How to calculate intersection points of lines or line sections with rectangles

```
function fuzz(x, fuzzFactor: double): double;
var
  s: string;
begin
  s := format('%.6f', [x]);
  result := StrToFloat(s);
end;
```

Warning: the following function assumes a fuzz factor in comparing values of doubles. This is because of the tendency of zero sloped edges to need some help in avoiding div-by-zero errors.

```
function Intersection (p1, p2, p3, p4: pt; var err: boolean): pt;
var
  m1, m2, b1, b2: double;
  pResult: pt;
begin
  err := false;
  if p2.x = p1.x then
    m1 := MaxReal
  else
    m1 := (p2.y - p1.y) / (p2.x - p1.x);
  if p4.x = p3.x then
    m2 := MaxReal
  else
    m2 := (p4.y - p3.y) / (p4.x - p3.x);
  if m1 = m2 then
  begin  {parallel lines never intersect}
    err := true;
    exit;
  end;
  b1 := (p1.y) - (m1 * p1.x);
  b2 := (p3.y) - (m2 * p3.x);
  if m2 = 0 then
    pResult.y := p3.y
  else
    if m1 = 0 then
      pResult.y := p1.y
    else
      pResult.y := ((m1*b2) - (m2 * b1)) / (m1-m2);
    if (fuzz(m1, 0.0001)) = fuzz(MaxReal, 0.00001) then
      pResult.x := p1.x
    else
      if m1 = 0 then
        if fuzz(m2, 0.00001) = fuzz(MaxReal, 0.00001) then
          pResult.x := p3.x
      else
        pResult.x := (pResult.y - b1)  {/ 0.00001}
      else
        pResult.x := (pResult.y - b1) / m1;
      Result := pResult;
end;
```

Original resource:	The Delphi Pool
Author:	Cliff W. Estes
Added:	2013-04-09
Last updated:	2013-04-09