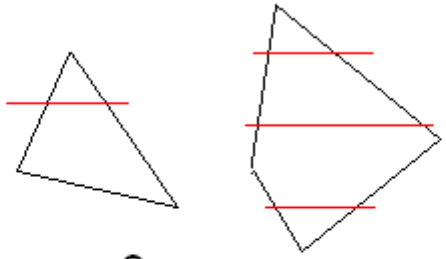
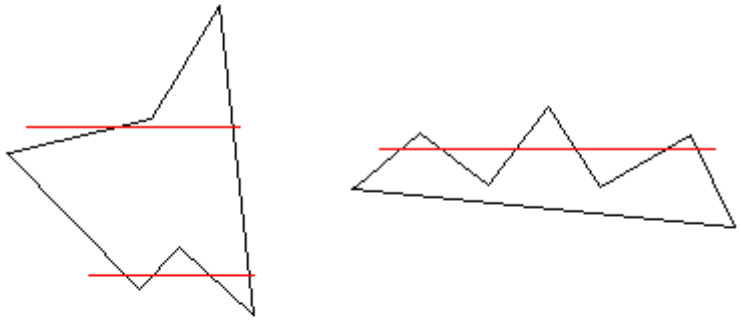
Rasterizarea suprafetelor poligonale

Prof. univ. dr. ing. Florica Moldoveanu

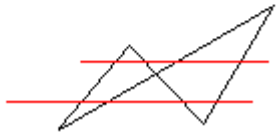
Rasterizarea suprafetelor poligonale (1)



Convexe



Concave



Autointersectant

Forma generala a algoritmului

pentru toate liniile raster care intersecteaza suprafata poligonului

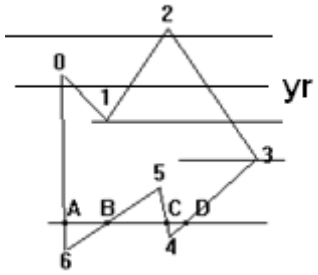
- calculeaza punctele de intersectie dintre linia raster si laturile poligonului
- coloreaza pixelii interiori poligonului, de pe fiecare linie raster

Probleme

1) O linie raster nu intersecteaza toate laturile poligonului.

2) Intersectia dintre o linie raster si un poligon concav produce mai multe segmente de pixeli interiori.

Rasterizarea suprafetelor poligonale (2)



1) O linie raster nu intersecteaza toate laturile poligonului.

➢ se determina pentru fiecare linie raster setul laturilor intersectate:

setul laturilor active (ymin_latura <= yr <ymax_latura)

2) Intersectia dintre o linie raster si un poligon concav produce mai multe segmente de pixeli interiori.

➢Ordonează punctele de intersecție crescător după abscisă: x1, x2, x3, x4 . Exemplu: A, B, C, D.

➢Colorează pixelii de pe segmentele de dreaptă interioare suprafetei poligonului:

(x1, yr) – (x2,yr); (x3-y3) – (x4,yr). Exemplu: (A-B, C-D).

❖Punctele de pe segmentele (x1, yr) – (x2,yr); (x3-y3) – (x4,yr), ..,respecta **definitia punctului interior unui poligon**: "***un punct este interior unui poligon dacă orice semidreaptă dusă din punctul respectiv intersectează poligonul într-un număr impar de puncte*** "

Rasterizarea suprafetelor poligonale (3)

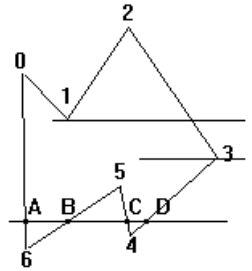
❖ **Cazuri particulare: liniile raster care traverseaza varfurile poligonului:**

- daca fiecare varf apartine la 2 laturi, atunci, o semidreapta orizontala dusa dintr-un punct interior poligonului

  -va produce 3 puncte de intersectie daca traverseaza varful 1

  -va produce 2 puncte de intersectie daca traverseaza varful 3

- Varfurile 0, 1, 2, 4, 5, 6 sunt varfuri de minim / maxim local



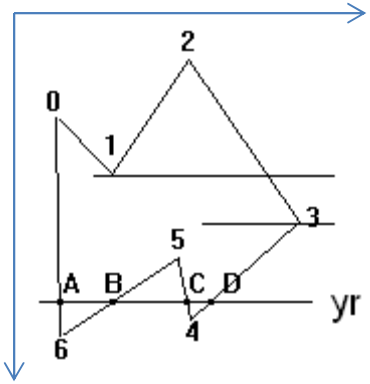
Conventie:

- la traversarea unui varf de minim / maxim local trebuie sa rezulte 0 sau 2 puncte de intersectie
- la traversarea oricarui alt varf trebuie sa rezulte un singur punct de intersectie

Implementarea conventiei: se considera fiecare latura un interval inchis la un capat si deschis la celalalt, de exemplu: [ymin_latura – ymax_latura); Atunci:

-un vârf de maxim local nu aparține nici uneia dintre laturile care-l formează iar un vârf de minim local aparține ambelor laturi.

Rasterizarea suprafetelor poligonale (4)



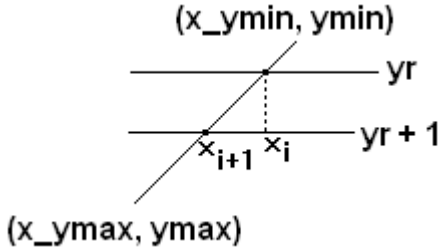
Algoritmul de rasterizare

1) Se crează lista laturilor poligonului, ordonată crescător după ordonatele minime ale laturilor: 1-2, 2-3, 0-1, 0-6, 3-4, 4-5, 5-6

2) for(yr = ymin_poligon; yr< ymax_poligon; yr++)

- Determina setul laturilor active
- Calculează intersecțiile dreptei y=yr cu laturile active. De exemplu: A, D, C, B.
- Ordonează punctele de intersecție crescător după abscisă: A, B, C, D.
- Colorează pixelii de pe segmentele de dreaptă interioare suprafetei poligonului (A-B, C-D).

Rasterizarea suprafetelor poligonale (5)



Calculul intersectiilor liniilor raster cu laturile poligonului

$m = (ymax – ymin)/(x\_ymax – x\_ymin) = (yr+1 – yr)/(x_{i+1} – x_i)$

$x_{i+1} = x_i + 1/m$ sau $x_{i+1} = x_i + dx$

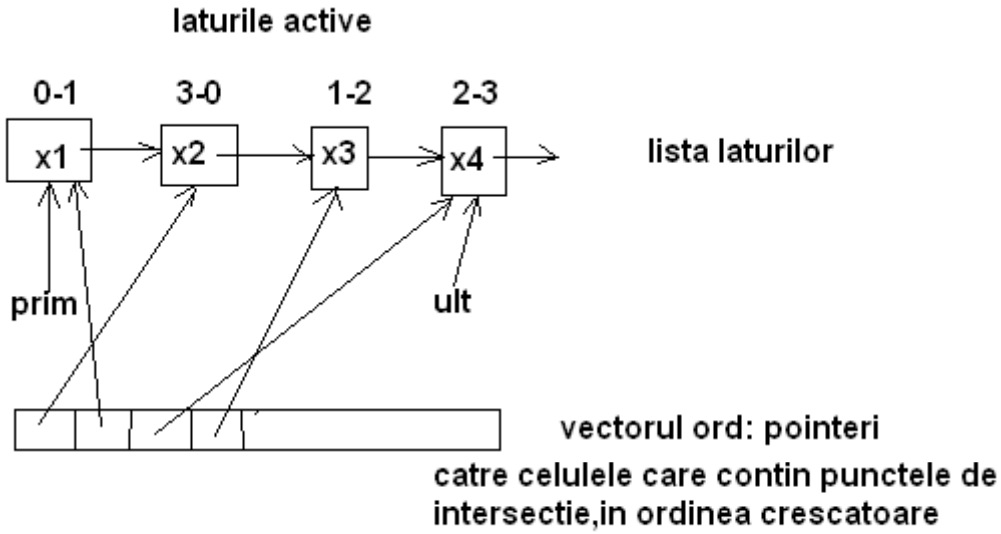
x1 = x_ymin – dx (inainte ca latura sa devina activa)

O implementare(1)

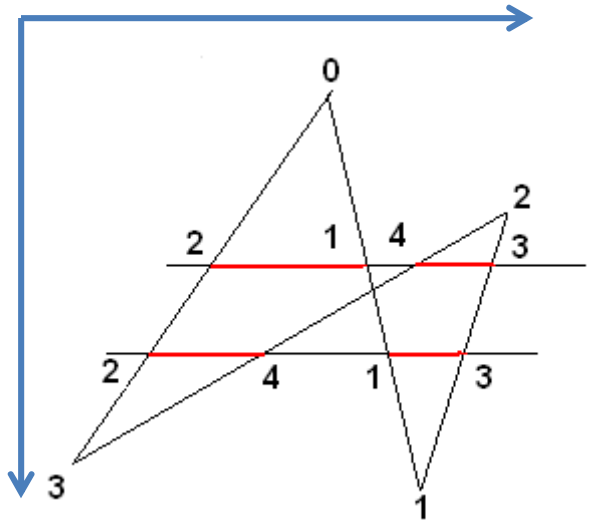
Reprezentarea unei laturi in lista de laturi:

- ymax_latură şi ymin_latură, folosite în testul de latură activă;
- xi - abscisa ultimului punct de intersecție al laturii; initial, xi = abscisa varfului de ymin_latura - dx
- dx - constanta folosită în calculul incremental al absciselor punctelor de intersecție.

O implementare(2)




```
typedef struct{int x,y;}virf;

struct latura{    int ymax,ymin;
                  float xi,dx;
                  struct latura * urm;
                 };
typedef struct latura latura;

latura * prim,*ult;

latura **ord;
```

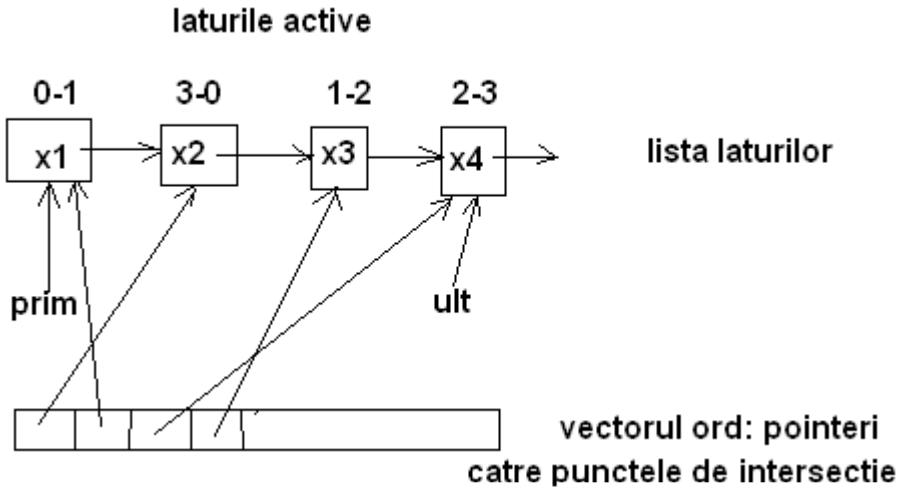


- Vectorul ord trebuie ordonat pentru fiecare linie raster,

  chiar daca nu s-a schimbat setul laturilor active.

O implementare (3)

Determina lista laturilor active
pentru noua linie raster



int set_activ()

{

* Adaugă laturile care devin active

(avans **ult** cat timp yr >=ymin_latura)

* Elimină din lista laturilor, intre prim si ult, laturile care au devenit inactive

(yr > ymax_latura)

return 1 daca s-a modificat setul laturilor active,

0 altfel;

}

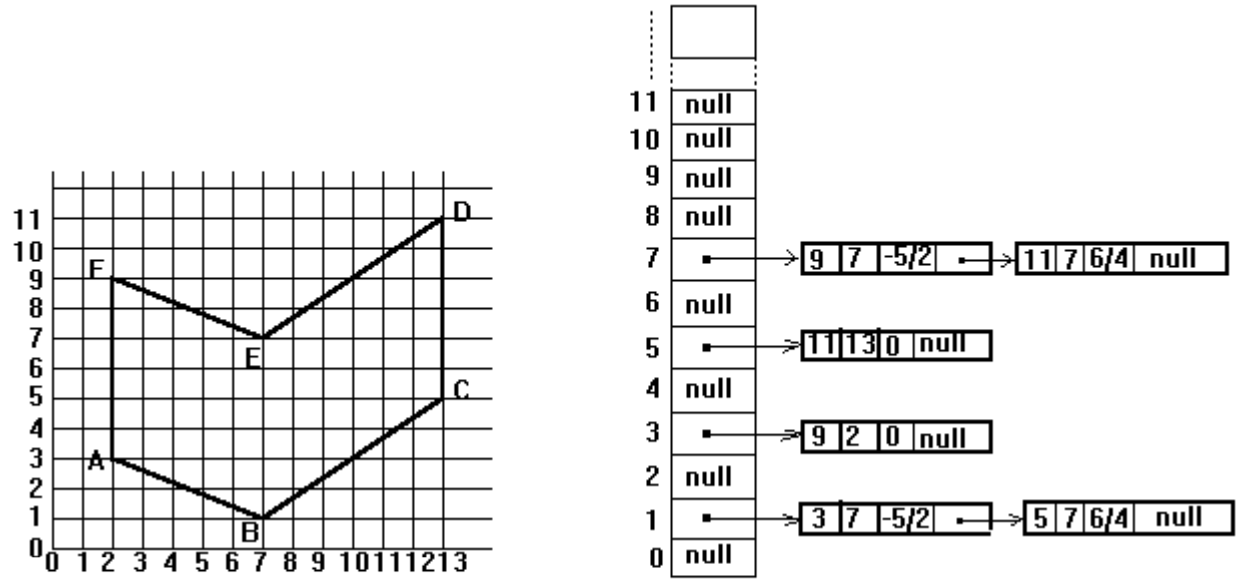
O implementare (4)

```
int colorare_poligon(int nrv, virf poly[])
{  // nrv este numarul de vârfuri ale poligonului;  poly este vectorul vârfurilor
    latura **ord; latura * k; int I, k, modif, x;
  * Creaza lista de laturi ordonata crescator dupa ymin_laturi;
  ult = prim;
 // Genereaza interiorul poligonului prin intersectii cu linii raster, de la ymin_poligon la
    ymax_poligon
 for(yr=ymin_poligon; yr < ymax_poligon; yr++)
{  modif=set_activ(); // calculeaza setul laturilor active
  * Calculeaza punctele de intersecție cu laturile active (memorate in celulele listei de
    laturi): xi = xi + dx
  if(modif) // s-a modificat setul laturilor active
    * Memoreaza in vectorul ord pointeri catre laturile active
```

O implementare (5)

```
// Ordoneza vectorul ord a.i. sa puncteze catre punctele de intersecție in ordinea
    crescatoare a absciselor
  do  {o=0;
       for(i=0; i<nrint-1; i++)
           if(ord[i]->xi > ord[i+1]->xi) {k=ord[i]; ord[i]=ord[i+1]; ord[i+1]=k;o=1;}
   }while(o);

//Coloreaza pixelii interiori poligonului de pe linia curenta

  for(i=0; i<nrint-1; i+=2)// nrint este numarul de puncte de intersectie

    for(x = (int)(ord[i]->xi+0.5); x<=(int)(ord[i+1]->xi+0.5); x++)

          putpixel(x, yr, culoare(x,yr)); // afisare fragment

}//for yr

return 1;
```

Varianta de reprezentare a listei de laturi(1)



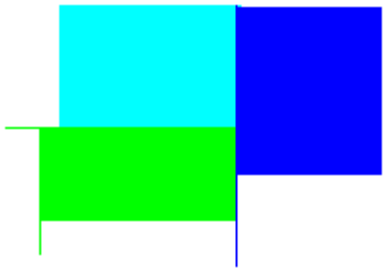
- Lista de laturi este un vector de pointeri cu un număr de intrări egal cu numărul de linii imagine, implicit ordonata crescator dupa ordonatele minime ale laturilor .
- Fiecare intrare punctează către lista laturilor care au ordonata minimă egală cu indicele intrării
- Lista laturilor cu acelaşi ymin este ordonată crescător după abscisa vârfului cu ordonata ymax.
- Pentru fiecare latură se memorează ordonata maximă , abscisa vârfului de ordonată minimă şi incrementul (1/m) folosit în calculul punctelor de intersecție.

Varianta de reprezentare a listei de laturi(2)

- Lista laturilor active este reprezentată separat, printr-o listă înlănțuită, în care laturile sunt memorate în ordinea crescătoare a absciselor punctelor de intersecție cu linia curenta.
- Având în vedere organizarea listei de laturi, timpul necesar construirii sale este mai mic decât în soluția prezentată anterior.
- Actualizarea listei de laturi active la trecerea de la o linie raster la alta poate fi efectuată mai rapid.
- Soluția este dezavantajoasă în privința spațiului de memorie ocupat de lista laturilor, ştiind că în majoritatea cazurilor numai un număr mic dintre intrările sale sunt folosite (in prezent acesta nu mai constituie un dezavantaj!).

Generarea pixelilor strict interiori

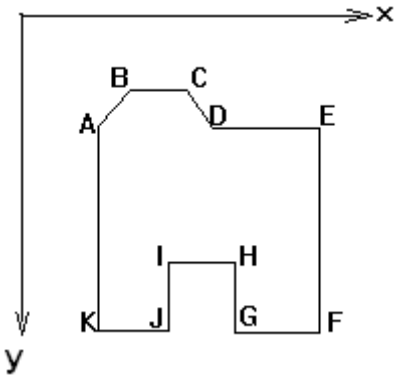
Trebuie evitata suprascrierea pixelilor de pe laturile comune poligoanelor .



Conventie:

-Pixelii de pe laturile de xmin si ymin apartin poligonului (se afiseaza)

-Pixelii de pe laturile de xmax si ymax nu se afiseaza



Datorita conventiei folosite pentru traversarea varfurilor poligonului:

- Liniile orizontale care unesc 2 varfuri de minim local vor fi afisate
- Liniile orizontale care unesc 2 varfuri de maxim local nu vor fi afisate
  - ➢In acest fel se respecta conventia

Generarea pixelilor strict interiori

Pentru punctele de xmin si xmax ale segmentelor interioare poligonumui:

- fiecare segment interior poligonului este un interval inchis in punctul de abscisa minima si deschis in punctul de abscisa maxima, deci se respecta conventia.

```
for(x = (int)(ord[i]->xi+0.99999); x<=(int)(ord[i+1]->xi-0.000001); x++)
    putpixel(x, yr, culoare(x,yr));
```

❖ Generarea numai a pixelilor strict interiori nu dă rezultate bune atunci când unghiul dintre două laturi este foarte mic. Este posibil ca între cele două laturi să fie afişat un singur pixel sau chiar niciunul.

Generarea interiorului unui poligon folosind un şablon

Modificarea implementarii anterioare pentru generarea interiorului folosind un sablon

```
int sablon[8][8]={255,0,0,0,0,0,0,255, 0,255,0,0,0,0,255,0, 0,0,255,0,0,255,0,0,
                  0,0,0,255,255,0,0,0,0,0,0,255,255,0,0,0, 0,0,255,0,0,255,0,0,
                  0,255,0,0,0,0,255,0,255,0,0,0,0,0,0,255
                 }; //matrice de nivele de intensitate
int hs=8, ls=8, lin, x;
// sablonul se aplica incepand din (x=0, y=0): originea imaginii
void colorare_poligon(int nrv,virf poly[] )
{
 for(yr=ymin_poligon; yr < ymax_poligon; yr++)
 { ……..
   lin=yr%hs; // linia din matricea sablon folosita la colorare
   for(i=0;i<nrint-1;i+=2)
       for(x=(int)(ord[i]->xi+0.5; x<=(int)(ord[i+1]->xi+0.5); x++)
                putpixel(x,yr,sablon[lin][x%ls]);
  }
```

Cazul general: sablonul se mapeaza pe un dreptunghi dat, iar aplicarea sablonului consta in multiplicarea dreptunghiului sablon pe suprafata poligonului, incepand dintr-un punct dat.