Računarska grafika

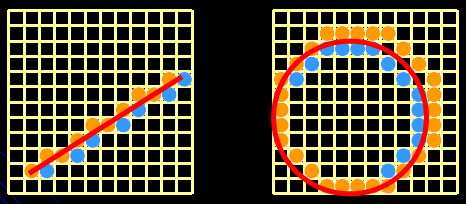
predavanja
v.prof.dr. Samir Lemeš
slemes@unze.ba

28. Rasterizacija

- Rasterizacija linija
- DDA algoritam
- Bresenhamov algoritam
- Rasterizacija kruga
- Rasterizacija elipse
- Rasterizacija trougla

Rasterizacija

- Pretvaranje iz kontinuiranog u diskretno

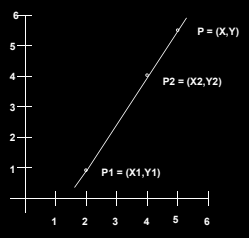


Malo matematike

Data je treća tačka na liniji:
P = (X,Y)

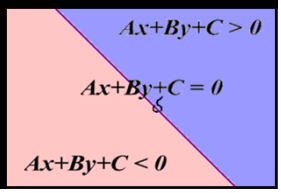
Nagib = $(Y - Y_1)/(X - X_1)$
= $(Y_2 - Y_1)/(X_2 - X_1)$

Rješenje po Y
$Y = [(Y_2-Y_1)/(X_2-X_1)]X + [-(Y_2-Y_1)/(X_2-X_1)]X_1 + Y_1$
ili
Y = mx + b



Jednačine linije

- Pitanje: *Koji je implicitni oblik jednačine linije?*
  - $Ax + By + C = 0$
- Pitanje: *Ako se znaju koordinate tačke (x,y), šta se dobije uvrštavanjem tih vrijednosti u jednačinu linije?*
  - Da li je tačka:
    - Na liniji: $Ax + By + C = 0$
    - "Iznad" linije: $Ax + By + C > 0$
    - "Ispod" linije: $Ax + By + C < 0$

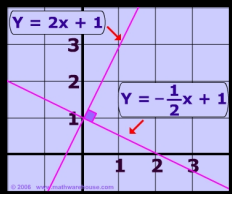


Druge korisne formule

Dužina segmenta linije između P_1 i P_2:
$L = [(X_2-X_1)^2 + (Y_2-Y_1)^2]^{1/2}$

Srednja tačka segmenta linije između P_1 i P_3:
$P_2 = [(X_1+X_3)/2 , (Y_1+Y_3)/2]$

Dvije linije su okomite ako je
1) $M_1 = -1/M_2$
2) Kosinus ugla između linija = 0.



Parametarski oblik jednačine 2D Linije

Date su tačke $P_1 = (X_1, Y_1)$ i $P_2 = (X_2, Y_2)$

$$X = X_1 + t(X_2 - X_1) \qquad Y = Y_1 + t(Y_2 - Y_1)$$

t se naziva "parametar". Kad je

t = 0 dobije se (X_1, Y_1)

t = 1 dobije se (X_2, Y_2)

Kako je $0 < t < 1$ dobiju se sve ostale tačke na segmentu linije između (X_1, Y_1) i (X_2, Y_2).

Osnovni algoritmi za liniju i krug

1. Moraju se izračunati cjelobrojne koordinate piksela koji leže na ili u blizini linije ili kruga.
2. Algoritmi za vrednovanje piksela se pozivaju stotinama ili hiljadama puta svaki put kad se slika kreira ili promijeni.
3. Linije moraju formirati vizualno prihvatljive slike.
   - Linije moraju izgledati prave
   - Linije moraju imati precizno definisane krajeve
   - Linije moraju imati konstantnu debljinu
   - Debljina linije ne smije zavisiti od dužine i nagiba linije.
4. Algoritmi za linije moraju uvijek biti definisani.

Jednostavni DDA algoritam za linije

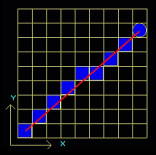
{Zasnovan na parametarskoj jednačini linije}

```
Procedure DDA(X1,Y1,X2,Y2 :Integer);
Var         Length, I  :Integer;
    X,Y,Xinc,Yinc    :Real;

Begin
    Length := ABS(X2 - X1);
    If ABS(Y2 - Y1)  > Length Then
           Length := ABS(Y2-Y1);
    Xinc := (X2 - X1)/Length;
    Yinc := (Y2 - Y1)/Length;
    X := X1;
    Y := Y1;
    For I := 0 To Length Do
    Begin
          Plot(Round(X), Round(Y));
          X := X + Xinc;
          Y := Y + Yinc
    End  {For}
End;  {DDA}
```

Digital Differential Analyzer (Digitalni diferencijalni analizator)

DDA kreira dobre linije ali je prespor zbog funkcije "round" i sporih operacija nad realnim brojevima.



DDA primjer

Izračunati koji pikseli trebaju biti uključeni da prikažu liniju od (6,9) do (11,12).

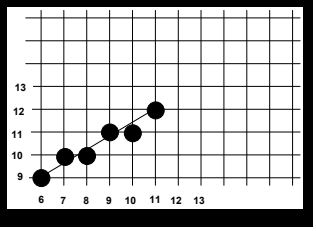
Dužina := Max od (ABS(11-6), ABS(12-9)) = 5

Xinc := 1

Yinc := 0.6

Izračunate vrijednosti su:

(6,9), (7,9.6), (8,10.2), (9,10.8), (10,11.4), (11,12)



Jednostavni algoritmi za kružnicu

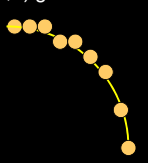
Jednačina kružnice radijusa r sa centrom u (0,0) glasi:

$$x^2 + y^2 = r^2,$$

očigledno treba nacrtati:

$$y = \pm (r^2 - x^2)^{1/2}$$

za $-r <= x <= r$.



To funkcioniše, ali je neefikasno zbog množenja i kvadratnog korijena. Također kreira velike greške u kružnici za vrijednosti x koje su blizu R.

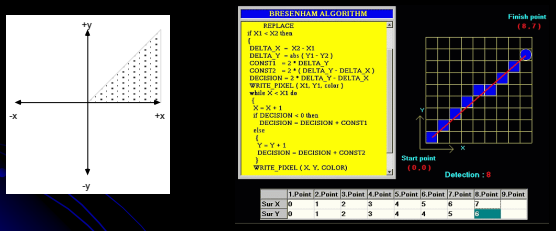
Bolji pristup, koji je još uvijek neefikasan, ali izbjegava greške je crtanje:

$$x = r \cos\varphi$$
$$y = r \sin\varphi$$

tako da φ uzima vrijednosti od 0 do 360 stepeni.

Bresenhamov algoritam

Pretpostavka: crtanje linije nagiba m od 0 do 1

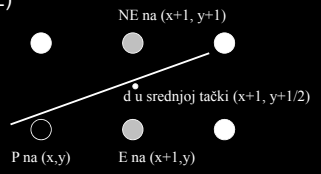


Koristi se implicitna jednačina linije: **y = mx + B**

gdje je **m** nagib linije a **B** je presjek sa y.

Brze linije

Sljedeći piksel je desno (E) ili desno gore (NE)



Ako je d pozitivno linija siječe iznad srednje tačke i bliža je tački T.

Ako je d negativno, linija siječe ispod srednje tačke i bliža je tački S.

Da bi se izabrala prava tačka treba samo znati predznak tačke d.

Brze linije – varijabla odluke

$d_i = f(x_i+1, y_i+ 1/2) = a(x_i+1) + b(y_i+ 1/2) + c$
$= ax_i + by_i + c + a + b/2$
$= f(x_i, y_i) + a + b/2$

d_i je poznata kao **varijabla odluke**.

Algoritam:
Ako je $d_i \geq 0$ izaberi NE = $(x_i + 1, y_i + 1)$ kao sljedeću tačku

$d_{i+1} = f(x_{i+1} + 1, y_{i+1} + 1/2) = f(x_i +1+1, y_i +1+1/2)$
$= a(x_i +1+1) + b(y_i +1+1/2) + c$
$= f(x_i + 1, y_i + 1/2) + c + a + b$ (vidi prvi red iznad)
$= d_i + a + b$

A ako nije, izaberi E = $(x_i + 1, y_i)$ kao sljdeću tačku

$d_{i+1} = f(x_{i+1} + 1, y_{i+1} + 1/2) = f(x_i +1+1, y_i +1/2)$
$= a(x_i +1+1) + b(y_i +1/2) + c = f(x_i + 1, y_i + 1/2) + a$
$= d_i + a$

Bresenhamov algoritam za liniju

Samo vrijednost koja nije cjelobrojna je b/2 u početnoj varijabli odluke. Može se pomnožiti sa 2 da bi se izbjeglo dijeljenje.

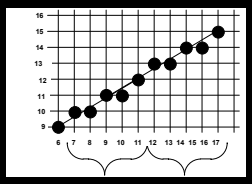
```
Begin  {Bresenham za linije s nagibom od 0 do 1}
    a := ABS(xend - xstart);
    b := ABS(yend - ystart);
    d := 2*a + b;
    If xstart > xend Then Begin
        x := xend;
        y := yend
    End
    Else Begin
        x := xstart;
        y := ystart
    End;
```

```
For I := 0 to a Do Begin
        Plot(x,y);
        x := x + 1;
        If d ≥ 0 Then
        Begin
            y := y + 1;
            d := d + a + b
        End
        Else d := d + a
    End  {For Loop}
End;  {Bresenham}
```

Optimizacije

Brzina se može još više povećati otkrivanjem ciklusa kod varijable odluke. Ti ciklusi odgovaraju ponavljajućem nizu izbora piksela.

Ponavljajući niz se snimi i ako se otkrije ciklus, ponavlja se bez ponovnog proračuna.

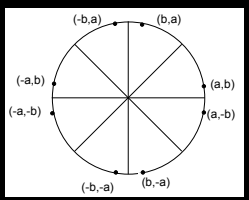


Algoritam za crtanje kružnice

Potrebno je samo izračunati vrijednosti na rubu kruga u prvom oktantu. Ostale vrijednosti se mogu dobiti simetrijom.

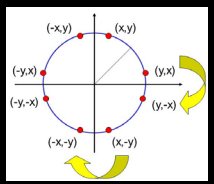
Polazi se od kružnice radijusa r s centrom u (0,0).

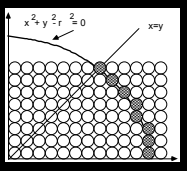
```
Procedure Circle_Points(x,y :Integer);
Begin
    Plot(x,y);
    Plot(y,x);
    Plot(y,-x);
    Plot(x,-y);
    Plot(-x,-y);
    Plot(-y,-x);
    Plot(-y,x);
    Plot(-x,y)
End;
```



Brze kružnice

Uzme se u obzir samo prvi oktant kružnice radijusa r sa centrom u koordinatnom početku. Počinje se crtanjem tačke (r,0) a završava kad bude x < y.



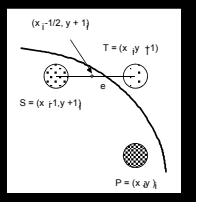


Odluka u svakom koraku je da li odabrati piksel direktno iznad posmatranog piksela ili piksel koji je iznad i ulijevo (8-smjerna simetrija).

Pretpostavke:
$P_i = (x_i, y_i)$ je piksel koji se posmatra.
$T_i = (x_i, y_i +1)$ je piksel direktno iznad njega
$S_i = (x_i -1, y_i +1)$ je piksel iznad i ulijevo od njega.

Brze kružnice – varijable odluke

$f(x,y) = x^2 + y^2 - r^2 = 0$



$f(x_i - 1/2 + e, y_i + 1)$
$= (x_i - 1/2 + e)^2 + (y_i + 1)^2 - r^2$
$= (x_i - 1/2)^2 + (y_i+1)^2 - r^2 + 2(x_i-1/2)e + e^2$
$= f(x_i - 1/2, y_i + 1) + 2(x_i - 1/2)e + e^2 = 0$
Let $d_i = f(x_i - 1/2, y_i+1) = -2(x_i - 1/2)e - e^2$

Ako je $e < 0$ onda je $d_i > 0$ pa se bira tačka $S = (x_i - 1, y_i + 1)$.
$d_{i+1} = f(x_i - 1 - 1/2, y_i + 1 + 1) = ((x_i - 1/2) - 1)^2 + ((y_i + 1) + 1)^2 - r^2$
$= d_i - 2(x_i - 1) + 2(y_i + 1) + 1 = d_i + 2(y_{i+1} - x_{i+1}) + 1$

Ako je $e \geq 0$ onda je $d_i \leq 0$ pa se bira tačka $T = (x_i, y_i + 1)$.
$d_{i+1} = f(x_i - 1/2, y_i + 1 + 1) = d_i + 2y_{i+1} + 1$

Brze kružnice – varijable odluke

Početna vrijednost za d_i je
$d_0 = f(r - 1/2, 0 + 1) = (r - 1/2)^2 + 1^2 - r^2$
$= 5/4 - r$ {1-r se može koristiti ako je r cijeli broj}

Kad se izabere tačka $S = (x_i - 1, y_i + 1)$ onda je
$d_{i+1} = d_i + -2x_{i+1} + 2y_{i+1} + 1$

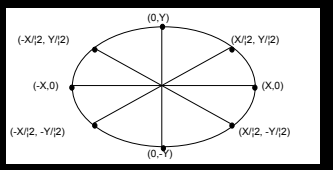
Kad se izabere tačka $T = (x_i, y_i + 1)$ onda je
$d_{i+1} = d_i + 2y_{i+1} + 1$

Algoritam za brzu kružnicu

```
Begin  {Circle}
    x := r;
    y := 0;
    d := 1 - r;
    Repeat
        Circle_Points(x,y);
        y := y + 1;
        If d < 0 Then
            d := d + 2*y + 1
        Else Begin
            x := x - 1;
            d := d + 2*(y-x) + 1
        End
    Until x < y
End;  {Circle}
```

Brze elipse

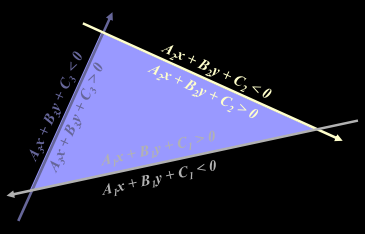
Algoritam za kružnicu se može generalizovati da bi radio i sa elipsom ali se može koristiti samo 4-smjerna simetrija.



Moraju se izračunati sve tačke u jednom kvadrantu.
Kako je Bresenhamov algoritam ograničen samo na jedan oktant, proračun se mora vršiti u dva koraka.
Promjena se dešava kad se postigne tačka na elipsi u kojoj tangenta na elipsu u njoj ima nagib od ±1.
U prvom kvadrantu, to se dešava kad obje koordinate x i y imaju vrijednost 0.707 svog maksimuma.

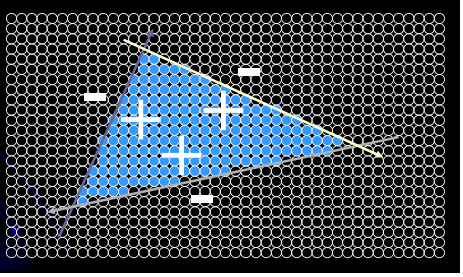
Rasterizacija trougla

- Trougao se može definisati kao presjek tri pozitivne poluravni:



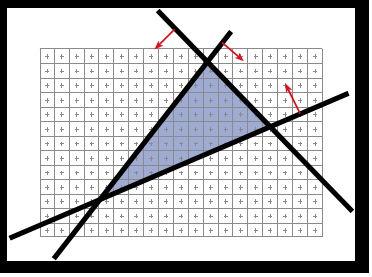
Rasterizacija trougla

- Uključeni su samo oni pikseli kod kojih su sve jednačine stranica trougla > 0:



Rasterizacija trougla

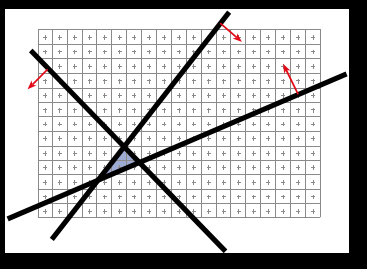
- Za svaki piksel je potrebno
  - Izračunati jednačine linija u centru piksela
  - "odsjeći" dio površine oko trougla



Rasterizacija trougla

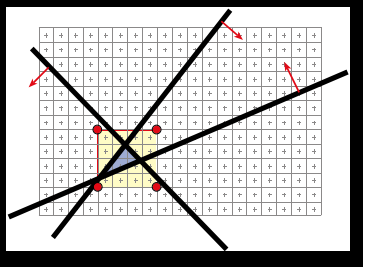
- Za svaki piksel je potrebno
  - Izračunati jednačine linija u centru piksela
  - "odsjeći" dio površine oko trougla

Problem?
Ako je trougao mali, ima previše beskorisnog proračuna



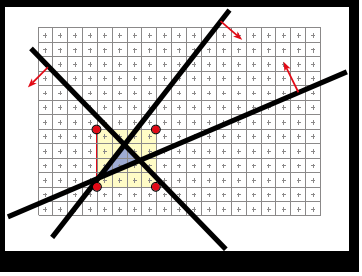
Rasterizacija trougla

- Unapređenje: Računati samo piksele trougla koji su unutar *gabarita ekrana*
- Kako se odredi taj gabarit?
  - Xmin, Xmax, Ymin, Ymax vrhova trougla



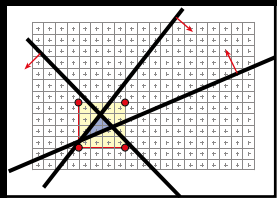
Moderne grafičke kartice

- Trouglovi su obično jako mali
- Problematično podešavanje
- Odsijecanje je naporno



Moderna rasterizacija

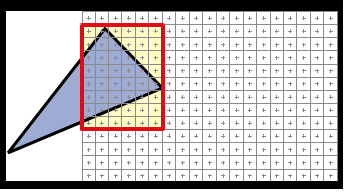
```
Za svaki trougao
    IzračunajProjekcije
    Izračunaj gabarit, odsijeci gabarit do granica ekrana
    Za sve piksele unutar gabarita
        Izračunaj jednačine linija
        Ako su sve jednačine linija>0 //piksel [x,y] unutar trougla
            Framebuffer[x,y]=BojaTrougla
```



Moderna rasterizacija

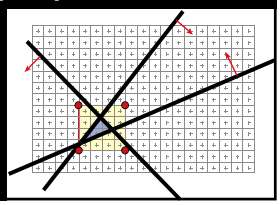
```
Za svaki trougao
    IzračunajProjekcije
    Izračunaj gabarit, odsijeci gabarit do granica ekrana
    Za sve piksele unutar gabarita
        Izračunaj jednačine linija
        Ako su sve jednačine linija>0 //piksel [x,y] je unutar trougla
            Framebuffer[x,y]=BojaTrougla
```

- Odsijecanje gabarita je trivijalno, za razliku od odsijecanja trougla



Može li bolje?

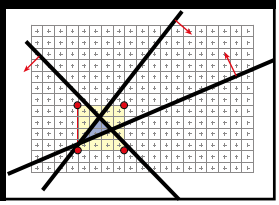
```
Za svaki trougao
    IzračunajProjekcije
    Izračunaj gabarit, odsijeci gabarit do granica ekrana
    Za sve piksele unutar gabarita
        Izračunaj jednačine linija
        Ako su sve jednačine linija>0 //piksel [x,y] unutar trougla
              Framebuffer[x,y]=BojaTrougla
```



Može li bolje?

```
Za svaki trougao
    IzračunajProjekcije
    Izračunaj gabarit, odsijeci gabarit do granica ekrana
    Za sve piksele unutar gabarita
        Izračunaj jednačine linija ax+by+c
        Ako su sve jednačine linija>0 //piksel [x,y] je unutar trougla
              Framebuffer[x,y]=BojaTrougla
```

- Ne mora se svaki put ponovo izračunavati jednačina linije ispočetka



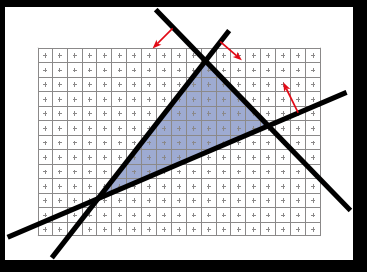
Može li bolje?

```
Za svaki trougao
  IzračunajProjekcije
  Izračunaj gabarit, odsijeci gabarit do granica ekrana
  Podesi jednačinu linije
      izračunaj aidx, bidy za 3 linije
      Daj početne vrijednosti jednačina linija,
        vrijednosti za vrhove gabarita
          Li=aix0+biy+ci
    Za svaku liniju skeniranja y unutar gabarita
      Za 3 linije, ažuriraj Li
      Za sve x unutar gabarita
          Inkrementiraj jednačine linija: Li+=adx
      Ako su sve Li>0 //piksel [x,y] unutar trougla
            Framebuffer[x,y]=BojaTrougla
```

- Ušteda: po jedno množenje za svaki piksel

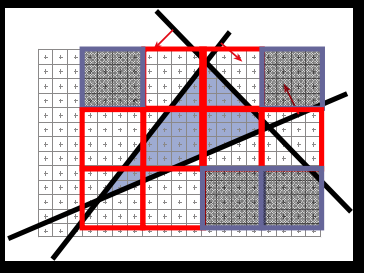
Može li bolje?

- Izračunavaju se jednačine linija za veliki broj piksela koji se ne koriste
- Šta se može uraditi?



Može li bolje?

- Hijerarhijska rasterizacija
  - Obično u dva nivoa
  - Tada je samo pitanje određivanja prave granulacije



U modernom hardveru

- Jednačine stranica trougla u homogenim koordinatama [x, y, w]
- Podjela da bi se dodala granulacija srednjeg nivoa
  - Rano se odbacuju beskorisna područja
  - Koherentan pristup memoriji