Računarska grafika

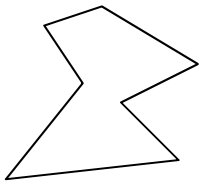
Popunjavanje

Popunjavanje

- Dva načina definisanja površi koja treba da se popuni:
  - vektorsko
    - površ je poligon - poznate su koordinate svakog njegovog temena
    - može da se “izračuna” koji su pikseli unutar poligona
  - rastersko
    - površ je oblast opšteg oblika i postoji jedino u memoriji
    - popunjavanje počinje od zadate tačke u unutrašnjosti oblasti i širi se na okolinu
- Popunjavanje vektorski zadate površi
  - **izračunato popunjavanje** ili **popunjavanje poligona** (*pre-computed filling* ili *poligon filling*)
- Popunjavanje rasterski definisane površi
  - **popunjavanje oblasti** (*region filling*)

Popunjavanje konveksnog poligona

- Ograničenje
  - zahteva se konveksnost poligona barem u pravcu jedne od osa 2D sistema
- Ovde se radi sa konveksnošću u X pravcu
  - slično bi bilo za konveksnost u Y pravcu



- Uvodi se `xval` - vektor čiji je indeks y-koordinata,
  a vrednosti elemenata su odgovarajuće vrednosti x-koordinate
- Algoritam:
  1. Postaviti sve elemente `xval` na `(-1)`
  2. Generisati poliliniju
     pri tome za piksel sa koordinatama `(x,y)` proveravati `xval[y]`:
     a) ako je `xval[y]<0` $\Rightarrow$ `xval[y]:=x`
     b) ako je `xval[y]>=0` $\Rightarrow$ popuniti sve piksele između `(x,y)` i `(xval[y],y)`

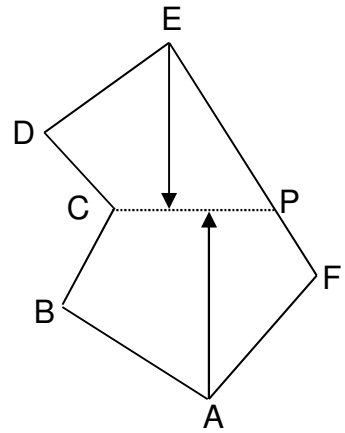
Implementacija

- Realizacija:
  1. pre crtanja potrebno je inicijalizovati `xval` pozivom `SetXVal`
  2. kada se zahteva crtanje popunjenog poligona, iz procedure za crtanje linijskog segmenta polilinije (npr. `BresenhamLineDraw`) poziva se procedura `PixelFill` umesto `SetPixel`

```
Var xval: Array[Yres] of Integer; {-1,0..MaxX}
Procedure SetXVal;
  Var y: Yres;
  Begin For y:=0 To MaxY Do xval[y]:=-1 End;
Procedure PixelFill(x:Xres; y:Yres; v: Value);
  Var x0,x1,i: Xres;
  Begin
    If xval[y]>=0 Then Begin
      x0:=min(x,xval[y]); x1:=max(x,xval[y]);
      For i:=x0 To x1 Do SetPixel(i,y,v)
    End;
    xval[y]:=x
  End;
```

Efekat popunjavanja

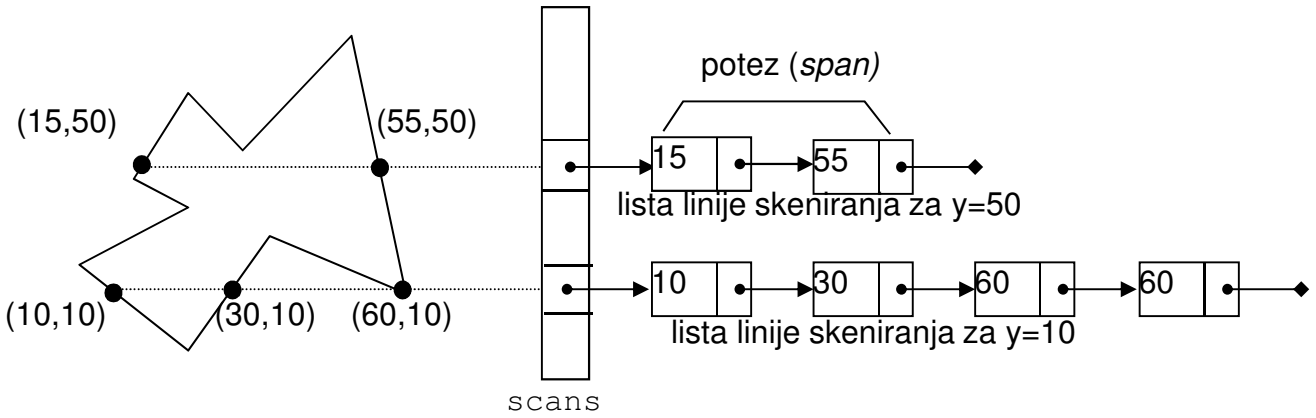
- Za poligon sa slike,
  ukoliko se crtanje počne
  od temena C u smeru (C→D)
  - popunjavanje počinje od E prema F i traje do tačke P;
  - od P do A nema popunjavanja;
  - popunjavanje ponovo počinje od tačke A do tačke C



- Napomena:
  - kada bi procedura za crtanje linije uvek crtala liniju PQ u smeru (P→Q), efekat bi bio upravo kako je i opisano
  - to najčešće nije slučaj, pa je i efekat nešto drugačiji

Popunjavanje proizvoljnog poligona

- Pretpostavka:
  - ivica (granica) i unutrašnjost se popunjavaju istom vrednošću

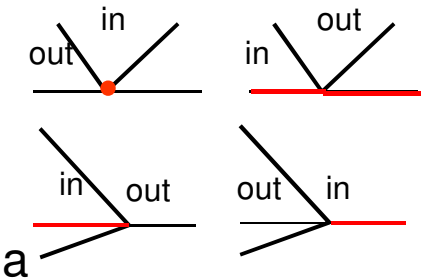


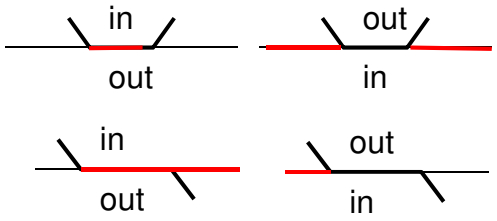
Algoritam

- Svaka y-vrednost definiše jednu horizontalnu liniju skeniranja (*scan line*)
- Svakoj liniji skeniranja odgovara jedan pokazivač u vektoru `scans`
- Ivice poligona se "obilaze" redom
  - modifikovanim Bresenham-ovim algoritmom za crtanje linije
- Izračunat `(x,y)` par koji leži na ivici (liniji) daje jedan element u listi na koju pokazuje `scans[y]`
- Liste `x` vrednosti se formiraju neopadajuće uređeno
- Sukcesivni elementi, i to neparan-paran, formiraju poteze - horizontalne linijske segmente (*span*)
- Potezi pripadaju unutrašnjosti oblasti i oni se prikazuju

Problemi

- Postoje sledeći problemi sa linijama skeniranja
- Linija skeniranja prolazi kroz teme poligona
  - ako su oba susedna temena sa iste strane:
    sve je u redu – kritično teme ulazi dva puta u listu
  - ako su susedna temena sa različitih strana:
    korekcija – kritično teme ulazi samo jedanput u listu
- Linija skeniranja prolazi kroz horizontalnu ivicu poligona
  - sama horizontalna ivica se preskače,
    ali se naknadno crta linija koja joj odgovara
  - ako su oba susedna temena sa iste strane:
    sve je u redu – oba kritična temena ulaze u listu
  - ako su susedna temena sa različitih strana:
    korekcija – samo jedno kritično teme ulazi u listu





Popunjavanje oblasti

- Oblast je grupa susednih, povezanih piksela
- Za oblast je karakteristično
  - definisana je rasterski - vrednošću piksela u (video) memoriji
  - nije definisana vektorski - poligonom koji je ograničava
- Popunjavanje oblasti je moguće samo na uređajima koji poseduju memoriju u kojoj je smeštena slika
- Potrebna je funkcija za očitavanje vrednosti proizvoljnog piksela:

```
Function Pixel(x:Xres; y:Yres): Value;
```

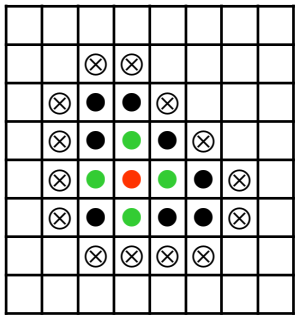
- Uobičajeno, oblast se definiše na jedan od dva načina:
  - svi pikseli koji pripadaju oblasti imaju datu vrednost
  - pikseli koji su na ivici oblasti imaju datu vrednost

Tipovi oblasti (1)

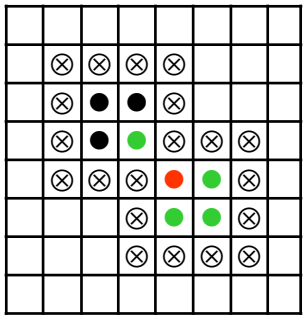
- Klasifikacija oblasti prema mestu piksela koji je defnišu:
- Oblast definisana unutrašnošću (*interior-defined*)
  - svi pikseli u oblasti imaju vrednost `old_value`,
    i nema piksela na granici oblasti sa tom vrednošću
  - algoritmi "poplavljivanja" (*flood fill*) rade nad ovakvim oblastima
    da postave njihove piksele na vrednost `new_value`
- Oblast definisana granicom (*boundary-defined*)
  - pikseli na granici oblasti imaju vrednost `boundary_value`,
    dok pikseli u oblasti imaju proizvoljnu drugu vrednost
  - algoritmi "oivičenog popunjavanja" (*boundary fill*) rade nad ovakvim oblastima
    da postave sve njihove piksele na vrednost `new_value`

Tipovi oblasti (2)

- Klasifikacija oblasti prema povezanosti piksela
- Četvoro-susedna oblast (*4-connected*)
  - od proizvoljnog piksela u oblasti do svih drugih u oblasti može da se stigne sekvencom horizontalnih i vertikalnih pomeraja za 1 piksel
- Osmo-susedna oblast (*8-connected*)
  - od proizvoljnog piksela u oblasti do svih drugih u oblasti može da se stigne sekvencom horizontalnih, vertikalnih i dijagonalnih pomeraja za 1 piksel



Četvorosusedna oblast ●
Osmosusedna granica ⊗



Osmosusedna oblast ●
Četvorosusedna granica ⊗

- Granica 4-susednih oblasti je 8-susedna, a 8-susednih oblasti 4-susedna

Popunjavanje poplavljivanjem

- Jednostavni rekurzivni algoritam za
  - 8-susednu oblast
  - definisanu unutrašnjošću (`old_val`)
  - koji postavlja unutrašnjost na `new_val`

```
Procedure FloodFill8(x:Xres; y:Yres; new_val,old_val:Value);
  Begin
    If  Pixel(x,y) = old_val Then
      Begin
        SetPixel(x,y,new_val);
        FloodFill8(x-1,y,new_val,old_val);    FloodFill8(x+1,y,new_val,old_val);
        FloodFill8(x,y-1,new_val,old_val);    FloodFill8(x,y+1,new_val,old_val);
        FloodFill8(x-1,y-1,new_val,old_val);  FloodFill8(x-1,y+1,new_val,old_val);
        FloodFill8(x+1,y-1,new_val,old_val);  FloodFill8(x+1,y+1,new_val,old_val)
     End
  End;
```

Popunjavanje oivičene oblasti

- Jednostavan rekurzivni algoritam za
  - 4-susednu oblast
  - definisanu granicom (`boundary_val`)
  - koji postavlja celu unutrašnjost na `new_val`

```
Procedure BoundaryFill4(x:Xres;y:Yres;new_val,boundary_val:Value);
  Begin
    If (Pixel(x,y)<>boundary_val) and (Pixel(x,y)<>new_val) Then
      Begin
        SetPixel(x,y,new_val);
        BoundaryFill4(x-1,y,new_val,boundary_val);
        BoundaryFill4(x+1,y,new_val,boundary_val);
        BoundaryFill4(x,y-1,new_val,boundary_val);
        BoundaryFill4(x,y+1,new_val,boundary_val)
     End
  End;
```

Iterativni algoritam (1)

- Uvodi se kružni bafer `queue` veličine `Qmax`
  - indeksi `qfirst` (ukazuje na prvu tačku u baferu) i `qlast` (ukazuje na poslednju tačku u baferu)
  - procedure za pristup baferu `Insert` (stavljanje tačke) i `Remove` (uzimanje tačke)
- Za 4-susednu oblast definisanu unutrašnjom vrednošću `old_val`:

```
Procedure ForestFireFill4(x:Xres;y:Yres;new_val,old_val:Value);
    Const  Qmax  = ...;
    Type   Qindex = 0..Qmax-1;
    Var    queue: Array[Qindex] of Point;   qfirst,qlast: Qindex;
    Procedure Insert(x:Xres; y:Yres);
      Begin
        qlast:= (qlast + 1) mod Qmax;
        queue[qlast].x:= x;  queue[qlast].y:= y
      End;
    Procedure Remove(Var x:Xres; Var y:Yres);
      Begin
        x:= queue[qfirst].x; y:= queue[qfirst].y;
        qfirst:= (qfirst + 1) mod Qmax
      End;
```

Iterativni algoritam (2)

```
Procedure SetPixQ(x: Xres; y: Yres);
  Begin
    If Pixel(x,y)=old_val Then
      Begin SetPixel(x,y,new_val); Insert(x,y); End
  End;
 Begin
   qfirst:= 1; qlast:= 0;   SetPixQ(x,y);
   Repeat
     Remove(x,y);
     SetPixQ(x-1,y); SetPixQ(x+1,y);
     SetPixQ(x,y-1); SetPixQ(x,y+1)
   Until qfirst=(qlast+1) mod Qmax {dok qfirst ne prestigne qlast}
 End;
```

- U datoj implementaciji nije rešen problem prepunjavanja bafera
- Procedura ima efekat "šumskog požara" (*forest fire*)
- Manja potrošnja memorije i brži od rekurzivnog algoritma
- Može dodatno da se ubrza

Ubrzan iterativni algoritam

- Algoritam se zasniva na popunjavanju horizontalnih redova piksela
- Crta se horizontalna linija kroz datu tačku sve do granica u oba pravca
- Krajnje tačke linije se stavljaju u red čekanja
  - ne stavlja se u red svaka popunjena tačka
- Posledice: smanjuje se red čekanja i procedura se ubrzava
- Problem sa otkrivanjem “džepova” u oblasti