Bogusław Jackowski

Winding number for Bézier splines

The fairly compact (although not especially efficient) MetaPost code given below can be used to compute a winding angle of a path at a given point:

```
vardef mock_arclength(expr p) = % |p| -- Bézier segment
  % |mock_arclength(p)>=arclength(p)|
  length((postcontrol 0 of p)-(point 0 of p)) +
  length((precontrol 1 of p)-(postcontrol 0 of p)) +
  length((point 1 of p)-(precontrol 1 of p))
enddef;
vardef windingangle(expr p,q) = % |p| -- point, |q| -- Bézier segment
  save a,b,v;
  a=length(p-point 0 of q); b=length(p-point 1 of q);
  if min(a,b)<2eps: % MP is not the master of precision, we'd better stop now
    errhelp "It is rather not advisable to continue. Will return 0.";
    errmessage "windingangle: point unsafely near Bézier segment (dist="
      & decimal(min(a,b)) & ")";
    0
  else:
    v:=mock_arclength(q); % |v| denotes both length and angle
    if (v>=a) and (v>=b): % possibly too long Bézier arc
      windingangle(p, subpath (0,1/2) of q)+windingangle(p, subpath (1/2,1) of q)
    else:
      v:=angle((point 1 of q)-p)-angle((point 0 of q)-p);
      if v>180: v:=v-360; fi
      if v<-180: v:=v+360; fi
      v
    fi
  fi
enddef;
```

The operation *windingangle* can be used to compute a winding number, given a point and a curve, and to determine the mutual position of two nonintersecting (also without selfintersections) cyclic curves, i.e., to test whether one is embeded inside the other or not.

```
vardef windingnumber (expr p,q) = % |p| -- point, |q| -- Bézier spline
  save a; a:=0;
  for t:=1 upto length(q):
    a:=a+windingangle(p, subpath(t-1,t) of q);
  endfor
  a/360
enddef;
tertiarydef a inside b =
  if path a: % |and path b|; |a| and |b| must not touch each other
    begingroup
      save a_,b_; (a_,b_)=
        (windingnumber(point 0 of a,b), windingnumber(point 0 of b,a));
      (abs(a_-1)<eps) and (abs(b_)<eps)
    endgroup
  else: % |numeric a and pair b|
    begingroup
      (a>=xpart b) and (a<=ypart b)
    endgroup
  fi
enddef;
```