Računarska grafika

Odsecanje

Odsecanje

- Odsecanje ili isecanje (*Clipping*)
  - uklanjanje svih delova primitiva izvan prozora kroz koji se slika posmatra
- Pokrivanje (*Covering*, *Shielding*)
  - uklanjanje delova primitiva unutar prozora
- Predmet lekcije: 2D algoritmi odsecanja
- Algoritmi za odsecanje se dele prema sledećim kriterijumima:
  - prema primitivi koja se odseca
    - tačka
    - linija
    - poligon
  - prema prozoru za odsecanje (*clipping window*)
    - pravougaoni prozor, paralelan osama
    - konveksni poligon
    - proizvoljan poligon

Selektivno postavljanje piksela

- Algoritam rešava problem odsecanja proizvoljne tačke izvan prozora
- Umesto `SetPixel`, primitive se iscrtavaju pozivima `SelSetPixelIn`*`Win`*
- Ako je prozor pravougaoni, ivice paralelne osama, a pripadaju prozoru:

```
Type Window = Record left,right:Xres; bottom,top:Yres End;
Procedure SelSetPixelInRect(w:Window, p:Point, v:Value);
Begin
    If (w.left <= p.x) and (p.x <= w.right) and
       (w.top >= p.y) and (p.y >= w.bottom) Then
       SetPixel(p.x,p.y,v)
End
```

- Ako je prozor proizvoljan poligon:

```
Procedure SelSetPixelInPolygon(py:Polygon, p:Point, v:Value);
Begin If Inside(p,py) Then SetPixel(p.x,p.y,v) End
```

- Metod je neefikasan jer se za svaki piksel vrši ispitivanje

Cohen-Sutherland algoritam

- Algoritam `SelSetPixelInRect` primenjen na liniju je podjednako spor:
  - kada se radi o linijama koje se delimično odbacuju
  - kada se radi o linijama koje se u celini prihvataju ili odbacuju
- Cohen-Sutherland-ov algoritam efikasno rešava problem odsecanja linija izvan pravougaonog prozora
- Osnovna ideja algoritma
  - da se najpre pokuša da se linija u celini prihvati ili odbaci
  - ukoliko se ne uspe – određuje se presek linije i produžene ivice prozora
  - krajnja tačka linije van prozora se premešta u tačku preseka
  - ponovo se pokušava prihvatanje ili odbacivanje preostalog dela linije
  - ponavlja se postupak za svaku od produženih ivica prozora do prihvatanja ili odbacivanja preostalog dela linije

C-S algoritam (1)

1. Ivice prozora se produže tako da se cela slika podeli u 9 oblasti
2. Svakoj oblasti se pridružuje 4-bitni položajni kod (*Outcode*):
   `b3b2b1b0` - gde svaki bit označava jednu oblast:
   - `b3` - iznad
   - `b2` - ispod
   - `b1` - desno
   - `b0` - levo

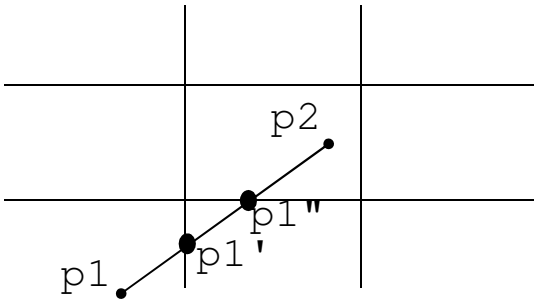
1001	1000	1010
0001	0000	0010
0101	0100	0110

3. Za krajnje tačke linije `p1` i `p2` određuju se položajni kodovi `c1` i `c2`
4. Vrši se ispitivanje:
   a. Ako su i `c1` i `c2` jednaki `0` tada je `(c1|c2)==0` ⇒ linija se trivijalno prihvata
   b. Ako `c1` i `c2` imaju barem 1 zajednički bit `(c1&c2)!=0` ⇒ linija se cela odbacuje

   Napomena: ovim se ne odbacuju sve linije koje su u celini van prozora (npr. `c1==0100`, `c2==0010`)

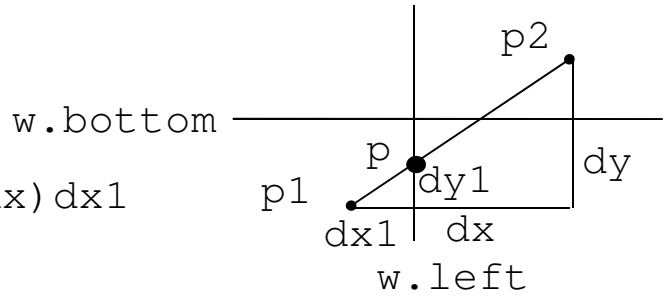
C-S algoritam (2)

5. Za preostale linije, ispituje se da li je tačka koja nije u prozoru (`c!=0`) levo, desno, iznad ili ispod prozora
   - nalazi se presek linije koja se odseca sa odgovarajućom produženom ivicom prozora
6. Krajnja tačka linije se premešta u tačku preseka (`p1`→`p1'`)
7. Ide se na korak 3



- Tačka preseka `P` se određuje na sledeći način (u slučaju da linija seče levu ivicu prozora):

```
p(w.left, p1.y+dy1)
dy=p2.y-p1.y
dx=p2.x-p1.x
dx1=w.left-p1.x
dy1:dx1=dy:dx   => dy1=(dy/dx)dx1
```



- Dobijeni izraz za koordinate tačke `p` ne zavisi od položaja `p1` i `p2` (bitno je samo da linija `p1p2` seče levu ivicu prozora)

C-S procedura (1)

- Definišu se položajni kodovi:

```
Const  LeftCode=1; RightCode=2; BottomCode=4; TopCode=8;
```

- Definiše se tip ivice prozora:

```
Type WindowEdge = (LeftEdge, RightEdge, BottomEdge, TopEdge);
```

- Uvodi se pomoćna funkcija `Outcode`
  - vraća položajni kod tačke `p` u odnosu na prozor `w`:

```
Function Outcode(w: Window; p: Point): Integer;
  Var code: Integer;
  Begin
    code:= 0;
    If (p.x > w.right) Then code:= RightCode
    Else If (p.x < w.left) Then code:= LeftCode;
    If (p.y > w.top) Then code:= code + TopCode {bitski OR}
    Else If (p.y < w.bottom) Then code:= code + BottomCode;
    Outcode:= code;
  End;
```

C-S procedura (2)

- Procedura `WCross` određuje presek `p` linije `l` sa ivicom `e` prozora `w`:

```
Procedure WCross(w:Window; e:WindowEdge; l:Line; Var p:Point);
   Var  dx, dy, dx1, dy1: LongInt;
   Begin
     dx:= l.p2.x - l.p1.x;       dy:= l.p2.y - l.p1.y;
     Case e of
       LeftEdge:   Begin p.x:= w.left;  dx1:= p.x - l.p1.x;
                         p.y:= Round(dy*dx1/dx) + l.p1.y End;
       RightEdge:  Begin p.x:= w.right; dx1:= p.x - l.p1.x;
                         p.y:= Round(dy*dx1/dx) + l.p1.y End;
       BottomEdge: Begin p.y:= w.bottom; dy1:= p.y - l.p1.y;
                         p.x:= Round(dx*dy1/dy) + l.p1.x End;
       TopEdge:    Begin p.y:= w.top; dy1:= p.y - l.p1.y;
                         p.x:= Round(dx*dy1/dy) + l.p1.x End;
     End {Case};
   End;
```

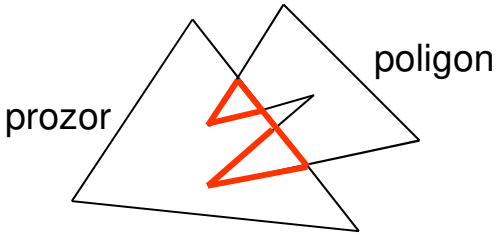
C-S procedura (3)

- Procedura `Clip` odseca delove linije `l` izvan prozora `w` i obaveštava da li se linija (makar i samo njen deo) prihvata (`accept`):

```
Procedure Clip(w: Window; Var l:line; Var accept: Boolean);
    Var  c1,c2,c: Integer;      p: Point;
    Begin
      accept:= True;  c1:= Outcode(w, l.p1);  c2:= Outcode(w, l.p2);
      While ((c1<>0) or (c2<>0)) and accept do
        Begin
          If (c1 and c2) <> 0 Then accept:= False
          Else Begin
            If c1 <> 0 Then c:= c1  Else c:= c2;
            If (c and LeftCode) <> 0 Then WCross(w, LeftEdge, l, p)
            Else If (c and RightCode)<>0 Then WCross(w, RightEdge, l, p)
            Else If (c and TopCode)  <>0 Then WCross(w, TopEdge, l, p)
            Else WCross(w, BottomEdge, l, p);
            If c = c1 Then Begin l.p1:= p; c1:= Outcode(w, l.p1) End
            Else Begin l.p2:= p; c2:= Outcode(w, l.p2) End
          End {Else}
        End {While}
End;
```

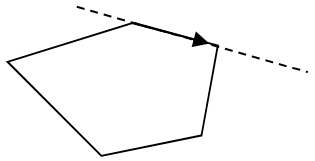
Sutherland-Hodgman-ov algoritam

- Algoritam rešava problem odsecanja proizvoljnog poligona izvan prozora, koji je konveksan poligon
- Strategija "podeli-pa-savladaj":
  - rešavaju se elementarni problemi čija kombinacija rešenja rešava ceo problem
- Elementaran problem:
  - odsecanje poligona u odnosu na jednu ivicu za odsecanje
- Ivica za odsecanje je prava – produžena ivica poligona-prozora
- Sukcesivno odsecanje svim ivicama za odsecanje daje konačni rezultat
- Rezultat predstavljaju delovi poligona koji pripadaju prozoru, zatvoreni ivicama prozora
  - za razliku od rezultata koji bi se dobio primenom Cohen-Sutherland algoritma (ukoliko bi prozor bio pravougaoni)
- Problem: moguće spajanje rezultantnih poligona



S-H algoritam (1)

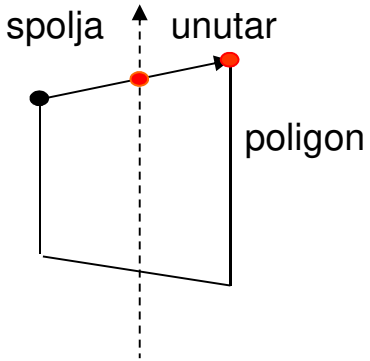
- Temena poligona su $v_1, v_2, \ldots, v_n$;
  ivice poligona su od v_i do v_{i+1}, za $i=1, \ldots, n-1$ i od v_n do v_1
- Ivica prozora je usmerena u smeru kazaljke na časovniku
  - desna strana ivice prozora predstavlja poluravan koja određuje "unutrašnjost"



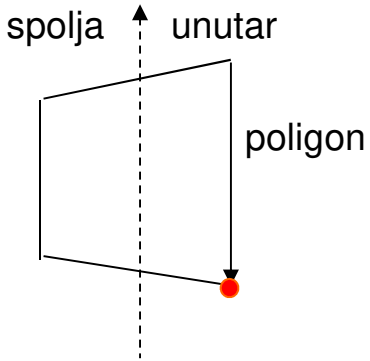
- Odsecanje prema ivici i generiše drugu (izlaznu) seriju temena koja definišu novi poligion
- Kreće se od ivice poligona (v_n, v_1), pa do (v_{n-1}, v_n)
- Ispituje se odnos između susednih temena poligona i ivice odsecanja: 0, 1 ili 2 temena se dodaju u izlaznu listu
- Postoje 4 slučaja koja treba posebno da se analiziraju

S-H algoritam (2)

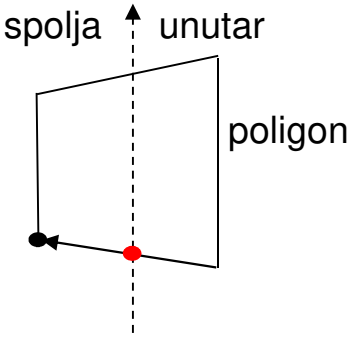
- 4 slučaja:



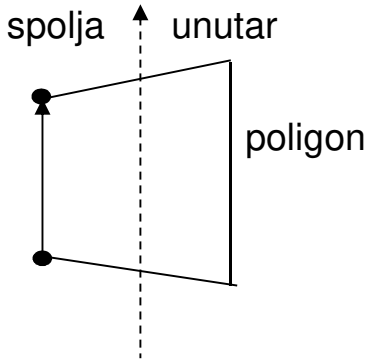
ivica ulazi



ivica unutar

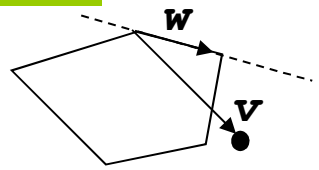


ivica izlazi

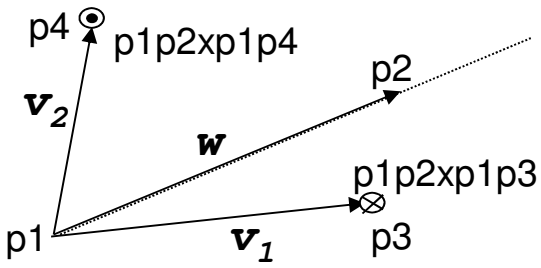


ivica spolja

S-H algoritam (3)



- Prozor je orijentisan u smeru kazaljke časovnika
- Unutrašnjost je definisana kao desna strana ivice odsecanja (orijentisane u smeru kazaljke)
- Test unutrašnjosti se zasniva na vektorskom proizvodu vektora ivice odsecanja $\boldsymbol{w}$ i vektora $\boldsymbol{v}$ od početne tačke ivice odsecanja do ispitivane tačke: $\boldsymbol{w} \times \boldsymbol{v}$
- Ako su oba vektora u XoY ravni desnog koordinatnog sistema sa Y osom naviše:
  - ako je je posmatrana tačka "unutar" (p3) tada je vektor proizvoda $\boldsymbol{w} \times \boldsymbol{v}_1$ u smeru negativne Z ose
  - ako je posmatrana tačka spolja (p4) tada je vektor proizvoda $\boldsymbol{w} \times \boldsymbol{v}_2$ u smeru pozitivne Z ose



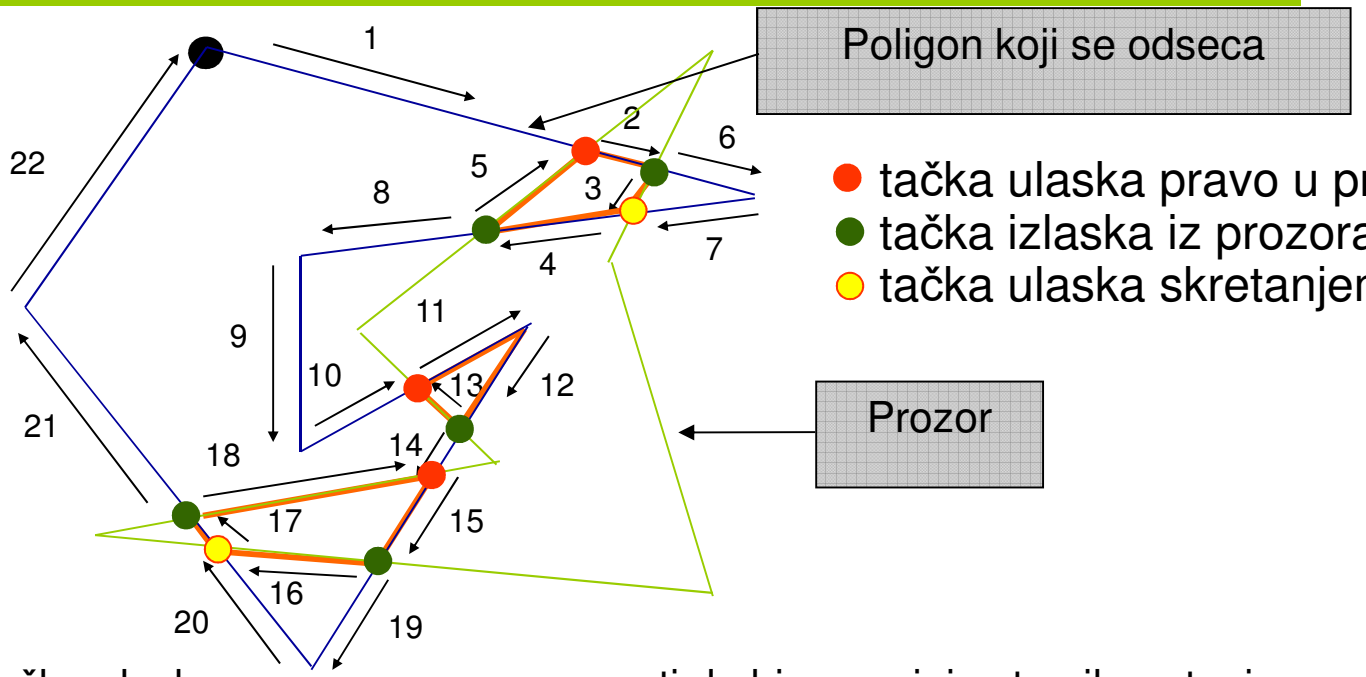
- Tačka je unutar ako je magnituda vektorskog proizvoda negativna
- Vektorski proizvod dva vektora $\boldsymbol{w} \times \boldsymbol{v}$ u XoY ravni ima magnitudu (z komponentu): $w_x v_y - w_y v_x$
- Tačka p je sa unutrašnje strane ivice odsecanja p1p2 ako je: $(p2.x-p1.x)(p.y-p1.y) - (p2.y-p1.y)(p.x-p1.x) < 0$

$$\det \begin{vmatrix} x & y & z \\ w_x & w_y & w_z \\ v_x & v_y & v_z \end{vmatrix}$$

Weiler-Atherton-ov algoritam

- Algoritam rešava problem odsecanja
  - delova proizvoljnog poligona koji su izvan prozora, koji je proizvoljan poligon
- Praktično, određuje se presek dva proizvoljna poligona
- Oba poligona mogu da budu i konkavna
- Algoritam rešava i problem „rupa“ u poligonima
  - poligoni-rupe se orijentišu suprotno od poligona koji se odseca i poligona prozora
- Algoritam
  - kreće se od proizvoljnog temena zadatog poligona koji se odseca,
    u pravcu kazaljke časovnika (temena poligona i prozora su tako uređena)
  - prati se ivica poligona koji se odseca sve do preseka sa ivicom prozora
  - ako ivica "ulazi" u prozor, nastavlja se praćenje ivice poligona koji se odseca
  - ako ivica "izlazi" iz prozora, skreće se "udesno" i nastavlja ivicom prozora,
    na način kao da je on sada poligon za odsecanje,
    a originalni poligon za odsecanje sada prozor (prozor i poligon menjaju ulogu)
  - tačke preseka se pamte da bi se obezbedilo
    da se svi putevi pređu tačno jedan put

W-A algoritam



- Tačka ulaska pravo u prozor se pamti da bi se u njoj zatvorila putanja
- Tačka izlaska iz prozora se pamti da bi se od nje nastavilo istraživanje
- Tačka ulaska skretanjem u prozor se pamti da bi se odustalo od zatvaranja putanje