Point Location Problem.

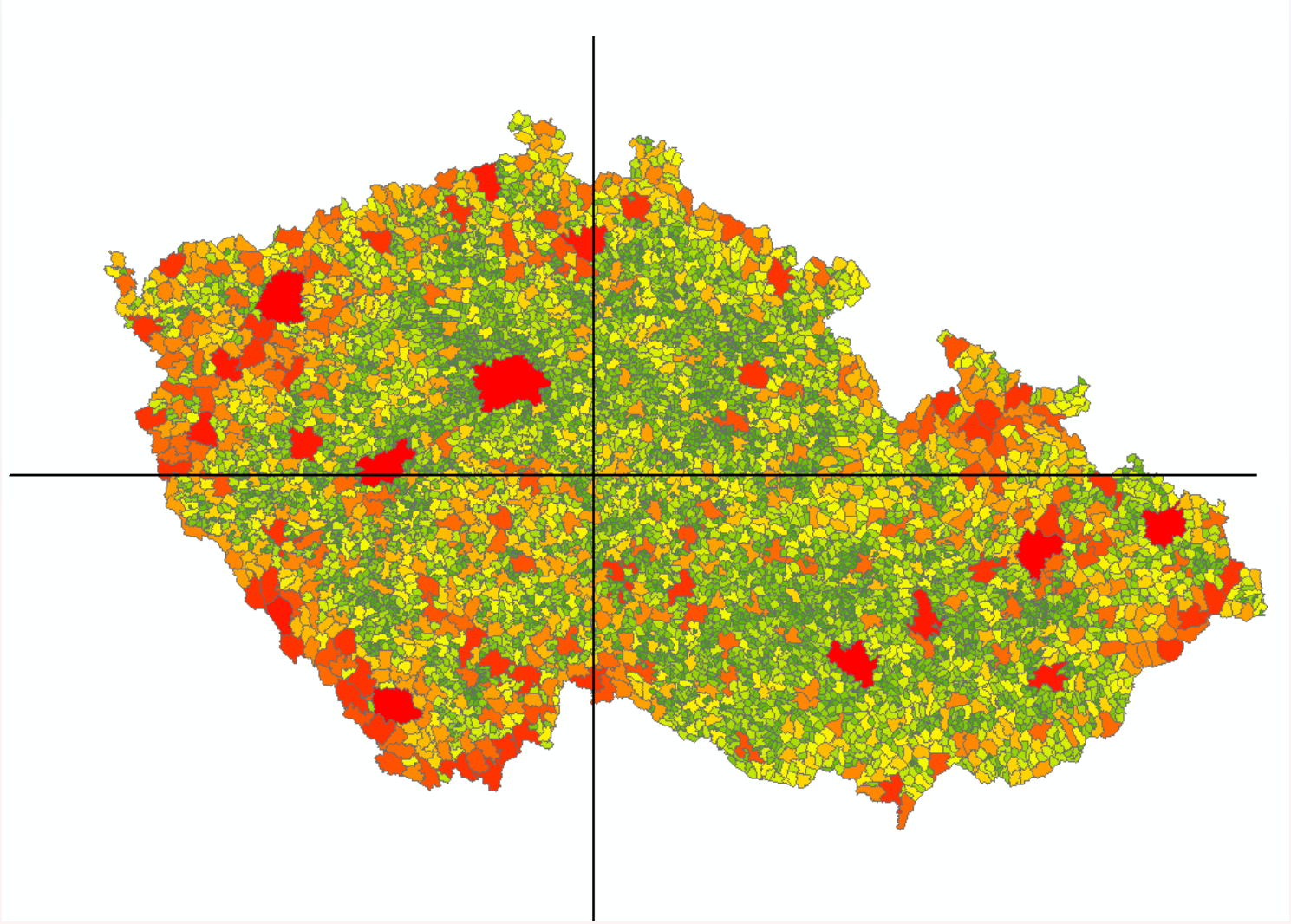
Konvexní a nekonvexní oblasti. Ray algoritmus. Winding Number algoritmus.

Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie. Přírodovědecká fakulta UK.

Obsah přednášky

1. Ve kterém mnohoúhelníku leží kurzor?



Data set tvoří 6000 mnohoúhelníků a cca 1 milión bodů.

2. Formulace problému

Dáno:	V rovině dána množina n bodů $\{p_i\}$ tvořících vrcholy m mnohoúhelníků $\{P_j\}$ a bod q.
Hledáme:	Mnohoúhelník P obsahující bod $q \Rightarrow$ tzv. *Point Location Problem*.

Problém často řešen v GIS.

Mapa představuje planární rozdělení roviny do regionů, bod q např. polohu kurzoru.

V praktickém životě často potřebujeme znát svoji polohu vzhledem k jiným objektům.

Rychlé nalezení hledaného mnohoúhelníku nezbytné.

Data sety mohou být tvořeny stovkami tisíc i miliony mnohoúhelníků.

Nutno nalézt datové struktury a algoritmy umožňující efektivně řešit pro konvexní i nekonvexní mnohoúhelníky.

Použití: triangulace nekonvexních oblastí, množinové operace, prostorová interpolace.

3. Techniky řešení problému

Techniky řešení:

- *Převedení problému na vztah bodu a mnohoúhelníku*
  Problém převeden na úlohu opakovaného určení polohy bodu vzhledem k mnohoúhelníku.
  Snadná implementace, avšak pomalejší.
- *Planární dělení roviny*
  Rovina rozdělena na množinu pásů či lichoběžníků (trapezoids).
  Vzniká *trapezoidální mapa*.
  Výrazně rychlejší vyhledání mnohoúhelníku, implementace obtížná.
  Pro vyhledávání používány speciální datové struktury (binární stromy).

Varianty řešení:

- Nekonvexní mnohoúhelníky: nejčastější varianta.
- Konvexní mnohoúhelníky: speciální případ, Delaunay triangulace, Voronoi diagram.

4. Princip testu polohy bodu a mnohoúhelníku

Lokální procedura. Opakované určení polohy q vzhledem k mnohoúhelníku.

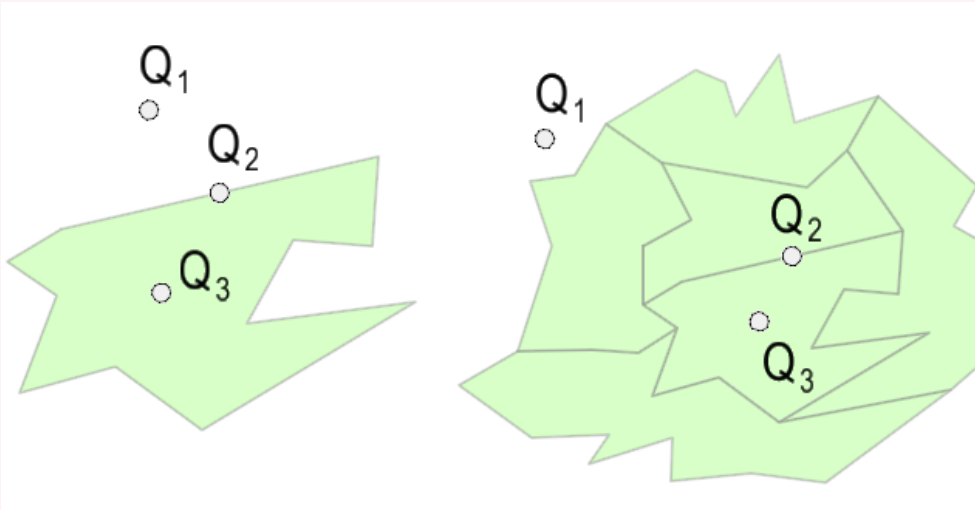
Výsledek lokální procedury:

- Bod q leží uvnitř testovaného mnohoúhelníku: $q \in P$.
- Bod q leží vně testovaného mnohoúhelníku: $q \notin P$.
- Bod q leží na hraně testovaného mnohoúhelníku: $q \in \partial P$.

Globální procedura. Lokální procedura opakovaná pro $\forall$ mnohoúhelník P.

Výsledek globální procedury:

- Bod q leží vně všech mnohoúhelníků
- Bod q leží uvnitř konkrétního mnohoúhelníku.
- Bod q leží na hraně/uzlu mnohoúhelníku (inciduje s více mnhoúhelníky).



5. Řešení pro konvexní mnohoúhelníky

Lokální procedura realizována nejčastěji dvěma způsoby:

- Test polohy bodu vůči každé hraně mnohoúhelníku (opakovaný Half-plane test).
- Paprskovým algoritmem (Ray Crossing Algorithm).

Opakovaný Half-plane test:

Bez předzpracování.

Na každou stranu mnohoúhelníku (p_i, p_{i+1}) a bod q aplikován test.

Výsledky všech testů stejné $\Leftrightarrow$ $q \in P$.

Algoritmus 1: Test polohy q vzhledem k P (HalfPlane test).

1: Opakuj pro $\forall$ stranu (p_i, p_{i+1}):
2: Urči orientaci o_i bodu q ke straně (p_i, p_{i+1}).
3: if $q \in (p_i, p_{i+1})$, bod $q \in \partial P$.
4: if $o_i \neq o_{i-1}$, bod $q \notin P$.
5: Bod $q \in P$.

Nelze použít pro nekonvexní mnohoúhelníky.

Složitost $O(N)$.

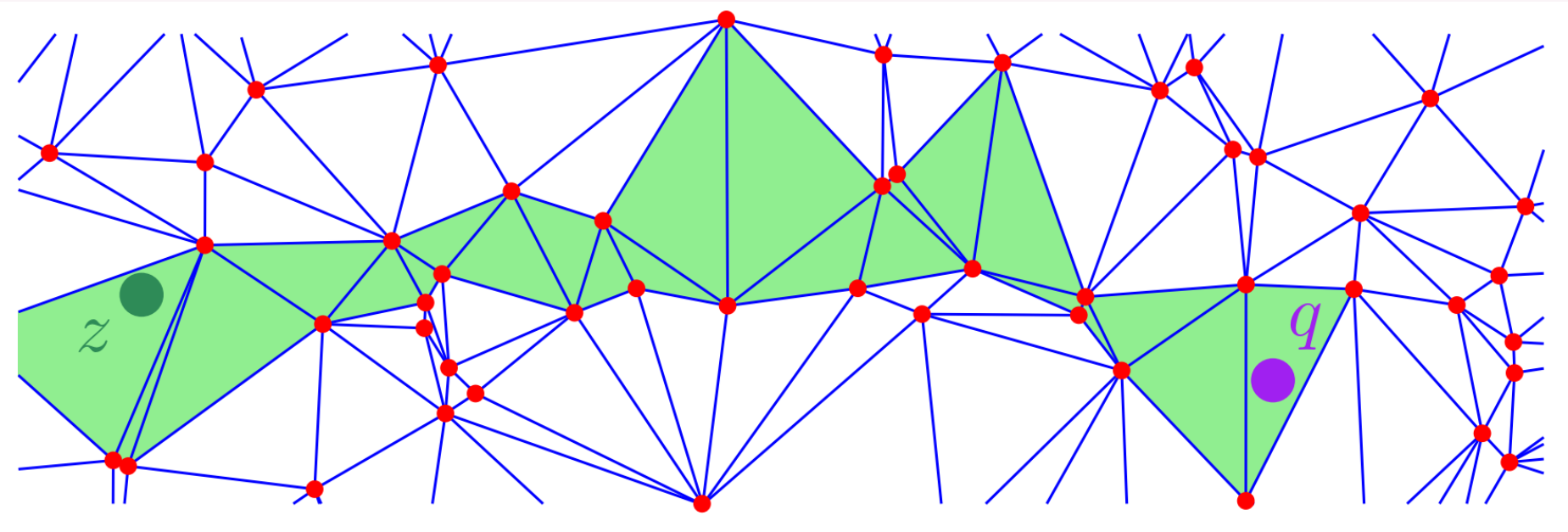
6. Aplikace v triangulačních algoritmech

Rychlé nalezení trojúhelníku obsahujícího bod.

Důležité při konstrukci planárních struktur (DT).
Kombinace s heuristikami, grafovými algoritmy: Walking Algorithms.

Postupná cesta přes sousedící trojúhelníky.

Nejefektivnější procházka, Lawson (1977).



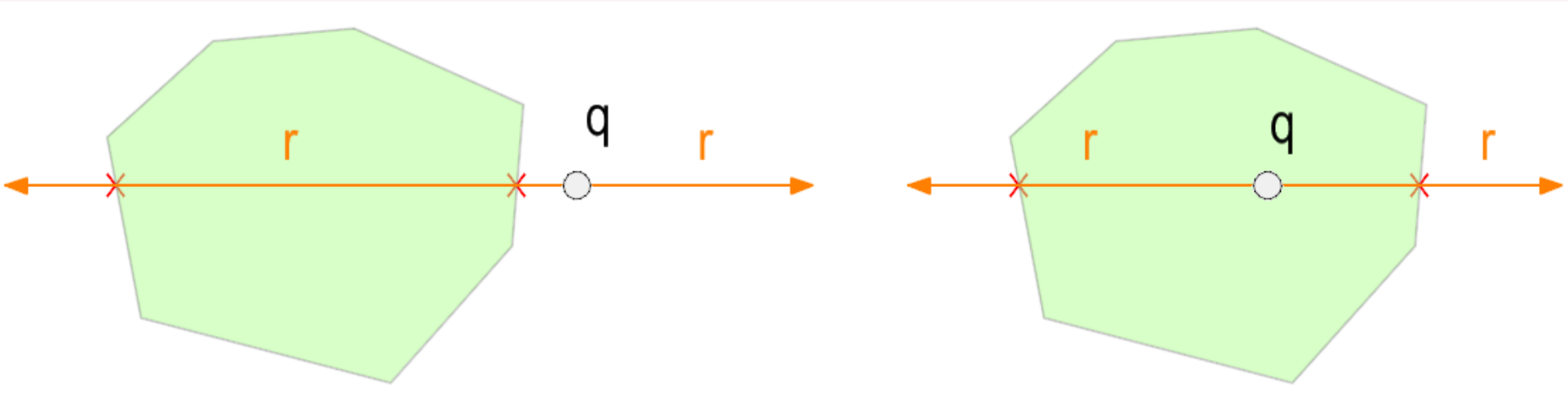
(Devillers, 2016).

7. Ray Crossing Method

Bodem q vedena polopřímka r (paprsek, tzv. Ray).

Počet průsečíků k paprsku r a oblasti P

$$k(q, P) = \begin{cases} 1, & q \in P, \\ 0 \vee 2, & q \notin P. \end{cases}$$



Algoritmus invariantní vůči volbě r, často volena $y = y_q$.

Pozor na singulární případy:

- r prochází vrcholem $p_i \in P$ (protíná 2 segmenty).
- r kolineární se stranou/stranami P (nesprávný počet průsečíků).

Lze zobecnit pro nekonvexní mnohoúhelníky.

8. Řešení pro nekonvexní mnohoúhelníky

Používány dva algoritmy:

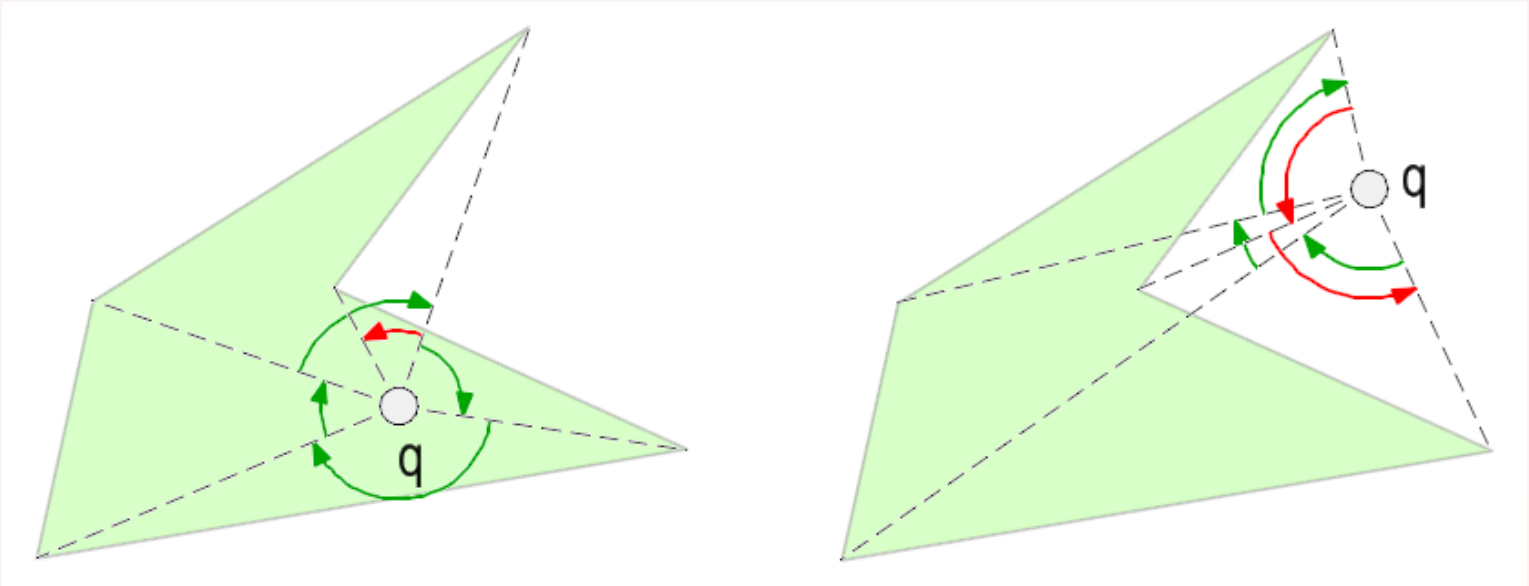
- Paprskový (Ray Algorithm).
- Metoda oviјení (Winding Number Algorithm).

Winding Number Algorithm

Pozorovatel stojí v bodě q.

Pokud $q \in P$ a pozorovatel chce vidět $\forall p_i \in P$ musí se otočit o úhel 2π.

Pokud $q \notin P$, je tento úhel menší než 2π.



9. Výpočet Winding Number Ω

Suma Ω všech rotací ω_i (měřená CCW)

$$\Omega(q, P) = \frac{1}{2\pi} \sum_{i=1}^{n} \omega(p_i, q, p_{i+1}),$$

které musí průvodič (q, p_i) opsat nad všemi body $p_i \in P$.

Přímka $p(p_i, p_{i+1})$ dělí σ na σ_l, σ_r (Left/Right Halfplane)

$$\sigma_l = \{[x; y], x \in \mathbb{R}, y \in \mathbb{R} | x > x_r, y > y_r\},$$
$$\sigma_r = \{[x; y], x \in \mathbb{R}, y \in \mathbb{R} | x < x_r, y < y_r\}.$$

Úhly $\omega(p_i, q, p_{i+1})$ orientované

$$\omega(p_i, q, p_{i+1}) = \begin{cases} +\omega(p_i, q, p_{i+1}), & q \in \sigma_l, \\ -\omega(p_i, q, p_{i+1}), & q \in \sigma_r. \end{cases}$$

Případ 1: Úhel $\omega(p_i, q, p_{i+1})$ má CW orientaci.

Případ 2: Úhel $\omega(p_i, q, p_{i+1})$ má CCW orientaci.

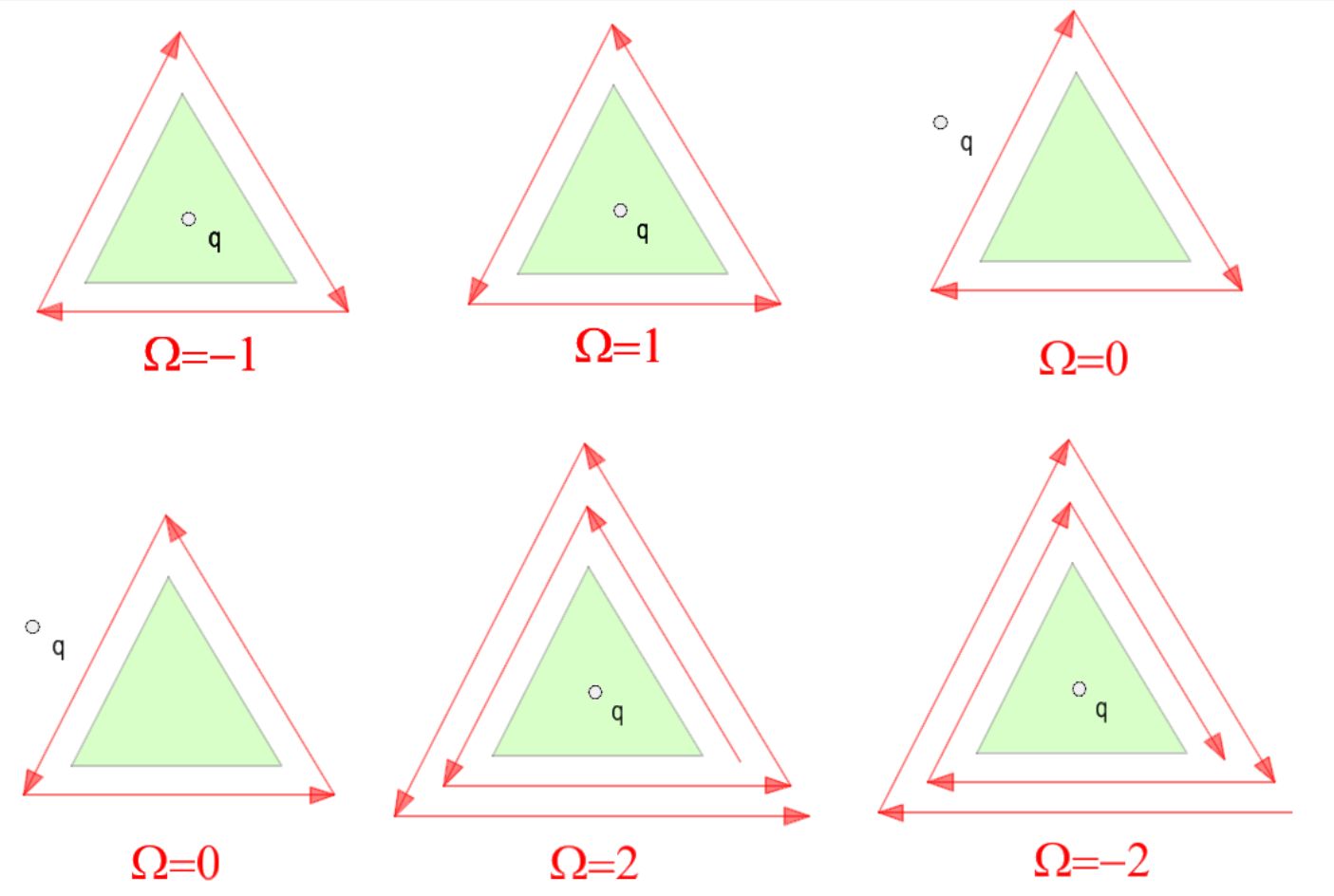
Winding Number

$$\Omega(q, P) = \begin{cases} 1, & q \in P, \\ 0, & q \notin P. \end{cases}$$

10. Winding number Ω

Hodnota Ω uváděna v počtech oběhů, tj. v násobcích 2π.

Výsledek znaménkově závislý na směru pohybu.



11. Winding Number Algorithm

Algoritmus 2: Winding Algorithm

1: Inicializuj $\Omega = 0$, tolerance ε.
2: Opakuj pro $\forall$ trojici (p_i, q, p_{i+1}):
3: Urči polohu q vzhledem k $p = (p_i, p_{i+1})$.
4: Urči úhel $\omega_i = \angle p_i, q, p_{i+1}$.
5: If $q \in \overline{\sigma}_l$, pak $\Omega = \Omega + \omega_i$. //Bod v leve polorovine
6: else $\Omega = \Omega - \omega_i$. //Bod v prave polorovine
7: if $\left|\left|\Omega\right| - 2\pi\right| < \varepsilon$, pak $q \in P$ //Test na odchylku od 2π
8: else $q \notin P$

Postačuje výpočet $\sum \omega$, 2π konstanta.

\+ Lepší ošetření singulárních případů než u paprskového algoritmu.

\+ Pomalejší než paprskový algoritmus.

\- Problematický případ $q \equiv p_i$.

\- Nutnost předzpracování $O(N)$.

Složitost $O(N)$.

12. Ray Crossing Method

Bodem q vedena polopřímka r (paprsek, tzv. ray)

$$r(q) : y = y_q.$$

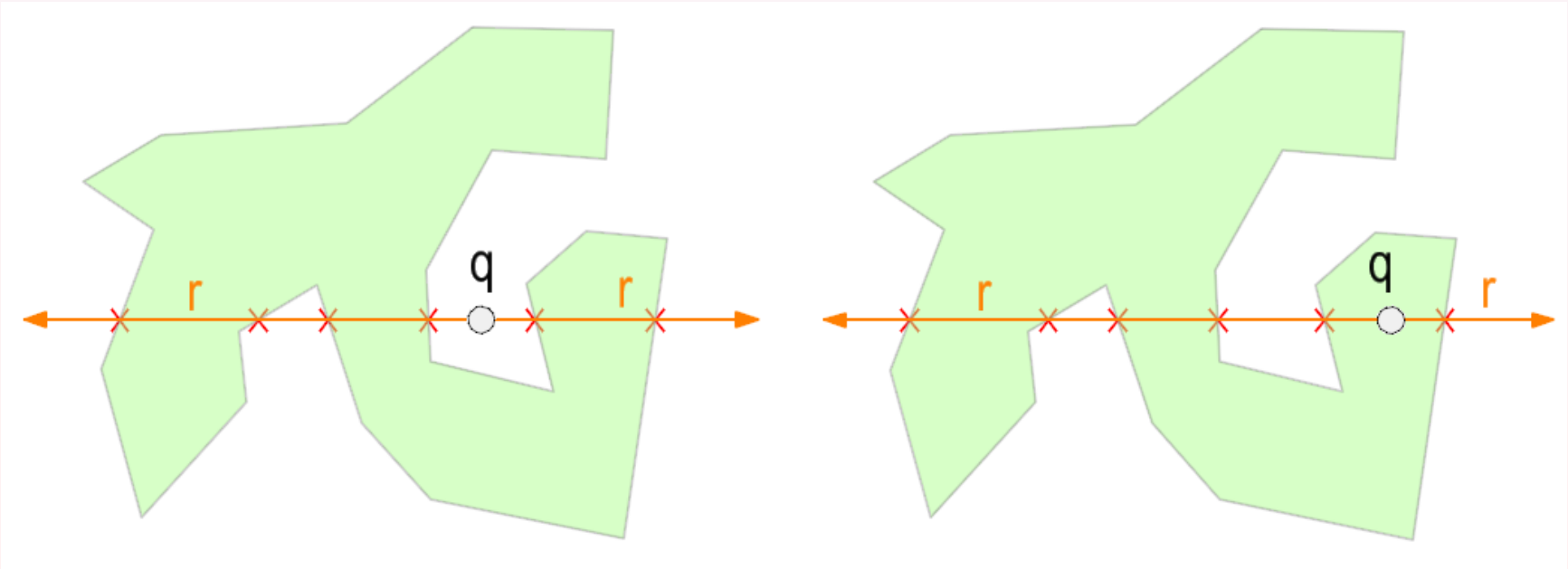
Invariance vůči směru r.

Počet průsečíků přímky r s oblastí P

$$k\%2 = \begin{cases} 1, & q \in P, \\ 0, & q \notin P. \end{cases}$$

\+ O řád rychlejší než Winding Number.

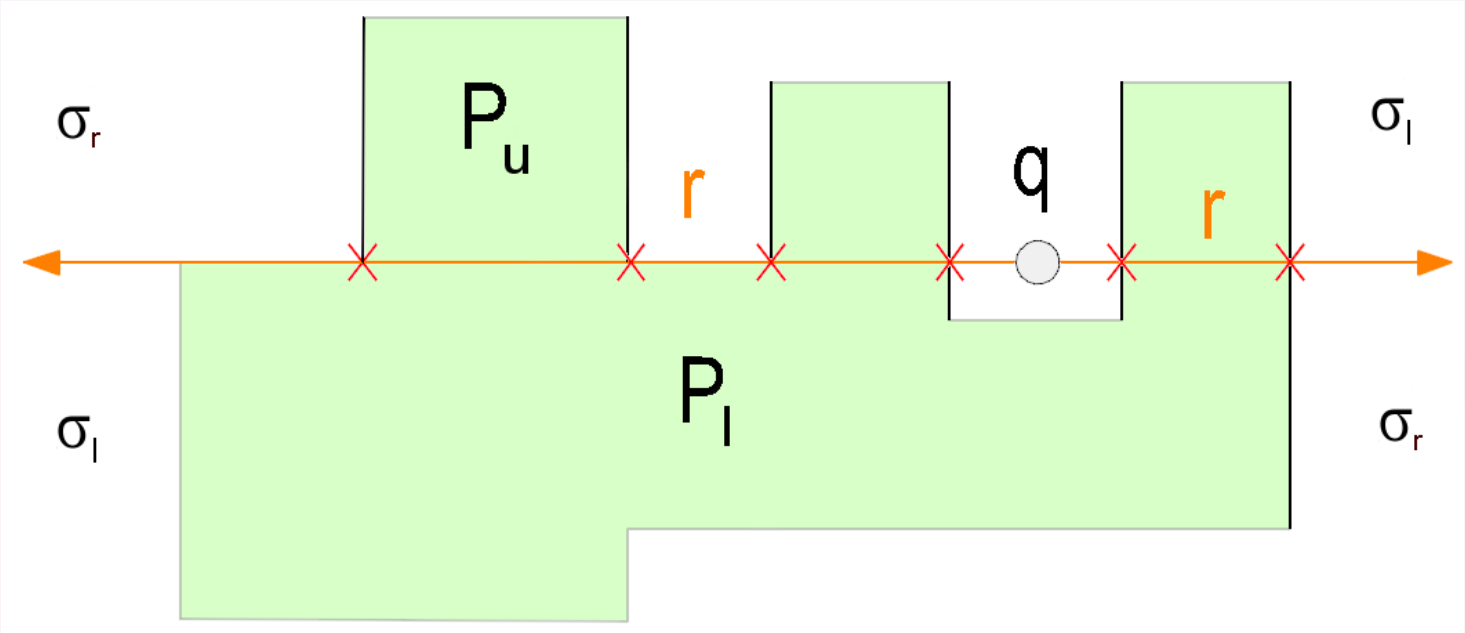
\- Problémem singularity.



13. Upravená varianta Ray Crossing

Eliminace singularit: $r(q)$ prochází vrcholem P nebo hranou.

Přímka definovaná $r(q)$ dělí σ na σ_u, σ_l.

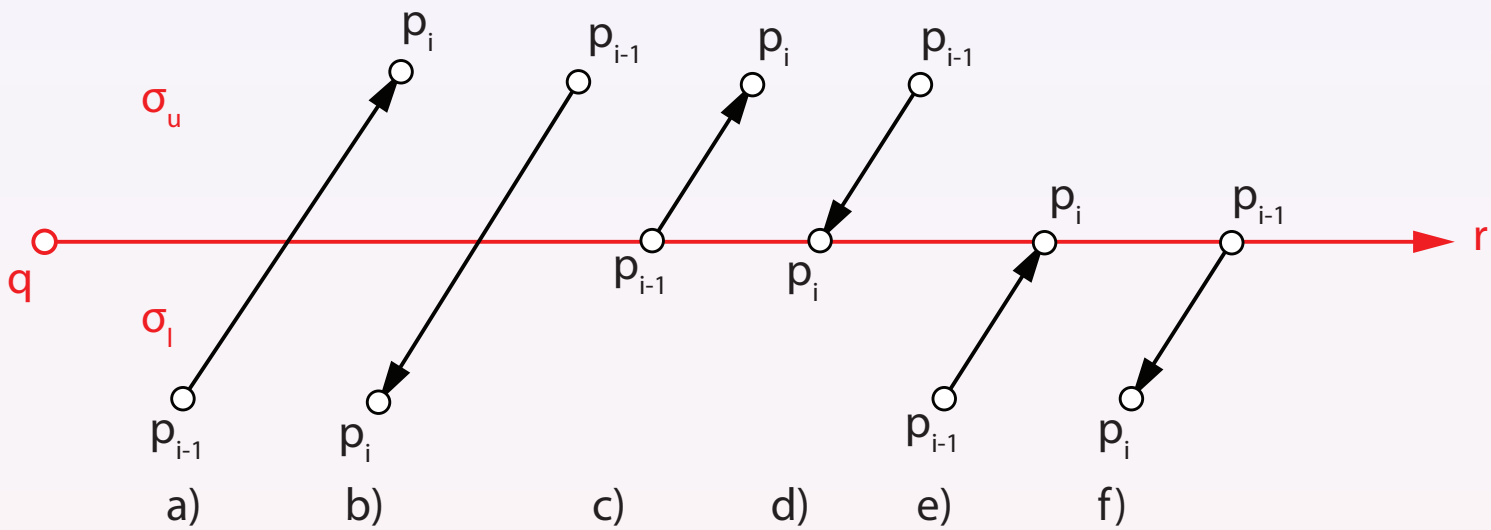


Modifikace Ray Crossing:

Inkrementace $k(q, P)$:

- jeden z bodů hrany nad $r(q)$ (v σ_l) a druhý z bodů na/pod $r(q)$.
- Kolineární segmenty nejsou započítávány.

14. Inkrementace počtu průsečíků



Průsečík M hrany (p_{i-1}, p_i) a $r(q)$ započítáváme, pokud:

1. *Hrana v obou polorovinách nebo jen v horní*
   Bod p_i nad paprskem, bod p_{i-1} na nebo pod paprskem: situace a), c).
   Bod p_{i-1} nad paprskem, p_i na nebo pod paprskem: situace b), d).
   Kritérium
   $$(y_i > y_q) \wedge (y_{i-1} \leq y_q) \vee (y_{i-1} > y_q) \wedge (y_i \leq y_q).$$

2. *Hrana v obou polorovinách nebo jen v dolní*
   Bod p_i pod paprskem, p_{i-1} na nebo nad paprskem: situace a), e).
   Bod p_{i-1} pod paprskem, p_i na nebo nad paprskem: situace b), f).
   Kritérium
   $$(y_i < y_q) \wedge (y_{i-1} \geq y_q) \vee (y_{i-1} < y_q) \wedge (y_i \geq y_q).$$

   V praxi použijeme jednu z variant, nikoliv obě.

Podmínku lze přepsat do tvaru

$$y_i > y_q \neq y_{i-1} > y_q.$$

15. Varianta s redukcí ke q (1/2)

Lokální souřadnicový systém (q, x', y'), zjednodušení výsledných vztahů.

Posunutí počátku hodnotu q.

Rovnice paprsku $r(q) \equiv x'$

$$y' = 0.$$

Redukce bodů p_i ke q

$$x'_i = x_i - x_q, \qquad y'_i = y_i - y_q.$$

Hledáme průsečíky M paprsku $r(q)$ a (p_{i-1}, p_i) ležící v σ_u.

Výhody:

+ Snadnější detekce hran (p_{i-1}, p_i) protínajících $r(q)$.
+ Jednodušší výpočet průsečíku M.
+ Lepší numerická stabilita.
+ Nezávislost na orientaci P (CW nebo CCW).

Nevýhody:

- Nelze detekovat stav, kdy $q \in \partial P$.

16. Varianta s redukcí ke q (2/2)

Přímka p

$$q : y = kx + q,$$

prochází body p_{i-1}, p_i

$$y_{i-1} = kx_{i-1} + q, \qquad y_i = kx_i + q.$$

Z rozdílu obou rovnic

$$y - y_{i-1} = k(x - x_{i-1}), \qquad k = \frac{y_i - y_{i-1}}{x_i - x_{i-1}},$$

určíme souřadnici x

$$x = \frac{(y - y_{i-1})(x_i - x_{i-1})}{y_i - y_{i-1}} + x_{i-1}.$$

Systém (q, x', y'): průsečík segmentu a paprsku $r(q)$ (osa x') bod $M = [x'_m, y'_m = 0]$

$$x'_m = \frac{-y'_{i-1}(x'_i - x'_{i-1})}{y'_i - y'_{i-1}} + x'_{i-1} = \frac{x'_i y'_{i-1} - x'_{i-1} y'_i}{y'_i - y'_{i-1}}.$$

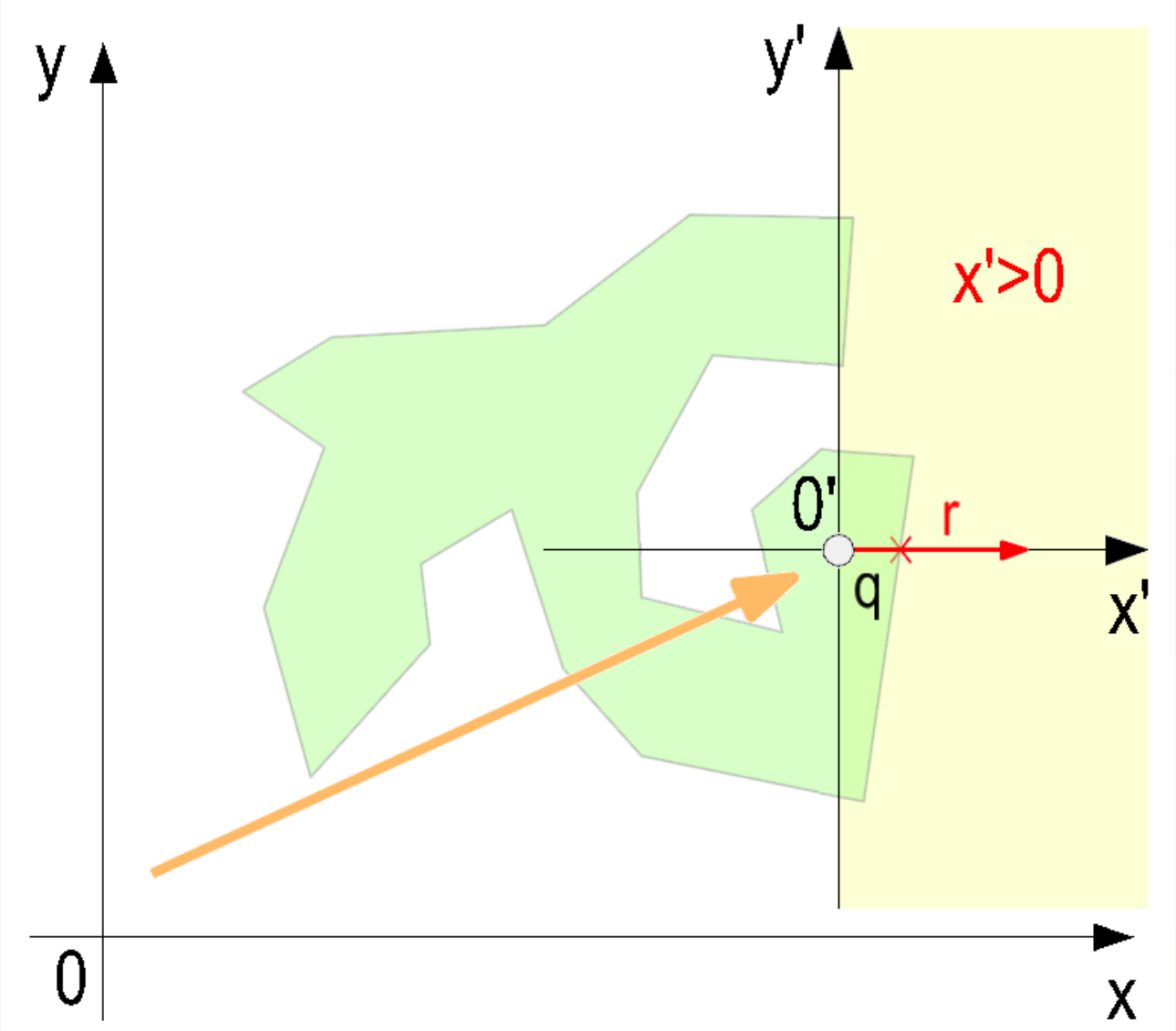
Test existence průsečíku p a paprsku $r(q)$

$$(y'_i > 0) \wedge (y'_{i-1} <= 0) \vee (y'_{i-1} > 0) \wedge (y'_i <= 0).$$

Započítány pouze průsečíky M v ***pravé*** polorovině vzhledem ke q

$$k = \begin{cases} k + 1, & x'_m > 0, \\ k & x'_m \leq 0. \end{cases}$$

17. Ilustrace varianty s redukcí ke q



18. Implementace Ray Crossing s redukcí ke q

Algoritmus 3: Ray Crossing Algorithm ($P = \{p_1, ..., p_n\}, q$)

1. Inicializuj $k = 0$ //Pocet prueciku
2: Opakuj pro $\forall$ body $p_i \in P$:
3: $x'_i = x_i - x_q$.
4: $y'_i = y_i - y_q$.
5: if $(y'_i > 0)\&\&(y'_{i-1} <= 0)||(y'_{i-1} > 0)\&\&(y'_i <= 0)$. //Vhodny segment
6: $x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i)/(y'_i - y'_{i-1})$. //Vhodny prusecik
7: if $(x'_m > 0)$ pak $k = k + 1$.
8: if $(k\%2) \neq 0$ pak $q \in P$
9: else $q \notin P$

Výhodou snadná implementace.

Nevýhoda: Špatné přiřazení bodu p_i do oblasti P

$$|y_i - c| \leq \varepsilon,$$

tj. p_i blízko paprsku r nebo strana příliš krátká.

19. Detekce stavu $q \in \partial P$

Stávající varianta neumí detekovat stav $q \in \partial P$.

Nutno použít dva paprsky r_1, r_2 s opačnou orientací: r_1 levostranný, r_2 pravostranný.

Udržujeme počet levostranných a pravostranných průsečíků k_l, k_r.

Myšlenka: pokud $q \in \partial P$, počet levostranných a pravostranných průsečíků různý.

Kolineární segmenty:

Levostranný paprsek pracuje s levou polorovinou (Lower)

$$t_l = y_i < y_q \neq y_{i-1} < y_q.$$

Pravostranný s levou polorovinou (Upper)

$$t_u = y_i > y_q \neq y_{i-1} > y_q.$$

Započtení průsečíků:

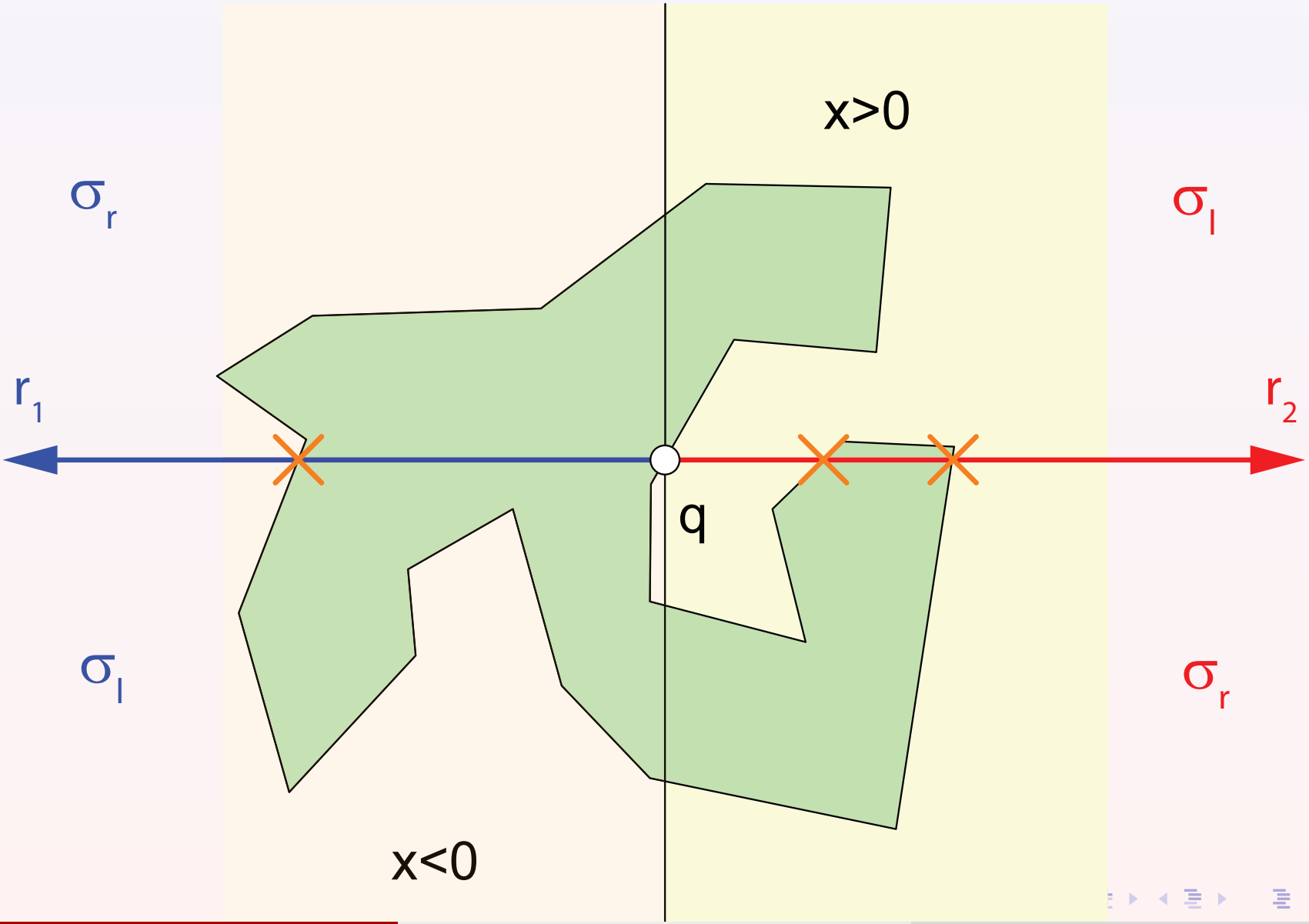
Pro každý paprsek zvlášť:

- Pokud t_l true (levostranný paprsek) a $x'_m < 0$, inkrementujeme k_l.
- Pokud t_u true (pravostranný paprsek) a $x'_m > 0$, inkrementujeme k_r.

Pak

$$q = \begin{cases} \in \partial P, & k_l \% 2 \neq k_r \% 2, \\ \in P & k_r \% 2 = 1 \\ \notin P, & \text{jinak.} \end{cases}$$

20. Detekce stavu $q \in \partial P$, ilustrace



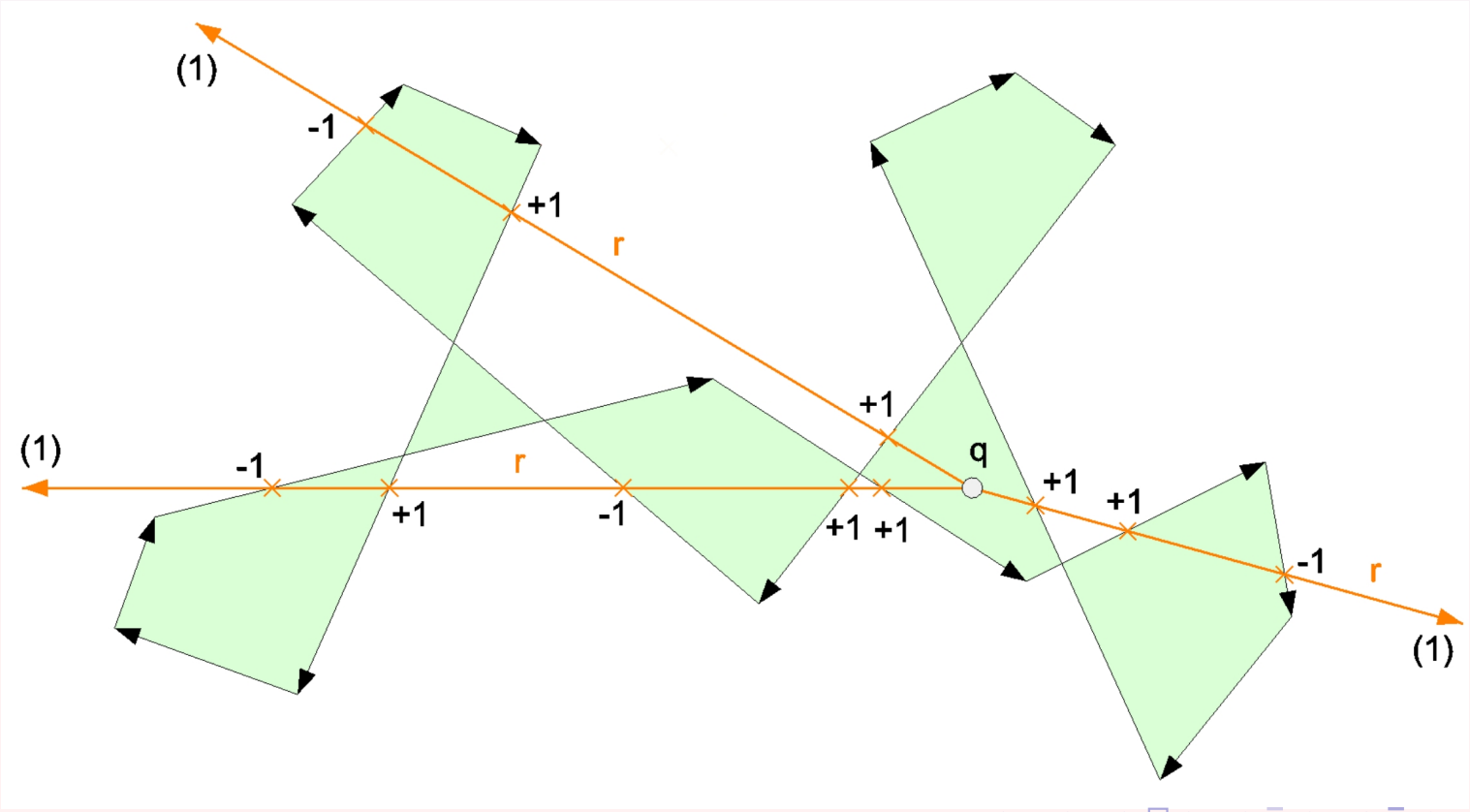
21. Použití u non-simple oblastí

Algoritmy lze použít i pro non-simple polygony: topologicky nekorektní.

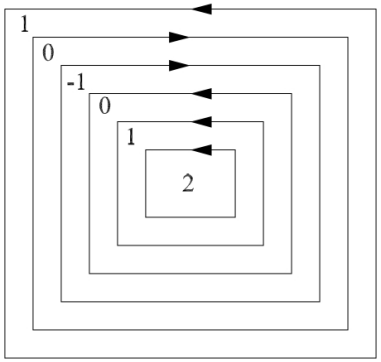
Kombinace RA a WA: orientace segmentů vůči paprsku.

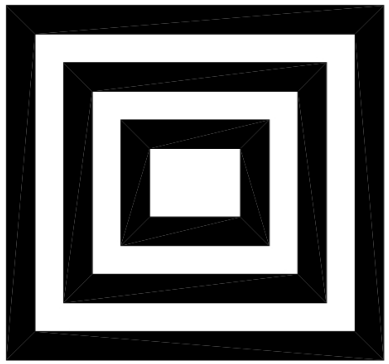
Redefinice $\omega(q, e_i)$, paprsek $r(q)$

$$\omega(q, e_i) = \begin{cases} 0, & r \wedge e_i = \emptyset, \\ 1, & p_i, q, p_{i+1} \text{ CCW z } q \ , \\ -1, & p_i, q, p_{i+1} \text{ CW z } q. \end{cases}$$

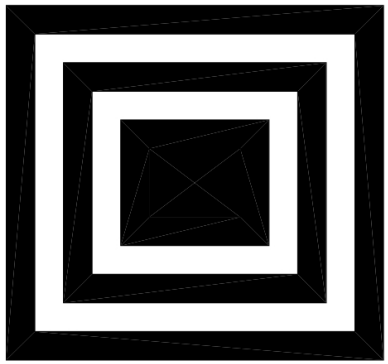


22. Vnořené polygony

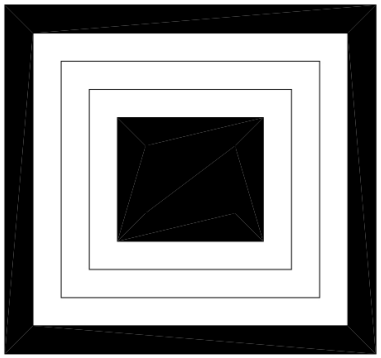




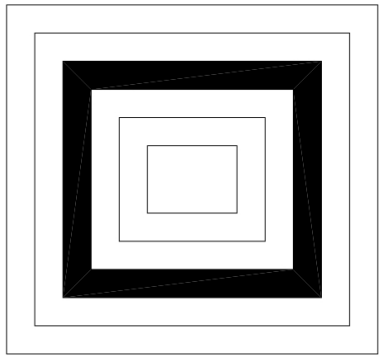
Ω liché



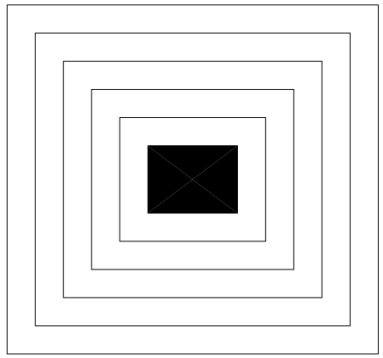
Ω <> 0



Ω > 0



Ω < 0



Ω >= 2