

Računarska grafika

Geometrijski odnosi



Odnos dve tačke prema pravoj

- Cilj je utvrditi da li su dve tačke
 - sa iste strane, ili
 - sa suprotnih strana jedne geometrijske prave
- Polazi se od jednačine prave l kroz dve tačke (x_1, y_1) i (x_2, y_2) :

$$y = [(y_2 - y_1) / (x_2 - x_1)] (x - x_1) + y_1, \text{ ili}$$

$$f(p, l) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) = 0,$$

$$\text{za } p(x, y) \in l$$

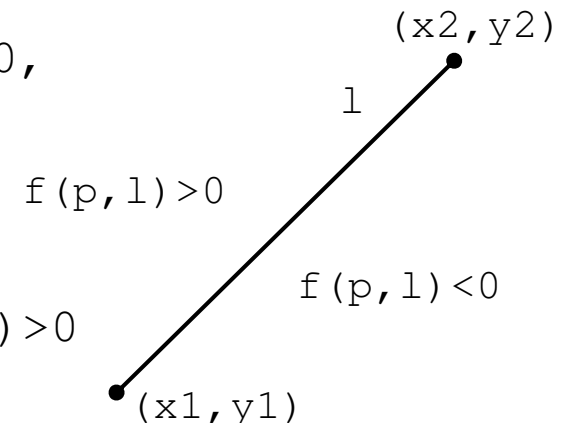
- Ova prava deli ravan na dve poluravni

- u jednoj važi:

$$f(p, l) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) > 0$$

- a u drugoj važi:

$$f(p, l) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) < 0$$



Izvođenje

- Zamenjujući x i y za tačke p_1 i p_2 (čiji odnos prema pravoj l ispitujemo) i množeći odgovarajuće izraze, dobija se izraz:

$$g(p_1, p_2, l) = f(p_1, l) * f(p_2, l) = \\ ((x_2 - x_1)(p_1.y - y_1) - (y_2 - y_1)(p_1.x - x_1)) * \\ ((x_2 - x_1)(p_2.y - y_1) - (y_2 - y_1)(p_2.x - x_1)), \text{ koji je:}$$

- > 0, ako su tačke sa iste strane prave linije
 - < 0, ako su tačke sa suprotnih strana linije i
 - = 0, ako je barem jedna tačka na liniji
- Ako se odnos određuje u celobrojnim koordinatama
 - ovde može da se javi problem prekoračenja pri množenju
 - Pošto je od interesa samo znak, a ne i vrednost proizvoda uvodimo:

```
Type Sign=-1..1;  
Function Sgn(x: LongInt):Sign;  
Begin If x=0 Then Sgn:= 0 Else If x>0 Then Sgn:= 1 Else Sgn:= -1 End;
```

Realizacija SameSide

- Funkcija koja određuje odnos tačaka p1 i p2 prema pravoj liniji l:

```
Function SameSide(l: Line; p1,p2: Point): Sign;
  Var dx,dx1,dx2,dy,dy1,dy2: LongInt;
  Begin
    dx:= l.p2.x - l.p1.x;      dy:= l.p2.y - l.p1.y;
    dx1:= p1.x - l.p1.x;      dy1:= p1.y - l.p1.y;
    dx2:= p2.x - l.p1.x;      dy2:= p2.y - l.p1.y;
    SameSide:= Sgn(dx*dy1 - dy*dx1) *
               Sgn(dx*dy2 - dy*dx2)
  End;
```

Presek pravolinijskih segmenata

- Potrebno je utvrditi da li se dva pravolinijska segmenta (linije) l_1 i l_2 seku
- Direktan način (skupo izračunavanje) da se odredi da li se linije seku:
 - naći tačku preseka geometrijskih pravih
 - proveriti da li se tačka nalazi između krajnjih tačaka datih segmenata
- Ovde nije od interesa tačka preseka
- Postojanje preseka može da se utvrdi korišćenjem `SameSide` funkcije:

```
Function Intersect(l1, l2: Line): Boolean;  
  Begin  
    Intersect := (SameSide(l1, l2.p1, l2.p2) <= 0) And  
                (SameSide(l2, l1.p1, l1.p2) <= 0)  
  End;
```

- Napomene:
 - proglašava se da se dve linije od kojih jedna leži na drugoj seku
 - funkcija proglašava čak i da se dve kolinearne linije seku (greška)

Odnos tačke i poligona

- Problem: da li je tačka unutar ili izvan poligona?
- Ideja rešenja:
 - iz tačke koja se ispituje povuče se linija u bilo kom pravcu do beskonačnosti
 - ako linija seče poligon neparan broj puta - tačka je unutar poligona, inače je izvan
- Algoritam:
 - "beskonačnost" se uzima u pravcu i smeru pozitivne X-ose, pa se ispitna linija (`try`) povuče iz ispitivane tačke (`p`) paralelno sa X-osom
 - u iterativnom postupku, ide se od jedne do druge stranice (`edge`) poligona (`py`) i ispituje da li se ova seče sa ispitnom linijom
 - ako se seče, inkrementira se broj preseka (`count`)
 - na kraju se odlučuje da je tačka `p` unutar `py` ukoliko je `count` neparan
- Pretpostavka – na kraju niza temena se nalazi ponovljeno prvo teme:

```
Type Poly = Array[1..MaxVertex] of Point;  
p[n+1]=p[1];
```

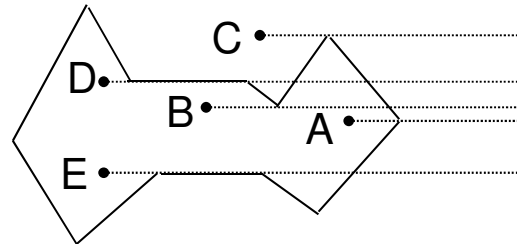
Realizacija Inside

```
Function Inside(p: Point; n: Integer; py: Poly): Boolean;
  Var
    i, count: Integer;  infin: Point;  try, edge: Line;
  Begin
    infin.x:= Max_X;    infin.y:= p.y;
    try.p1:= p;         try.p2:= infin;  count:= 0;
    For i:= 1 to n do
      Begin
        edge.p1:= py[i]; edge.p2:= py[i+1];
        If Intersect(try, edge) Then count:= count + 1;
      End;
    Inside:= Odd(count)
  End;
```

- Još efikasnije: da se uvede Boolean promenljiva `ins`, inicijalno `false`, pa se invertuje na svakom preseku => nema potrebe za funkcijom `Odd()`

Problemi

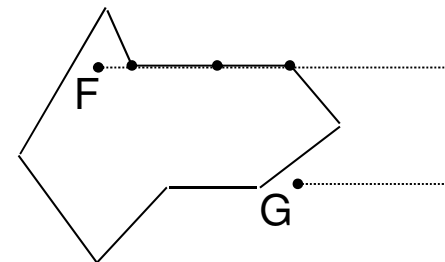
- Problem:
 - ova procedura nije korektna za neke slučajeve kada ispitna linija prolazi kroz teme poligona
- Posmatraju se sledeći slučajevi:



- Za slučajeve D i E se ignoriše presek ispitne linije sa kolinearnom stranicom
 - (A) ispitna linija iz unutrašnjosti poligona prolazi kroz teme i nastavlja u spoljašnjosti
⇒ `count` se inkrementira dva puta: **greška**
 - (B) ispitna linija iz unutrašnjosti prolazi kroz teme i ostaje u unutrašnjosti
⇒ `count` se inkrementira dva puta: **nije greška**
 - (C) ispitna linija iz spoljašnjosti prolazi kroz teme i ostaje u spoljašnjosti
⇒ `count` se inkrementira dva puta: **nije greška**
 - (D) ispitna linija iz unutrašnjosti prolazi kroz dva temena i ignorisanu stranicu koja ih spaja, te nastavlja u spoljašnjosti ⇒ `count` se inkrementira dva puta: **greška**
 - (E) ispitna linija iz unutrašnjosti prolazi kroz dva temena i ignorisanu stranicu koja ih spaja i ostaje u unutrašnjosti ⇒ `count` se inkrementira dva puta: **nije greška**
 - Slučaj D se svodi na slučaj A, a slučaj E na slučaj B

Korekcija greške

- Ideja – korekcija greške u posmatranim slučajevima:
 - za slučajeve A, B i C:
 - posmatrati odnos temena iza i temena ispred kritičnog temena, u odnosu na liniju t_{ry} :
 - ako su sa iste strane (B i C), sve je u redu
 - ako su susedna temena sa različitih strana (A) treba da se dekrementira $count$
 - slučajevi D i E treba da se svedu na A i B, respektivno:
 - treba da se ignorišu horizontalne ivice (ne treba da se odbrojavaju preseci)
 - treba da se analizira odnos temena ispred horizontalne ivice i temena iza te ivice sa linijom t_{ry}
- Navedena tehnika rešava i probleme:
 - više povezanih horizontalnih ivica (F)
 - lažnih preseka koje vraća $Intersect$ za kolinearne ivice sa linijom t_{ry} (G)



Korigovana realizacija (1)

```
Function Inside(p: Point; n: Integer; py: Poly): Boolean;
  Var
    i, count, prev: Integer;
    infin: Point;
    try, edge: Line;
  Begin
    infin.x:= Max_X;    infin.y:= p.y;
    try.p1:= p;        try.p2:= infin;
    count:= 0;         prev:=n;
    For i:=1 to n do
      Begin
        edge.p1:= py[i];
        edge.p2:= py[i+1];
```

Korigovana realizacija (2)

```
    If (p.y<>edge.p1.y) or (p.y<>edge.p2.y) Then
      Begin {not Collinear(try,edge)}
        If Intersect(try, edge) Then
          Begin
            count:= count + 1;
            If (py[i].y = p.y) and
              (SameSide(try,py[prev],py[i+1]) < 0)
              Then count:= count - 1
              {tačke su na suprotnim stranama}
          End;
          prev:=i;
        End
      End; {For}
      Inside:= Odd(count)
    End; {Inside}
```

Povećanje efikasnosti (1)

- Data implementacija je korektna, ali može da se učini nešto efikasnijom
- Funkcije `SameSide` i `Intersect` su generalne i ne koriste činjenicu da je `try` horizontalna
- Ako je `try` linija horizontalna i prolazi kroz tačku `p`
 - dovoljno je posmatrati algebarska y -rastojanja tačaka `p1` i `p2` u odnosu na `p`
 - ako su istog znaka – `p1` i `p2` su sa iste strane, a u suprotnom su sa različitih
- Funkcija `HSameSide` određuje odnos tačaka `p1` i `p2` prema horizontalnoj liniji (`try`) čija je krajnja leva tačka `p`:

```
Function HSameSide(p,p1,p2: Point): Sign;  
    Begin HSameSide:=Sgn((p1.y-p.y)*(p2.y-p.y))  
End;
```
- Funkcija `HIntersect` određuje da li se proizvoljna linija `l` seče sa horizontalnom linijom čija je krajnja leva tačka `p`

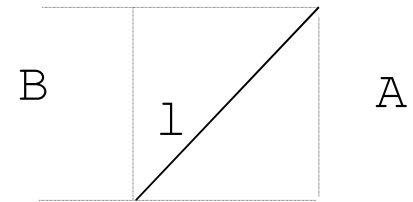
Povećanje efikasnosti (2)

- Posmatraju se sledeće oblasti u kojima može da bude tačka p u odnosu na liniju l :

A: $[x > \max(l.p1.x, l.p2.x)]$ or
 $[y > \max(l.p1.y, l.p2.y)]$ or
 $[y < \min(l.p1.y, l.p2.y)]$

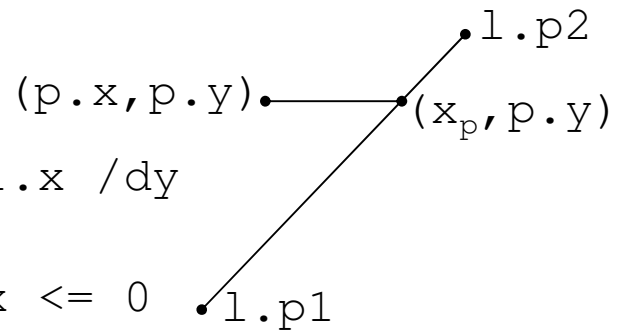
B: $[x < \min(l.p1.x, l.p2.x)]$ and $p \notin A$

- Ako je p u oblasti A
 - ispitna horizontalna linija sigurno ne seče liniju l
- Ako je p u oblasti B
 - ispitna horizontalna linija sigurno seče liniju l
- Ako p nije ni u oblasti A ni u oblasti B
 - mora posebno da se ispita da li postoji presek



Povećanje efikasnosti (3)

- Presek postoji ako je $p.x \leq x_p$
- Iz jednačine prave kroz $l.p1$ i $l.p2$:
$$x_p = (dx/dy)(p.y - l.p1.y) + l.p1.x$$
- Presek postoji ako je:
$$p.x \leq (dx/dy)(p.y - l.p1.y) + l.p1.x / dy$$
- Za $dy > 0$ presek postoji ako je:
$$(p.x - l.p1.x) dy - (p.y - l.p1.y) dx \leq 0$$
- Za $dy < 0$ presek postoji ako je:
$$(p.x - l.p1.x) dy - (p.y - l.p1.y) dx \geq 0$$
- Za svako dy presek postoji ako je:
$$[(p.y - l.p1.y) dx - (p.x - l.p1.x) dy] dy \geq 0$$



Realizacija funkcije HIntersect

```
Function HIntersect(p: Point, l: Line): Boolean;  
  Var dx,dy: LongInt;  
  Begin  
    If p.x>max(l.p1.x,l.p2.x) Or  
       p.y>max(l.p1.y,l.p2.y) Or  
       p.y<min(l.p1.y,l.p2.y) Then { oblast A }  
      Begin HIntersect:=False; return End;  
    If p.x<min(l.p1.x,l.p2.x) Then { oblast B }  
      Begin HIntersect:=True; return End;  
    dx:=l.p2.x-l.p1.x; dy:=l.p2.y-l.p1.y;  
    HIntersect:=((p.y-l.p1.y)*dx-(p.x-l.p1.x)*dy)*dy >=0  
  End;
```