How to draw dotted or dashed lines using a pen with a width greater than 1

I once got around this silly limitation by writing a wrapper procedure that would figure out where each dot or dash belonged, and draw many tiny line segments using a solid pen. One of the parameters to the procedure was a string indicating the pattern, which looked like morse code (any arrangement of dots and dashes is allowed).

However, the procedure is ugly. It will handle polylines, and if a polyline vertex falls right in the middle of a dash, then the dash will bend around the corner correctly.

The *LineTo* method of *TCanvas* cannot reliably render dashed lines more than one pixel wide. This procedure provides a workaround.

```
{Copyright (c) 1996 G. Williams}

procedure PlotDashedLine(const Canvas: TCanvas;
  const Vertices: array of TPoint; const Pattern: String;
  const DashLength: Integer);
var
  PenDown: Boolean;
  Index: Integer;

  procedure PlotTo(const Position: TPoint);
  begin
    with Canvas, Position do
      if (PenDown) then
        LineTo(X, Y)
      else
        MoveTo(X, Y);
  end;

  function Advance(const Distance: Integer): Boolean;
  var
    DistanceRemaining: Single;
    DistanceToNextVertex: Single;
  begin
    Result := false;
    DistanceRemaining := Distance;
    DistanceToNextVertex := PointDist(Canvas.PenPos, Vertices[Index]);
    while (DistanceRemaining > DistanceToNextVertex) do
    begin
      DistanceRemaining := DistanceRemaining - DistanceToNextVertex;
      PlotTo(Vertices[Index]);
      Inc(Index);
      if (Index > High(Vertices)) then
        Exit;
      DistanceToNextVertex := PointDist(Canvas.PenPos, Vertices[Index]);
    end;
    with Canvas.PenPos do
      if (FltEqual(DistanceToNextVertex, 0)) then
        PlotTo(Vertices[Index])
      else
        PlotTo(
          Point(
            Round(
              X + DistanceRemaining / DistanceToNextVertex
                * (Vertices[Index].X - X)
            ),
            Round(
              Y + DistanceRemaining / DistanceToNextVertex
                * (Vertices[Index].Y - Y)
            )
          )
        );
      Result := true;
  end;
```

```
var
  PatternIndex: Integer;
  OldPenStyle: TPenStyle;
begin
  OldPenStyle := Canvas.Pen.Style;
  Canvas.Pen.Style := psSolid;
  Canvas.MoveTo(Vertices[0].X, Vertices[0].Y);
  PatternIndex := 1;
  Index := 1;
  while (true) do
  begin
    PenDown := true;
    case Pattern[PatternIndex] of
      '.':
        if not(Advance(0)) then
          Break;
      '-':
        if not(Advance(DashLength)) then
          Break;
      else
        ShowError('');
    end;
    PenDown := false;
    if not(Advance(DashLength)) then
      Break;
    Inc(PatternIndex);
    if (PatternIndex > Length(Pattern)) then
      PatternIndex := 1;
    end;
  Canvas.Pen.Style := OldPenStyle;
end;
```

Original resource:	The Delphi Pool
Author:	Gary Williams
Added:	2009-11-06
Last updated:	2009-11-06