

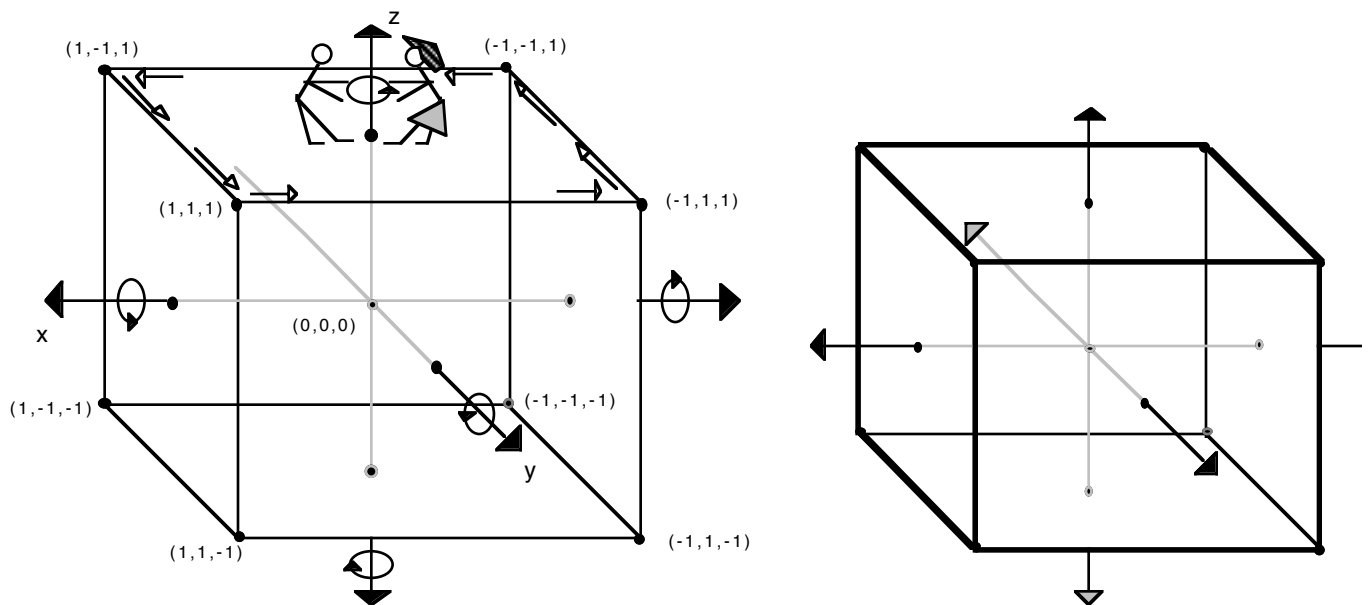
## ASSIGNMENT III

(continued)

### THE HIDDEN LINE ALGORITHM FOR CONVEX POLYHEDRA

For a more realistic representation of an opaque solid object, say a polyhedron, it is necessary to omit drawing the edges and faces that are hidden from view by the object itself. If the solid is convex, then there is a very simple algorithm for determining which faces are visible and which are not. The idea is to start from an input file describing the solid *one face at the time*. Of course then each edge ends up being repeated twice (once for each of the two faces containing it), but that is a minor price to pay for the advantages gained.

To understand why we need such a structure in the data, can best be explained by studying the pictures below



Imagine that you are an observer looking at a given rotated picture from a point below the  $x, y$ -plane. In fact, assume the  $x, y$ -plane is the ceiling and that you are looking up. Say what you see is the smaller cube depicted above on the right. We have drawn there all the faces that you see in a bold line and the hidden faces by thin lines. Note now that the hidden faces have an outward pointing normal which points away from you (that is upwards, those with gray arrow tips) while the faces that you see have an outward pointing normal which points towards you (that is downwards, those with black arrow tips). This simple observation is the basis of a *hidden face algorithm* for the realistic representation of opaque convex polyhedra. The algorithm is easily formulated:

(1) Rotate your object, (2) calculate the outward pointing normal of each of the faces, (3) draw only the faces whose outward pointing normal has negative  $z$ -component.

As we indicated in the “ROTATION” part of this assignment, the input solid should be given as a list of its planar faces. In turn each face should be presented by giving the coordinates  $(x, y, z)$  of the

successive vertices of its polygonal boundary as they are encountered in the *outer counterclockwise order* (as you can see for the two figurines standing on the big cube on the left).

If your data is entered in this manner, and the first three corners of a face of the rotated object are

$$\{ \{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}, \{x_3, y_3, z_3\}, \dots \} \quad (1)$$

then the direction of the outward pointing normal is given by the cross product of the two vectors

$$(x_2 - x_1, y_2 - y_1, z_2 - z_1) \text{ and } (x_3 - x_2, y_3 - y_2, z_3 - z_2) . \quad (2)$$

From the reasoning above we thus deduce that we must draw this particular face if and only if

$$\det \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{bmatrix} = (x_2 - x_1)(y_3 - y_2) - (x_3 - x_2)(y_2 - y_1) \quad (3)$$

turns out to be negative.

This part of the assignment consists in writing a procedure with heading “**VISIBLES[obj]**” which returns the projections in the  $x, y$ -plane of the visible faces of the input “**obj**”. More precisely, a call of **VISIBLES[obj]** carries out, in succession the following operations:

- (1) picks a face  $\mathbf{F} = \{\{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}, \{x_3, y_3, z_3\}, \dots\}$  of the input **obj**,
- (2) computes the  $z$ -coordinate, say “**Nz**”, of the outer normal of  $\mathbf{F}$  as indicated in (3),
- (3) and if  $\mathbf{Nz} < \mathbf{0}$  then it concatenates to the output sequence the  $x, y$ -projection of  $\mathbf{F}$  in the form

$$\mathbf{Line}[\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \dots\}]$$

This should create an output of the form

$$\begin{aligned} \mathbf{ProjObj} = & \{ \mathbf{Line}[\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \dots\}], \\ & \mathbf{Line}[\{\{x'_1, y'_1\}, \{x'_2, y'_2\}, \{x'_3, y'_3\}, \dots\}], \\ & \mathbf{Line}[\{\{x'_1, y'_1\}, \{x'_2, y'_2\}, \{x'_3, y'_3\}, \dots\}], \dots \} \end{aligned}$$

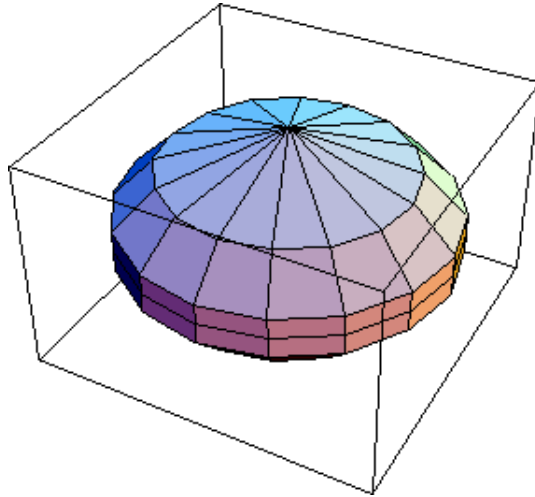
Which a upon a call of

$$\mathbf{Show}[\mathbf{Graphics}[\mathbf{ProjObj}], \mathbf{AspectRatio} \rightarrow \mathbf{Automatic}]$$

Should display a good rendering of the  $x, y$ -projection of your object.

You may construct yourselves a convex polyhedron of your choice to experiment with and demonstrate that your program works. Alternately, you may use the “*flying saucers*” that can be constructed from the procedures that come with this handout. These procedures will also be made available on the Web from a file called “SAUCERS”. If you construct an 18-sided saucer by the command **mkspship[8]**, and display it

you will get a picture such as given below



Note however that the procedure we give you contains the word “*Polygon*” in front of each face of the output, you need these words if you want to use 3D graphics to display the saucer. However, for this assignment these words must be removed before rotating the object. You can then apply the rotation procedure to each vertex of each face. This done you must add the word “**Line**” again in front of every “projection” of a rotated face as indicated above to display the projection of the rotated saucer.

Your assignment consists in posting your procedures on your website together with an animated display of your rotating object. If you want to display something fancier you may try a scene such as the flying saucer rotating as it disappears in the horizon. To do this you must add some other object in the foreground that doesn't move, and then use a translating procedure to make the saucer fly away from the observer.