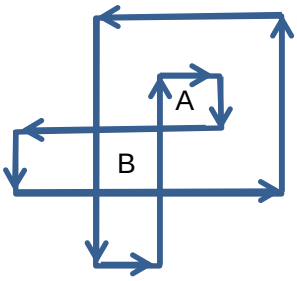
2013/09/20

スポット検知における多角形内外判定のアルゴリズムと実装

動体検知ソフト Msako には、時間帯に応じて映像の特定の領域を動体検知の対象にする、しないを指定できる、スポット検知と呼ぶ機能がある。スポット検知で指定できる領域の形状は、矩形、楕円、多角形のいずれかである。この中でも特に多角形は、対象とする領域が複雑な形状の場合に好適である。Msako でこの多角形の領域を指定する上で領域の内外判定は、次の点で重要な意味を持つ。

- 動体検知の対象になるか否かの判定に関わる
- 領域を選択し変更する場合、その対象となる領域となるか否かの判定に関わる



動体検知の対象領域の指定としては、このようなケースはまれであろうが、10 個の頂点を持つ右の多角形を例にとって考えてみよう。この例で領域 A、B 内の点は、はたして多角形の内外、いずれと判定すべきだろうか。A は明らかに穴（Hole）であり多角形の外側である。一方 B は、重なった領域（自己交差）と考えられ、多角形の内側と判断するべきである。ところが、理由は不明であるが、Microsoft Word 2010 や Excel 2010 の描画ツールでこの多角形を描くと、なぜか B の自己交差領域を多角形の外側としてしまう。

ここで、ある点が多角形内か外かを計算機を使って計算する方法として、代表的な次の 2 つのアルゴリズム[1]を比較してみる。

- Crossing number algorithm
- Winding number algorithm

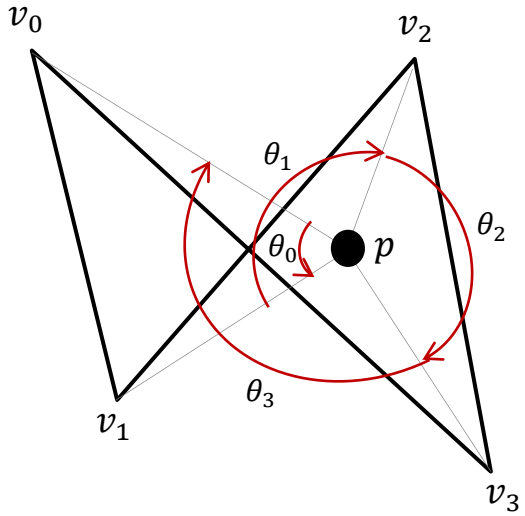
Crossing number algorithm は、判定すべき点から特定の座標軸方向に引いた直線が多角形の辺を何本横切ったか（交差数）で判定する。すなわち、偶数ならばその点は外側、奇数ならば内側である。別名 even-odd rule algorithm と呼ばれる所以である。

一方 Winding number algorithm は、二次元空間上にある閉連続曲線について一般化されたアルゴリズムで、ある点から閉連続曲線上の各点を元に戻るまで見まわすと何回転するかを数え、どちら方向かに 1 回転以上回転すればその点は内側にあると判定するもので

ある[2]。閉連続曲線上の有限数の点を離散的に直線でつなぐと多角形になるので、多角形にもこのアルゴリズムが適用できる。多角形についてこのアルゴリズムをもう少し厳密に書いてみよう。n個の頂点v_i $(i = 0..n-1, v_n = v_0)$ を持つ多角形に対して点pから隣接する 2 つの頂点v_iとv_{i+1}に引いた 2 本の直線$\overline{pv_i}$と$\overline{pv_{i+1}}$の成す角度$\theta_i = \angle(\overline{pv_i}, \overline{pv_{i+1}})$（符号付、単位はラジアン）をすべての頂点について積算し、その結果を2πで割った値 wn が winding number、つまり巻き数である。このwnは次式で表され、wnが 0 ならば点pは多角形の外側、0 以外ならば多角形の内側にあると判定する。

$$wn = \frac{1}{2\pi}\sum_{i=0}^{n-1}\theta_i$$

$$= \frac{1}{2\pi}\sum_{i=0}^{n-1}\cos^{-1}\left(\frac{(v_i - p)\cdot(v_{i+1} - p)}{|v_i - p||v_{i+1} - p|}\right)$$

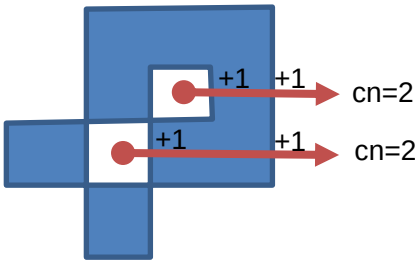


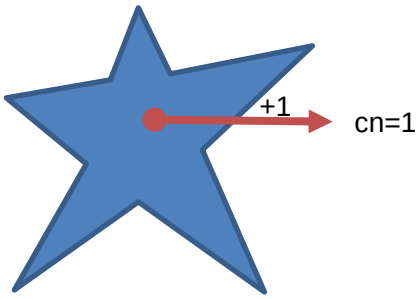
勿論、この表現通りに計算すれば正しい答えを得ることができる。しかし、計算コストの点で高価な逆三角関数を使うことになり好ましくない場合もある。そこで考えられたのが次の方法である[3]。まず Crossing number algorithm と同じように、判定すべき点から特定の座標軸方向に直線を引く。この直線が多角形の辺を横切るとき、この直線と直交する座標軸で見て、その辺がこの座標軸の正負どちらの向きに引かれたかによって＋１、－１の値を与え、この値を積算していく。その結果が 0 であれば外側、0 以外であれば内側と判定するものである。

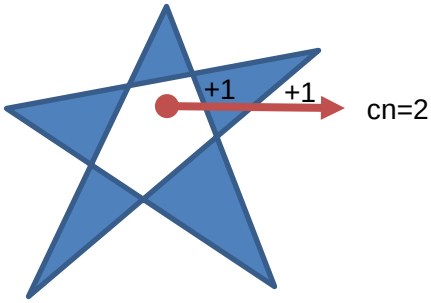
この 2 つのアルゴリズムの違いは、自己交差した領域内の点の判定結果にある。すなわち、自己交差領域について、Cross number algorithm は多角形外と誤認してしまうが、Winding number algorithm は正しく多角形内と判定する。しかも、上記の方法なら Crossing number algorithm よりわずかに多い計算コストを支払うだけで正しい答えが得られる。

Crossing number(cn) algorithm

- Inside : cn is odd
- Outside : cn is even

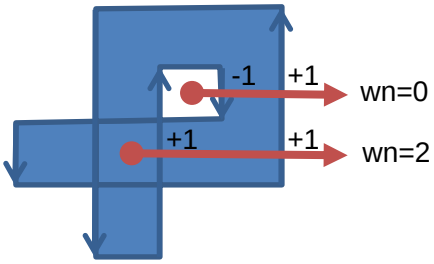


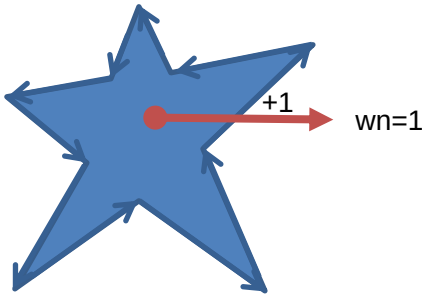


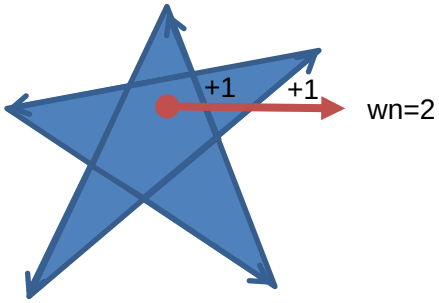


Winding number(wn) algorithm

- Inside : wn $\neq$ 0
- Outside : wn = 0







Msako では、この優れた Winding number algorithm の方を使って多角形の内外判定を行うことにした。このため、例えば五画で一筆書きした星形の中心部分にある自己交差領域内にマウスポインタを移動すると、正しく多角形内と判断しカーソルが次のような上下左右矢印に変化し移動可能であることを示す。そして、マウスをドラッグするとマウスの移動に合わせてこの星形が移動する。

参考資料

[1] Point of Polygon
http://en.wikipedia.org/wiki/Point_in_polygon

[2] Winding number
http://en.wikipedia.org/wiki/Winding_number

[3] Inclusion in a point in Polygon
http://geomalgorithms.com/a03-_inclusion.html