[6.45] Determine if a point is inside a triangle

This tip is more application-specific than most, but the idea is so clever that I had to include it. Suppose that we have a triangle defined by the three points A, B and C, with rectangular coordinates (x_a,y_a), (x_b,y_b) and (x_c,y_c). We have another point P at (x,y), and we want to know if P is inside the triangle ABC. In general, if P is in ABC, then the area of ABC will equal the sum of the areas of the three triangles PBC, APC and ABP. If P is outside ABC, then the area of ABC is less than the sum of the areas of the three triangles formed by A, B, C and P. To find the area of a triangle with points P_1, P_2 and P_3, we can use

$$\text{area}(P_1P_2P_3) = \frac{\left\| \begin{matrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{matrix} \right\|}{2} = \text{abs}\left(\frac{x_1y_2+x_3y_1+x_2y_3-x_1y_3-x_2y_1-x_3y_2}{2}\right)$$

where the double bars indicate the absolute value of the determinant of the matrix. By using the absolute value, we ensure that the area is positive, regardless of the ordering of the three points. This function implements the idea:

```
PntTrian(px,py,ax,ay,bx,by,cx,cy,tol)
Func
©(px,py,ax,ay,bx,by,cx,cy,tol) return True if point P is in triangle ABC
©25may01/dburkett@infinet.com

local tArea,at

©Define local function to find triangle area
Define tArea(xx1,yy1,xx2,yy2,xx3,yy3)=Func
abs(xx1*yy2-xx1*yy3-xx2*yy1+xx2*yy3+xx3*yy1-xx3*yy2)/2
EndFunc

©Find area difference & return result

tArea(ax,ay,bx,by,cx,cy)→at

return
abs((at-(tArea(px,py,bx,by,cx,cy)+tArea(ax,ay,px,py,cx,cy)+tArea(ax,ay,bx,by,px,
py)))/at)≤tol

EndFunc
```

The function returns *true* if point (*px*,*py*) is within the triangle defined by the points (*ax*,*ay*), (*bx*,*by*) and (*cx*,*cy*), otherwise it returns *false*. *tol* is a small positive number which specifies the tolerance at which the two areas will be considered equal. If you are using Exact mode, you may set *tol* = 0. If you are using the function in Approx mode, with floating point coordinate arguments, set *tol* to specify the precision at which you want to consider the two areas equal. This will depend on your application. If, for example, you want to assume that the point P is in triangle ABC if the two areas are the same to six significant digits, then set *tol* = 1E-6. A safe minimum value for *tol* is 1E-12, since the 89/92+ results are accurate to about 12 significant digits.

Note that I use a relative tolerance, not an absolute tolerance. This is necessary because the point coordinates may have a wide range. For example, suppose that the three triangle points are (0,0), (0,2E20) and (1E20,1E20), and we want to determine if point P = (-1E10,-1E10) is in the triangle. The area of ABC is 1E40, and the sum of the areas of the three triangles including P is 1.0000000002E40. The absolute difference between the two areas is 2E30, but the relative difference is 2E-10. If we call *PntTrian*() with *tol* = 1E-12, it will correctly return *false* in this case.

Disregarding the tolerance, points at the triangle vertices and on the lines joining the vertices are considered inside the triangle.

As an example, suppose we have a triangle with vertices at (0,0), (7,0) and (4,4). Which of the points (4,1), (5,3) and (2,2) are within the triangle? We will use exact mode and *tol* = 0, so the function calls are

```
PntTrian(4,1,0,0,7,0,4,4,0)
PntTrian(5,3,0,0,7,0,4,4,0)
PntTrian(2,2,0,0,7,0,4,4,0)
```

returns *true*
returns *false*
returns *true*

So the first and third points are in the triangle, but the second point is not. Note that the third point is on a triangle edge.

The idea for this tip was taken from a Design Idea in the August 3, 2000 issue of *Electronic Design News* magazine, titled *Algorithm tests for point location*, by Lawrence Arendt, of the Mannitoba HVDC Research Centre, Winnipeg, Canada.