FAQ der Informatik.Ger

Punkt im Polygon

Wie kann ich feststellen, ob ein Punkt in einem BELIEBIGEN Polygon liegt?

Inhalt

Einleitung

Definieren wir als erstes:
Sei P[i] = (x[i], y[i]), mit i=1..n eine Folge von Punkten in der Ebene, die ein Polygon beschreibt und A=(xa,ya) der zu prüfende Punkt in der Ebene. Wegen der Geschlossenheit des Polygonzugs ist als P[n+1] wieder P[1] einzusetzen.

Dabei sollen sich die Kanten nicht überkreuzen oder gar Ecken oder sogar Kanten mehrfach auftreten. Wenn nur konvexe Polygone interessieren, siehe Abschnitt Konvexe Polygone.

Der allgemeine Ansatz (Strahl-Methode)

Man teste zuerst, ob A selbst auf einer Kante des Polygons liegt. Wenn ja, kann man willkürlich entscheiden. Andernfalls nehme man einen (erstmal beliebigen) Strahl, der vom Testpunkt A ausgeht. Man ermittelt, wie oft dieser Strahl eine Kante des Polygonzugs schneidet. Ist diese Anzahl ungerade, liegt der Punkt innerhalb, andernfalls außerhalb des Polygons.

Dieser Ansatz funktioniert wunderbar, solange der Strahl nicht auf eine Ecke des Polygons trifft. Um diesen Fall zu erschlagen, gibt es zweierlei Strategien:

die Straight-On-Variante

Man wählt solange zufällig Strahlen aus, bis einer gefunden ist, der keine Ecke des Polygons schneidet.

Ein einfacher Test, ob ein Strahl mit der Richtung (c,s) eine Ecke des Polygons schneidet, ist, für jede Ecke P[i]=P(x[i],y[i]) Polygons zu testen, ob
`c*(y[i]-ya) = s*(x[i]-xa)`
gilt. In diesem Fall ist eine andere Richtung (c,s) zu wählen.

Jetzt haben wir einen Strahl, der keine Ecke des Polygons schneidet. Wir ermitteln für jedes i die Ausdrücke

```
d =     c    *(y[i]-y[i+1]) -     s    *(x[i]-x[i+1])
a = (x[i]-xa)*(y[i]-y[i+1]) - (y[i]-ya)*(x[i]-x[i+1])
m =     c    *(y[i]-ya)     -     s    *(x[i]-xa)
```

Für x[n+1] ist hier x[1] einzusetzen. Entsprechendes gilt für y.

Der Strahl schneidet dann die Kante P[i]P[i+1] genau dann wenn:

```
((d>0) and (a>0) and (m>0) and (m<d) ) or
((d<0) and (a<0) and (m<0) and (m>d) )
```

Die Ignorier-Variante

Die zweite Methode erschlägt die Sonderfälle direkt im Algorithmus. Dadurch kommt man mit jedem Strahl zurecht, der von A ausgeht. Man sucht sich dann natürlich dann einen, der parallel zu einer der Koordinatenachsen ist, weil es sich da besonders einfach rechnen läßt.

Die Idee ist, anders zu zählen. Nicht die Anzahl der Schnittpunkte der Kanten mit dem Strahl ist interessant, sondern es wird gezählt, wie oft der Polygonzug die Seite des Strahls wechselt. Das ist zwar fast dasselbe, aber eben nur fast. Dazu umfährt man einfach das Polygon, und ignoriert die Punkte, die auf dem Strahl liegen. Dieses Verfahren ist auch im Sedgewick beschrieben, allerdings hat er da einen kleinen Fehler (Ausgabe 1994).

Voraussetzung ist, daß A nicht auf einer der Kanten liegt. Der Strahl heißt S, und wir definieren eine Gerade G, die S beinhaltet. D.h. S über A hinaus verlängert ist G.

Weiter wird vorausgesetzt, das wenigstens P[1] (oder wo man halt anfängt) nicht auf der Gerade G liegt. Man beachte, daß P[n+1]=P[1];

```
int a=1; int zaehl=0; boolean ignore=false;
for (b=2; b<=(n+1); b++)
{
  if ignore then
    if (P[b] liegt auf G)
      ignore=true
    else
    {
      if (Kante(P[a],P[b]) schneidet G) zaehl++;
      ignore=false;
      a=b;
    }
  else
    if (P[b] liegt auf S) then
      ignore:=true
    else
    {
      if (Kante(P[a],P[b]) schneidet S) zaehl++;
      ignore=false;
      a=b;
    }

}
```

Implementation in Java: Quelltext, Applet

Man beachte den wichtigen Unterschied. Wenn ignore false ist, wird der Strahl betrachtet. Wenn ignore true ist, wird die Gerade betrachtet! Genau das hat Sedgewick vergessen. Läßt man hier diese Unterscheidung weg, kürzt sich der Quellcode auf das zusammen, was im Buch steht. Ansonsten kann man Sedgewick aber wirklich empfehlen.

Der Ansatz für die Ebene (Winkelsummierung)

Man bildet die Summe über die Winkel (P[1],A,P[2]), (P[2],A,P[3]), ..., (P[n],A,P[1]). Das ist also der Gesamtwinkel, um den sich ein um den Punkt A drehbarer 'elastischer' Vektor dreht, wenn seine Spitze das Polygon umrundet.

Ist dieser Gesamtwinkel Null, liegt A außerhalb des Polygons. Ansonsten kommt +/-360 Grad je nach Drehrichtung des Polygons heraus. Die Winkel müssen im offenen Intervall (-180deg, +180deg) liegen. Ist ein Winkel 180deg, liegt A auf der entsprechenden Linie und man kann willkürlich entscheiden.

Variante mit komplexen Zahlen

Um die Rechnung durchzuführen, kann man auf komplexen Zahlen zurückgreifen.

Dabei wird der Gesamtwinkel mit dem Wert des komplexen Kurvenintegrals der Funktion 1/(z-a) über den Polygonzug ausgedrückt, wobei ein Punkt P(x,y) als komplexe Zahl z = x+i*y und der Testpunkt A(xa,ya) als a=xa+i*ya interpretiert wird.

Als erstes wird der Ursprung des Koordinatensystems auf den Testpunkt A verschoben. Die notwendige Transformation für das Polygon lautet: z1:=z1-a; z2:=z2-a; ... ; zn:=zn-a;

Danach summiere man die Ausdrücke

```
atan2(y[i+1]*x[i]-x[i+1]*y[i], x[i+1]*x[i]+y[i+1]*y[i])
```

von i=1..n, wobei für i=n+1 wegen der Geschlossenheit des Polygonzugs wieder i=1 einzusetzen ist.

atan2 sei wie in ANSI C spezifiziert. Ihr Wertebereich ist

```
-pi < atan2 <= pi
```

PI wäre ein Zweifelsfall, wenn A auf eine Kante des Polygons liegt, und sollte laut Voraussetzung hier nicht auftreten.

Variante mit Quadranten

Martin Ahlemeyer hat einen Algorithmus gefunden, der sogar ganz ohne trigonometrische Funktionen auskommt. Ausgenutzt wird, daß nur Endsummen von 0 Grad oder +/- 360 Grad unterschieden werden müssen und daß außerdem eine einzelne Drehung (zwischen zwei Polygonpunkten) immer <= 180 Grad ist, da ja der Betrachterpunkt und die beiden Polygonpunkte entweder ein Dreieck aufspannen oder auf einer Geraden liegen.

Gegeben sei der Beobachterpunkt (bx, by) und ein Polygon mit den Eckpunkten (x1,y1), (x2,y2) ... (xn,yn). Man transformiert durch einfache Verschiebung (Subtraktion von bx bzw. by) das ganze in ein Koordinatensystem, in dem der Beobachterpunkt in (0,0) liegt. (* Macht 2 n Additionen *)

Die kartesische Ebene ist durch die x-Achse und y-Achse in 4 Quadranten geteilt. Jedem Polygonpunkt kann man durch einfache Vorzeichenbetrachtung nun einen der Quadranten zuordnen. Dabei müssen Punkte, die auf den Achsen selbst liegen, jeweils einem Quadranten zugeordnet werden, und zwar so, daß jeder Quadrant genau eine Halbachse mit umfaßt.

Hilfweise verwendet man noch die Geraden y=x und y=-x, die jeweils zwei Quadranten halbieren.

Jetzt kann man die Drehung d pauschalieren. Sei Polygonpunkt (xa,ya) in Quadrant q.

- Wenn der nächste Polygonpunkt im selben Quadranten liegt ==> d := d + 0
- Wenn der nächste Polygonpunkt im Quadranten (q+1) mod 4 ==> d := d + 90
- Wenn der nächste Polygonpunkt im Quadranten (q+3) mod 4 ==> d := d - 90

- Wenn der nächste Polygonpunkt im Quadranten (q+2) mod 4 ==> Fallunterscheidung notwendig
  Betrachtet wird der Hilfspunkt, der sich durch den Schnitt der Geraden durch die beiden Polygonpunkte einerseits und durch die Diagonale, die durch die beiden nicht benutzten Quadranten geht, andererseits ergibt.

  Fall A
  Hilfspunkt = (0,0) ==> Beobachterpunkt liegt auf dem Polygonrand
  Fall B
  Hilfspunkt <> (0,0)
  Unterfall B1
  Hilfspunkt im Quadranten (q+1) mod 4 ==> d := d + 180
  Unterfall B2
  Hilfspunkt im Quadranten (q+3) mod 4 ==> d := d - 180

Am Schluß enthält d entweder 0 oder +/- 360. Netterweise bekommt man den Sonderfall des Polygonrands gleich mitgeliefert.

Auch wenn der Beobachterpunkt in einem Eckpunkt des Polygons liegt, ist das leicht vorab auszutesten.

Implementation in Java: Quelltext, Applet

Schnelle Aussortiertests

Wenn es schnell gehen soll, kann man auch ein paar einfache Tests vorher machen. So etwas lohnt sich vor allem, wenn für das gleiche Polygon viele unterschiedliche Punkte A getestet werden sollen.

- Man bildet das "einhüllende Rechteck" des Polygons. Also einfach den höchsten/niedrigsten X/Y-Wert aller Punkte P(i). Wenn A in diesem Rechteck nicht drin ist, kann A auch nicht im Polygon sein.
- Evtl. lohnt sich auch, vorher die konvexe Hülle des Polygons zu berechnen. Diese Hülle ist wieder ein (konvexes) Polygon. Wenn A dort nicht drin ist, ist A auch nicht im Originalpolygon. Vorteil ist, daß der Test "Punkt im Polygon" für konvexe Polygone wesentlich einfacher ist.

Konvexe Polygone

Dieser Test funktioniert nur für konvexe Polygone!

Der Flächeninhalt eines von drei Punkten A,B,C aufgespannten Dreieckes läßt sich berechnen mit:

$$A = \begin{vmatrix} 1 & ax & ay \\ 1 & bx & by \\ 1 & cx & cy \end{vmatrix} * 0.5$$ (Dies soll 1/2 * Determinante sein)

Ausführlich geschrieben:

```
A = 0.5 ( bx*cy + cx*ay + ax*by - cx*by - ax*cy - bx*ay)
```

Die Formel hat ein Extra. Das Vorzeichen des Flächeninhalt entspricht dem Drehsinn der drei Punkte. Also positiv für linksrum, negativ für rechtsrum.

Wenn man also für A die Koordinaten des Testpunkts einsetzt und für B und C die Endpunkte einer Linie nimmst, dann weiß man ob der Punkt links- oder rechtsseitig von der Linie liegt. Wenn der Flächeninhalt gleich null ist, dann liegt der Punkt auf der Linie oder auf ihrer Verlängerung.

Wenn das Polygon konvex ist, muß der Testpunkt von allen Kanten linksseitig liegen, um innerhalb des Polygons zu sein. Dies setzt allerdings voraus, daß das Polygon mathematisch positiv angeordnet ist. Genau andersherum ist es, wenn das Polygon mathematisch negativ angegeben ist. Damit ist man schon fertig.

Bei sehr vielen Testpunkten, kann man die Berechnung etwas abkürzen, indem man den vom Testpunkt unabhängigen Teil schon vorberechnet.

Beispiel: (ax,ay) ist der Testpunkt / (bx,by) und (cx,cy) soweit bekannt.

Dann können wir vorberechnen:

```
m = bx*cy - cx*by
n = by - cy
o = cx - bx
```

und in jedem Schleifendurchlauf berechnen:

```
A = m + ax*n + ay*o   (den Koeffizienten 0.5 kann man sich schenken)
```

Man kann die Schleife natürlich abbrechen, wenn zum ersten Mal der Testpunkt auf der "falschen" Seite der Kante liegt.

Implementation in Java: Quelltext, Applet

Dankeschön

Wer die Informatik.Ger schon eine Weile liest, dem werden Teile des Textes sehr bekannt vorkommen. Das Ganze ist mehr oder weniger zusammengestückelt aus Mails von Horst Krämer, Tim Schattowsky, Alexander Berndt, Martin Ahlemeyer und meiner Wenigkeit. Ich habe mir erlaubt, das Ganze zu sortieren, ein paar Tippfehler zu Ex-en, und die Bezeichnungen anzugleichen. Ich hoffe, das nimmt mir niemand übel.

Erstellt von Ralf Wiebicke, letzte Änderung 02.08.97