

ASSIGNMENT V

TESTING INSIDENESS

In further assignments we shall put together an algorithm that will enable us to construct the projections of solids with hidden faces in the most general (non convex) case. It is to be understood that the figures we may wish to represent could consist of two or more solids and, even if each of them is convex, the combined figure is not convex. Indeed, we see that in rotating such a figure, one solid may hide, from the observer's view, portions of the other solids' faces and edges, a situation that does not occur in the case of a single convex polyhedron.

The picture on the right illustrates an example of a non-convex object with rectangular faces. As you can see the visible portions of some faces are not quadrilateral. So any hidden line program that displays such an object will have to cut up the faces into visible and invisible portions. Your procedure **inside** described below will play an essential role in deciding visibility.

The construction of projections of such composite objects presents much greater difficulties and requires considerable more processing. To make our task less of a burden we need to break it up into smaller subtasks. Thus in this assignment we shall deal with only one of the steps needed in the final algorithm: namely, that of determining whether a simple closed polygonal curve C of the x, y - plane does or does not enclose a given point P .

The input data in this case can be taken to be the pair $\{a, b\}$ giving the x, y coordinates of the point P and an $n \times 2$ matrix

$$C = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \dots & \dots \\ a_n & b_n \\ a_1 & b_1 \end{bmatrix}$$

whose rows contain the x, y coordinates of the successive vertices of C , in the order they are encountered as we travel on the curve starting from a point and ending at the same point.

Let for a moment $P_1, P_2, \dots, P_n, P_{n+1} = P_1$ denote the successive vertices of C . Moreover let

$$V_i = P_i - P$$

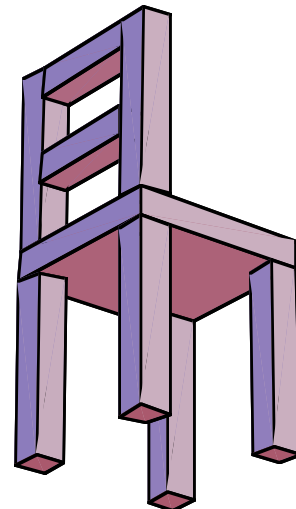
denote the vector joining P to P_i and α_i denote the angle formed by V_i and V_{i+1} measured in the counter-clockwise direction. According to this convention α_i will be positive if a point Q traveling from P_i to P_{i+1} is seen moving left by an observer placed at P and negative if Q is seen moving right.

Now, it is a well known mathematical fact that

$$\alpha_1 + \alpha_2 + \dots + \alpha_n = 0$$

if and only if the point P lies outside the curve C . If the point is inside this sum will be $\pm 2\pi$ depending on how the curve is traveled.

This result is quite intuitive and easy to verify in the case that the curve is a triangle. The general case (see figure below) may be deduced from the case of a triangle by decomposing the given closed polygonal curve into an appropriate sum of triangles. You may also prove it directly by following the motion of the vector $Q - P$ as Q travels along the curve.



Try this approach on the picture on the right hand side.

Write a MATHEMATICA function with heading

`inside[P_, C_]`

where P is a 2-vector $\{a, b\}$ and C the matrix giving the x,y-coordinates of the vertices of the curve. The output B should be a boolean which is equal to **True** or **False** according as the point P is or is not inside the curve given by the matrix C .

It is suggested that the calculation of the angle between two vectors

$$\begin{aligned} V_1 &= (x_1, y_1) \\ V_2 &= (x_2, y_2) \end{aligned}$$

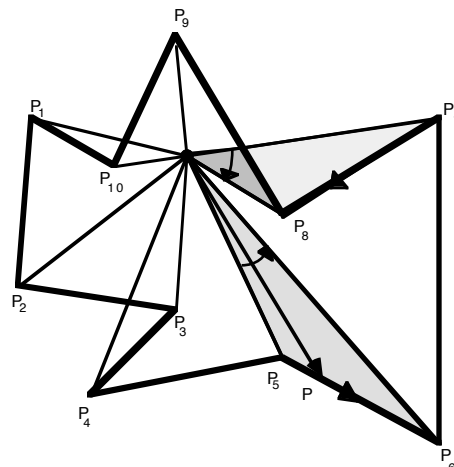
be obtained by the following steps

- (1) Calculate first the two quantities

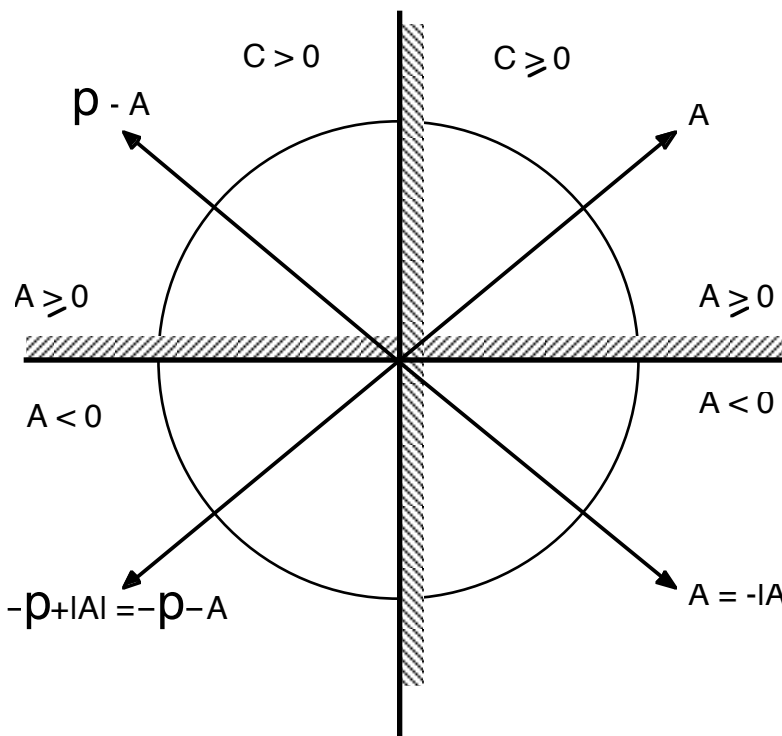
$$A = \text{ArcSin} \left[\frac{x_1 y_2 - x_2 y_1}{\sqrt{x_1^2 + y_1^2} \sqrt{x_2^2 + y_2^2}} \right] \text{ and } C = x_1 x_2 + y_1 y_2. \tag{1}$$

- (2) This done set

$$\alpha = \begin{cases} A & \text{if } A \geq 0 \text{ and } C \geq 0 \\ \pi - A & \text{if } A \geq 0 \text{ and } C < 0 \\ A & \text{if } A < 0 \text{ and } C \geq 0 \\ -\pi - A & \text{if } A < 0 \text{ and } C < 0 \end{cases} \tag{2}$$



The diagram on the right should make it clear why this formula yields the right angle



Note that in the applications we have in mind we may have to call the function **inside** many times and for polygons having many vertices. This given, the computation of all the angles involved, may turn out to be very expensive timewise. So it is suggested that the function **inside** starts with a test which weeds out without much cost, the cases in which the decision (inside or not inside) is easier to reach. This is the case in which the point P being tested is *outside* the smallest rectangle containing the curve (let us refer to the latter as the *bounding rectangle*). The MATHEMATICA functions **Max** and **Min** are precisely what we need to carry out this decision. In fact note that, if the curve C is stored as a list of pairs $\{a_i, b_i\}$ then the commands

$$\mathbf{min} = \mathbf{Map}[\mathbf{Min}, \mathbf{Transpose}[C]] \quad \text{and} \quad \mathbf{max} = \mathbf{Map}[\mathbf{Max}, \mathbf{Transpose}[C]] \quad (3)$$

store in **min** and **max** respectively the pairs

$$\{a_{min}, b_{min}\} \quad \text{and} \quad \{a_{max}, b_{max}\}$$

where a_{min} gives the minimum of all the a_i 's, b_{min} gives the minimum of all the b_i 's and similarly a_{max} gives the maximum of all the a_i 's and b_{max} gives the maximum of all the b_i 's. Thus the opposite vertices of the bounding rectangle are precisely **min** and **max**. Then you can be certain that a point $P = \{a, b\}$ is outside C if at least one of the two tests

$$a_{min} \leq a \leq a_{max} \quad , \quad b_{min} \leq b \leq b_{max}$$

fails. Note that you can carry out this test again by means of **Min** and **Max**. Infact, after the commands in (3) have been carried out, all you have to do is call

$$(\mathbf{Min}[P - \mathbf{min}] \leq 0) \quad || \quad (\mathbf{Max}[P - \mathbf{max}] \geq 0) \quad (4)$$

This should return **True**, if

the smaller of $a - a_{min}$ and $b - b_{min}$ is negative

(i.e. a is smaller than all the a_i 's or b is smaller than all the b_i 's)

or (that is the double bar in (4))

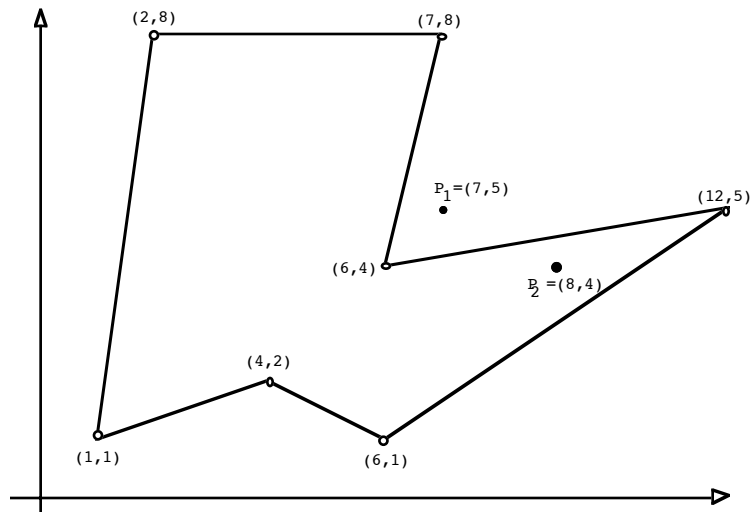
the bigger of $a - a_{max}$ and $b - b_{max}$ is positive

(i.e. a is bigger than all the a_i 's or b is bigger than all the b_i 's).

Either way the command returns **True** if P is not inside the bounding rectangle and **False** if P is inside. By starting your function **inside** with such a test, then you will need to carry out the *angle test* only when P is inside the bounding rectangle.

When you have completed the assignment test your program with the input curve C and the two

points P_1 and P_2 depicted below. This curve makes a good bench mark so I suggest that you use it.



To get some extra credit, you may wish to add a colorful display to your Website. You may do this by using also a curve of your choice. For instance the outline of a Xmass tree. Then write a procedure which creates a user chosen number of random points in a fixed square containing the tree, and each time tests the resulting point and if it is inside the tree displays it in green and if it is not it displays it in yellow. You may also want to display the tree as well. If your procedure **inside** is correct, you will see the outline of the tree appearing as more and more points are displayed.

