 4SIN(t) where $t = \sqrt{(x^2 + z^2)}$.

Exercise 10.7

Extend the above program so that it draws the top-side ofthe surface in a different colour to the under-side.

Complete Programs

I ‘lib3’ and listing 10.1 . Data required: the vertex coordinates ofa triangle (X(1), Y(1), Z(1)), where $1 \leq I \leq 3$. Try (1, 0, 1), (1, 1, 0) and (0, 1, 1); also the same vertices in a different order (1, 1, 0), (1, 0, 1) and (0, 1, 1).

II ‘lib1’, ‘lib3’ and listings 9.3 (‘scene3’) and 10.2 (‘cube’). Data required: mode, HORIZ, VERT, (EX, EY, EZ), (DX, DY, DZ). Try 1, 8, 6, (1, 2, 3), (0, 0, -1).

III ' lib1’, ‘lib3’ and listing 10.3 (‘icosa' hedron) and listing 9.3 (‘scene3’ but change ‘cube’ to ‘icosa’). Data required: mode, HORIZ, VERT, (EX, EY, EZ) and (DX, DY, DZ). Try the same data as used in II. Change line 6020 to hold 12 vertices.

IV ' lib1’, ‘lib3’ and listing 10.4 (‘cube’, ‘flag’ etc.). Data required: mode, HORIZ, VERT, (EX, EY, EZ) and (DX, DY, DZ). Try the same data as used in II.

V ' lib1 ‘, "lib3’ and listings 9.10 (‘scene3’) and 10.5 (‘revbod’). Data required: mode, HORIZ, VERT, NUMH, NUMV, PHI, (EX, EY, EZ), (DX, DY, DZ). Try 1, 3.2, 2.4, 10, 10, 1 , (1, 2, 3), (0, 0, 0).

VI ‘lib1’, ‘lib3’ and listing 10.6 (‘scene3’, ‘surface’, ‘setpt’, ‘f’, ‘zvecprod’ and ‘poly’). Data required: mode, HORIZ, VERT, NX, XMIN, XMAX, NZ, ZMIN, ZMAX, (EX, EY, EZ), (DX, DY, DZ). Try 1, 28, 21, 16, -8, 8, 16, -8, 8, (1, 2, 3) and (0, 0, 0). Use the VDU 19 command to change the colour of the surface; for example VDU 19, 1, 3, 0, 0, 0. If you find that there is not enough store even after changing PAGE, then you can run the program in mode 4, or strip off all the REMs from the program and rerun in mode 1. Just type AUTO 9, 10 and hold the RETURN key down and this will do the trick.