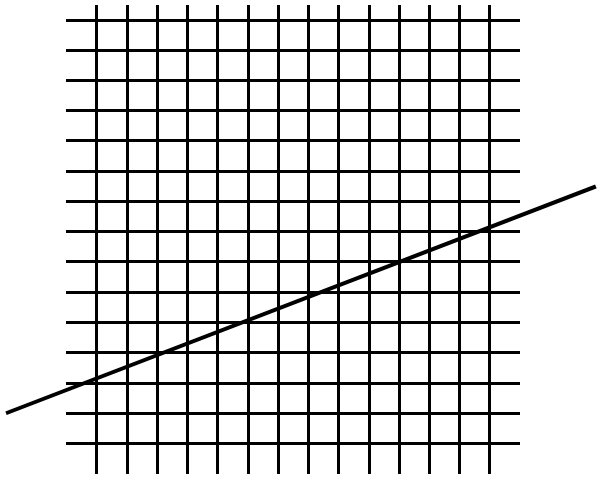
Σχεδίαση γραμμών στην οθόνη του Υπολογιστή
Γέμισμα πολυγώνων
Αποκοπή

1η προσέγγιση: εφαρμογή της εξίσωσης της ευθείας

- Έστω ότι επιθυμούμε να σχεδιάσουμε ευθεία $y=mx+b$
- Προσέγγιση 1: δίνουμε τιμές στο x παράγεται το y και προχωρούμε σε εφαρμογή του round() για τα pixel που θα πάρουν τιμή



```
For x=x0 to x1 do
  - y=m*x+b
  - putpixel(x,round(x,y))
end
```

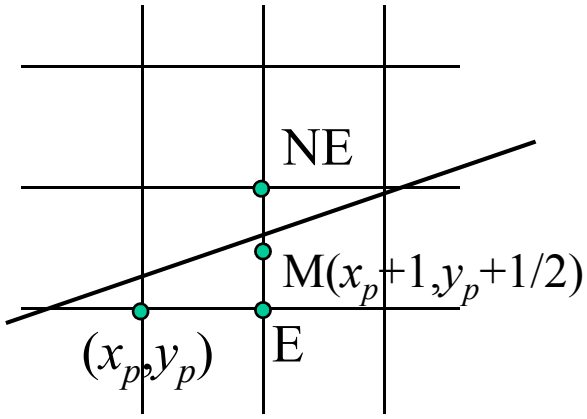
2η προσέγγιση: Βασικός αυξητικός (incremental) αλγόριθμος

```
x=x0; y=y0;
m=(y1-y0)/(x1-x0);
c=(1/m);
If m<1 then
   for x=x0 to x1
        putpixel(x,round(y));
        y=y0+m;
   end;
If m>1 then
   for y=y0 to y1
        x=x0+c;
        putpixel(round(x),y);
   end;
end;
```

3η προσέγγιση: Αλγόριθμος του μέσου σημείου (παραλλαγή του Bresenham)

- Είσοδος: αρχικός και τελικό σημείο ευθύγραμμου τμήματος (x_1,y_1), (x_2,y_2)
- Η εξίσωση της ευθείας: $F(x,y)=dy.x-dx.y+B.dx=0$. Πρέπει $dy>0$.

- Έστω ότι ο αλγόριθμος έχει εντοπίσει το σημείο (x_p, y_p) και ζητάμε το επόμενο.
- Το επόμενο θα είναι είτε το E, είτε το NE, ανάλογα με το πρόσημο της δύναμης του μέσου M του E-NE ως προς την εξίσωση της ευθείας:
  - $F(M)=a*(x_p+1)+b*(y_p+1/2)+c$
  - Αν $F(M)>0$ →NE,
  - Αν $F(M)<0$ →E
- Η φιλοσοφία του αλγόριθμου: *Η νέα τιμή του F(M) μπορεί να υπολογιστεί από την προηγούμενη, χωρίς πλήρεις πράξεις*



- Αν επιλεγεί το E→ $F\left(x_p+2, y_p+\frac{1}{2}\right)=F(M)+a \Rightarrow d=d+a$
- Αν επιλεγεί το NE→ $F\left(x_p+2, y_p+\frac{3}{2}\right)=F(M)+a+b \Rightarrow d=d+a+b$
- Αρχικοποίηση του αλγορίθμου:
  - Το (x_1,y_1) αποτελεί σημείο του ευθ. Τμήματος

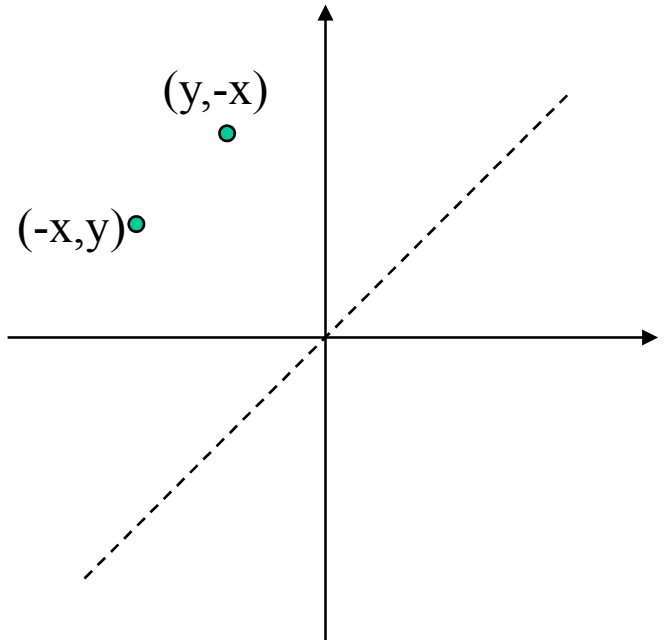
    $$F\left(x_0+1, y_0+\frac{1}{2}\right)=a(x_0+1)+b\left(y_0+\frac{1}{2}\right)+c=F(x_0, y_0)+a+\frac{b}{2}=a+\frac{b}{2} \Rightarrow$$

    $$d=a+\frac{b}{2}$$

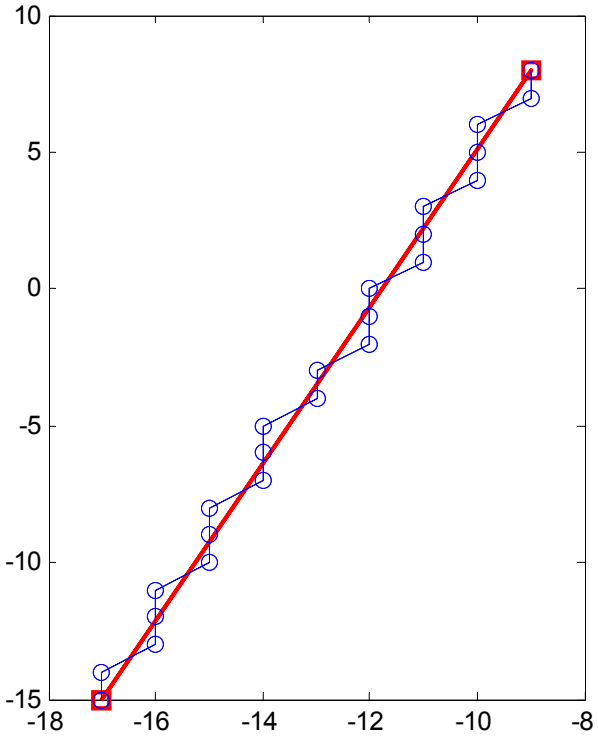
  - Για να επιτύχουμε προσθέσεις με ακέραιους μόνο, θέτουμε $F(x,y)=2ax+2by+2c$, $d=F(M)$ και $d=2a+b$

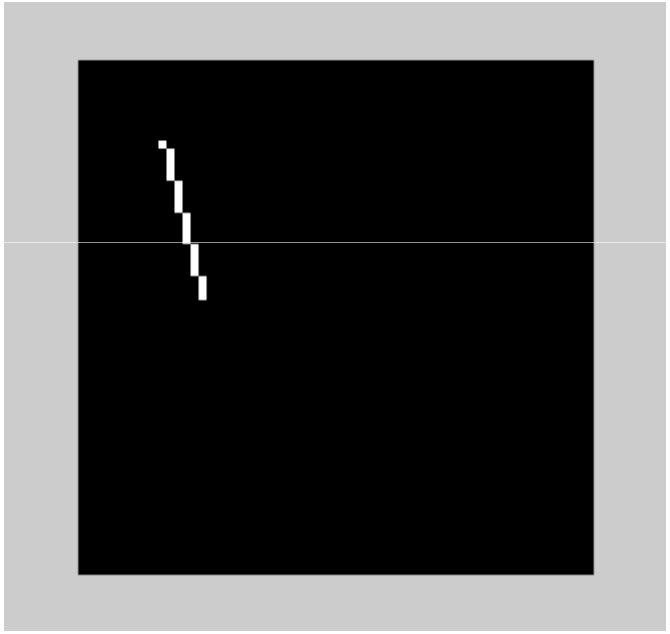
```
dx=x1-x0;
dy=y1-y0;
d=2*dy-dx;      //αρχικοποίηση του F(M)
incE:=2*dy; incNE=2*(dy-dx);
x=x0; y=y0
Putpixel(x,y,255);
While (x<x1) do
   if d<=0 →d:=d+incE; x++;
   if d>0 →d:=d+incNE; x++; y++;
   putpixel(x,y,255);
end;
```

- Ο αλγόριθμος ισχύει για κλίση m: 0<m<1. Για τις λοιπές περιπτώσεις εκτελούμε κατά σειρά τους εξής ελέγχους:
  - Αν dx<0 → εναλλάσουμε το πρώτο και το τελευταίο σημείο του ευθύγραμμου τμήματος
  - Αν dy<0 → καλείται η συνάρτηση σχεδίασης με τα συμμετρικά σημεία ως προς τον άξονα X.
  - Αν dx> dy → καλείται η συνάρτηση με τα συμμετρικά των σημείων ως προς την κύρια διαγώνιο



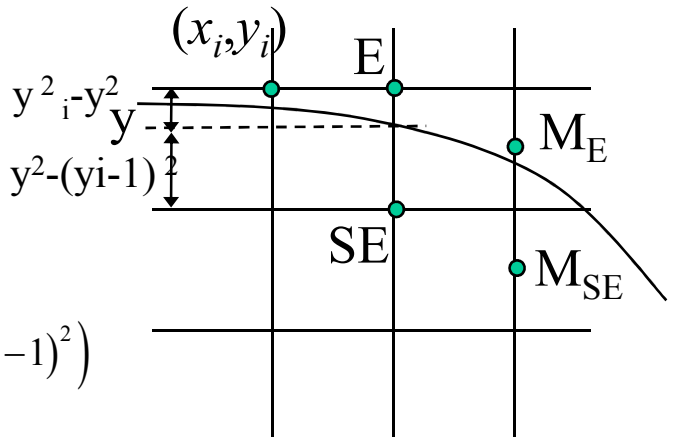
Παράδειγμα εφαρμογής του αλγόριθμου μέσου σημείου





Σχεδίαση κύκλων: Αλγόριθμος του Bresenham

- Σκοπός: να βρεθούν τα σημεία ενός κύκλου με κέντρο το (0,0) και ακτίνα *R*
- Είσοδος: *R,* Εξοδος: συντεταγμένες pixel που ανήκουν στον κύκλο.
- Εστω ότι έχει επιλεγεί το (x_i,y_i) στην προηγούμενη επανάληψη του αλγορίθμου.
- Υπολογίζεται η ποσότητα: $e_i = \left(y_i^2 - y^2\right) - \left(y^2 - \left(y_i - 1\right)^2\right)$
- Είναι προφανές ότι αν e_i<=0 επιλέγεται το σημείο E, αλλιώς επιλέγεται το σημείο SE. Για να ισχύει αυτό, ο αλγόριθμος εφαρμόζεται στο 2ο ογδοημόριο του επιπέδου (y>=x>0).
- Ο αλγόριθμος επιχειρεί να υπολογίζει γρήγορα την τιμή e_i του βάσει της τιμής κατά την προηγούμενη επανάληψη (αυξητικός υπολογισμός)



$$e_i = \left(y_i^2 - y^2\right) - \left(y^2 - \left(y_i - 1\right)^2\right)$$

- Υπολογισμός της τρέχουσας τιμής του e_i

$$e_i = y_i^2 - y^2 - \left(y^2 - (y_i - 1)^2\right)$$

$$y^2 = R^2 - (x_i + 1)^2$$

$$\Rightarrow e_i = y_i^2 - R^2 + (x_i + 1)^2 - \left(R^2 - (x_i + 1)^2 - (y_i - 1)^2\right) =$$

$$y_i^2 - 2R^2 + 2(x_i + 1)^2 + (y_i - 1)^2$$

- Αν επιλεγεί το E→ υπολογισμός του e_{i+1}

$$e_{i+1} = y_i^2 - 2R^2 + 2(x_i + 2)^2 + (y_i - 1)^2 = e_i + 4x_i + 6$$

- Αν επιλεγεί το SE→ υπολογισμός του e_{i+1}

$$e_{i+1} = (y_i - 1)^2 - 2R^2 + 2(x_i + 2)^2 + (y_i - 2)^2 = e_i + 4x_i + 6 - 4y_i + 4$$

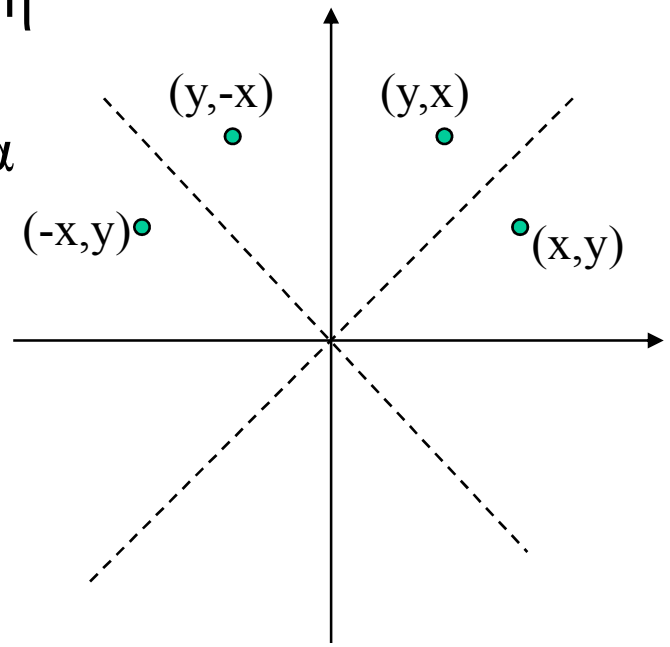
- Αρχικοποίηση e_0 για το σημείο (0,R)

$$e_0 = R^2 - 2R^2 + 2 + (R - 1)^2 = 3 - 2R$$

Ψευδοκώδικας του Αλγόριθμου του Bresenham για κύκλο

```
x=0
y=R
e=3-2*R
while (x<=y)
   if e<=0
       e=e+4x+6
       x=x+1
   else
       e=e+4x+6-4y+4
       x=x+1
       y=y-1
   end
end
```

- Ο αλγόριθμος ισχύει για κλίση m: 0<m<1. Για τις λοιπές περιπτώσεις ισχύει συμμετρία ογδοημορίων:



Σχεδίαση κύκλων: Αλγόριθμος του μέσου σημείου

- Σκοπός: να βρεθούν τα σημεία ενός κύκλου με κέντρο το (0,0) και ακτίνα R
- Είσοδος: R
- Η εξίσωση του κύκλου: $d=F(x,y)=x^2+y^2-R^2=0$.
  - Αν $F(x,y)<=0$ → (x,y) εντός του κύκλου.
  - Αν $F(x,y)>0$ → (x,y) εκτός του κύκλου.
- Ο αλγόριθμος εφαρμόζεται για το 2^o ογδοημόριο ($y>x>0$) με έναρξη το σημείο $(0,R)$.
- Επιλέγεται αρχικό σημείο (x_p,y_p) και υπολογίζεται η δύναμη σημείου ως προς κύκλο d για το μέσον M του τμήματος με άκρα τα πιθανά επόμενα σημεία (E, SE).
- Από το πρόσημο του d, αποφασίζεται το επόμενο σημείο και υπολογίζεται η νέα τιμή του d. Ενημερώνονται τα (x_p,y_p)
  - Η μεταβολή του d δε κάθε βήμα δεν είναι σταθερή, αλλά συνάρτηση του (x_p,y_p).
- Ο Αλγόριθμος συνεχίζεται έως να συμπληρωθεί το ένα όγδοο του κύκλου.

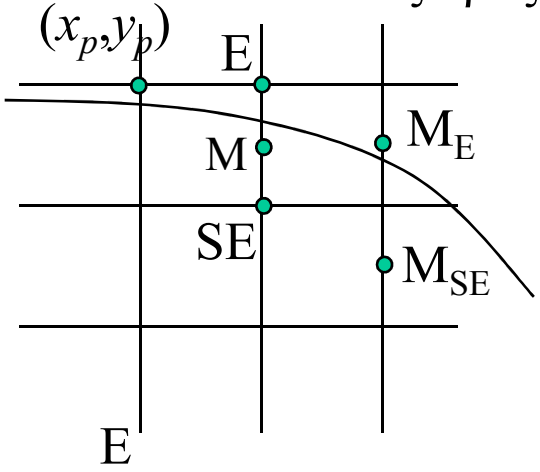
- Αν d=F(M)<0 →επιλέγεται το E σαν επόμενο μέσο σημείο→

$$F\left(x_p+2, y_p-\frac{1}{2}\right)=F\left(M\right)+2x_p+3 \Rightarrow d=d+2x_p+3$$

- Αν d=F(M)>=0 →επιλέγεται το SE σαν επόμενο μέσο σημείο →

$$F\left(x_p+2, y_p-\frac{3}{2}\right)=F\left(M\right)+2x_p-2y_p+5a \Rightarrow d=d+2x_p-2y_p+5a$$

- Αρχικοποίηση του αλγορίθμου:
  - Λαμβάνοντας το (0,R) ως πρώτο σημείο του κύκλου, η δύναμη του M ως προς το κύκλο θα είναι



$$F\left(M\right)=F\left(1, R-\frac{1}{2}\right)=\frac{5}{4}-R \Rightarrow d_{initial}=\frac{5}{4}-R$$

Ψευδοκώδικας του Αλγόριθμου του μέσου σημείου για κύκλο

```
xp:=0;
yp:=R;
D=5/4-R;
While (yp>xp)
   If d<0→  //next point E, midpoint M
        d:=d+2*xp+3;
        xp:=xp+1;
        yp:=yp;
   If d>0→  //next midpoint MSE or ME
        d:=d+2*(xp-yp)+5;
        xp:=xp+1;
        yp:=yp-1;
   putpixel(xp,yp);
end;
```

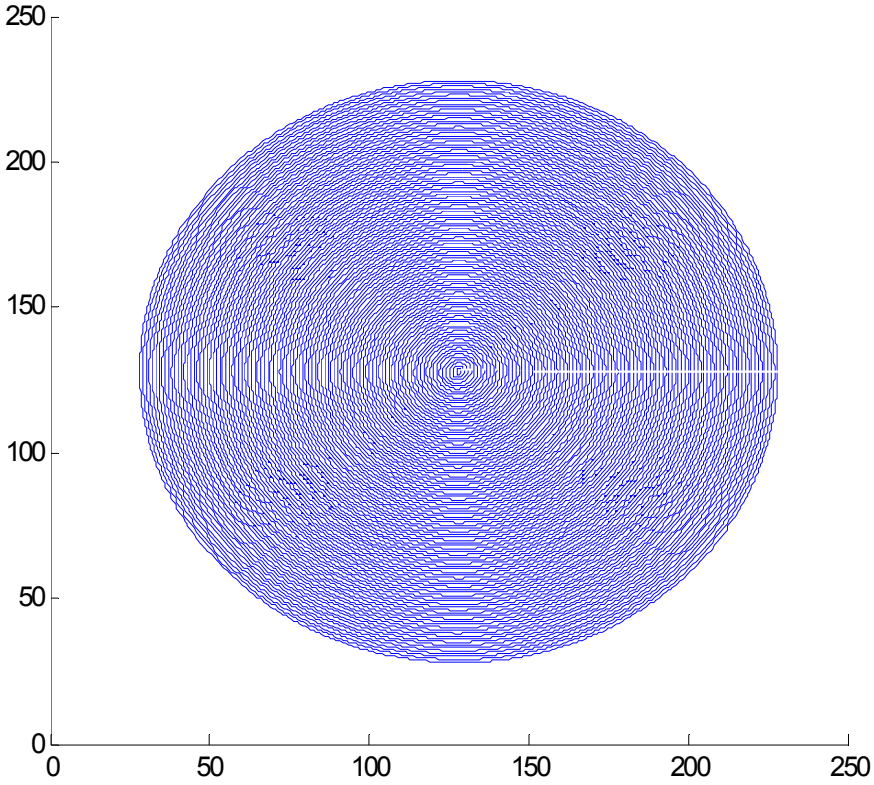
Σχεδίαση κύκλων: Αλγόριθμος του μέσου σημείου – Μεταβολές δεύτερης τάξης

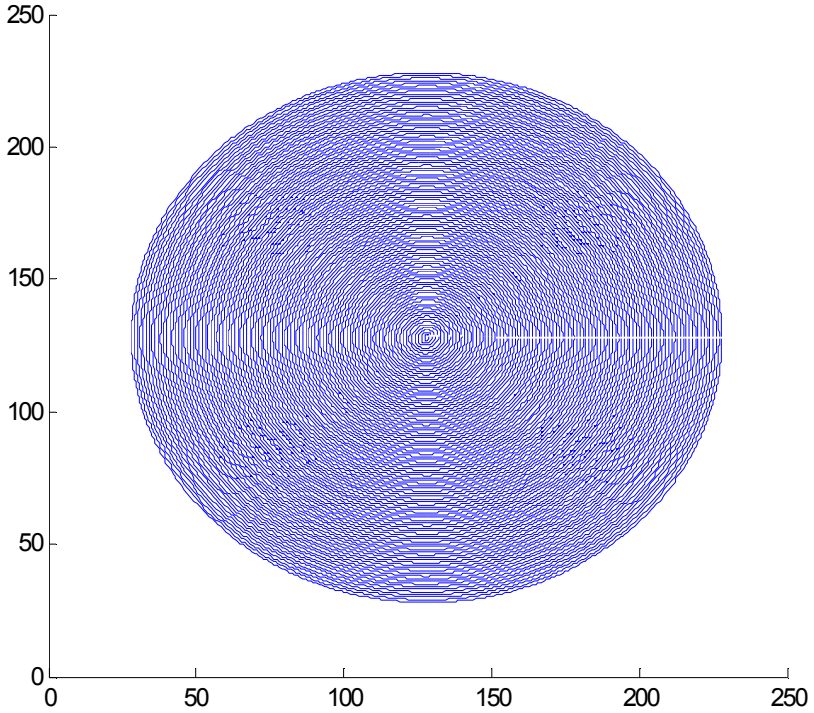
- Η μεταβολή του *d* δεν είναι σταθερή, αλλά συνάρτηση του *xp*, *yp*.
- $d<0 \rightarrow E \rightarrow \Delta(d)=2*x_p+3,\ x_p=x_p+1 \rightarrow$
  - $\Delta(d)_{new}=\Delta(d)+2 \rightarrow \Delta(\Delta(d))_E=2$
- $d>0 \rightarrow SE \rightarrow \Delta(d)=2*x_p-2*y_p+5,\ x_p=x_p+1,\ yp=yp-1 \rightarrow$
  - $\Delta(d)_{new}=\Delta(d)+4 \rightarrow \Delta(\Delta(d))_{SE}=4$
- → Η μεταβολή της μεταβολής του d είναι σταθερή.
- Αρχικές τιμές Δ(Δ(d)) στο σημείο (0,R):
  - $\Delta(\Delta(d))_E=F(M_E)-F(E)$
  - $\Delta(\Delta(d))_E=F(M_{SE})-F(E)$

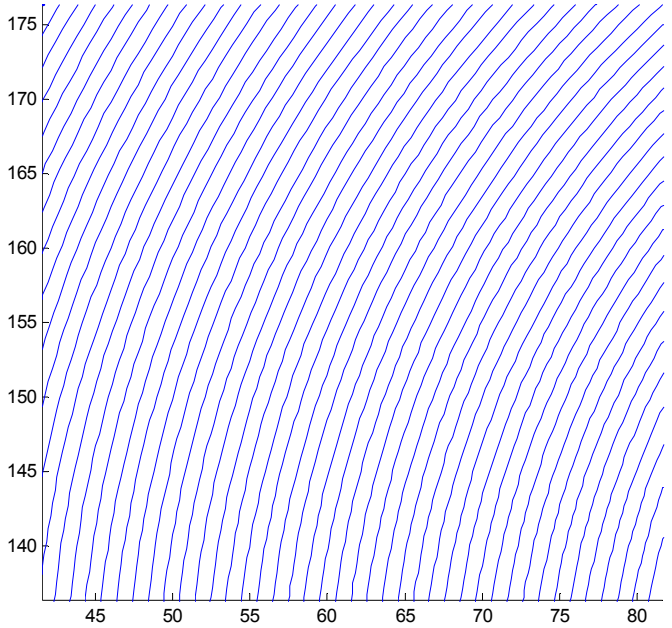
```
Xp:=0;
Yp:=R;
d=5/4-R;
ddE:=3;
ddSE:=-2*R+25/4;
While (yp>xp)
   If d<0→  //next point E, midpoint M
        d:=d+ddE;
        ddE:=ddE+3;
        ddSE:=ddSE+4;
        xp:=xp+1;
        yp:=yp;
   If d>0→  //next midpoint MSE or ME
        d:=d+ddSE;
        ddE:=ddE+3;
        ddSE:=ddSE+4;
        xp:=xp+1;
        yp:=yp+1;
   putpixel(xp,yp);
end;
```

Ταυτοποίηση συχνοτήτων - Aliasing

- Το pixel της μνήμης οθόνης, καθώς και της πλεγματικής οθόνης, έχει μη μηδενικές διαστάσεις.
- Κατά συνέπεια, όταν σχεδιάζεται μία γραμμή στην οθόνη, δειγματοληπτείται, με συνέπεια λεπτομέρειες μικρότερης διάστασης από τη διάσταση του pixel (μεγάλες χωρικές συχνότητες) να απεικονίζονται σαν αργά μεταβαλλόμενα σχήματα (χαμηλές χωρικές συχνότητες).
- Το φαινόμενο αυτό λέγεται ταυτοποίηση (aliasing) συχνοτήτων.



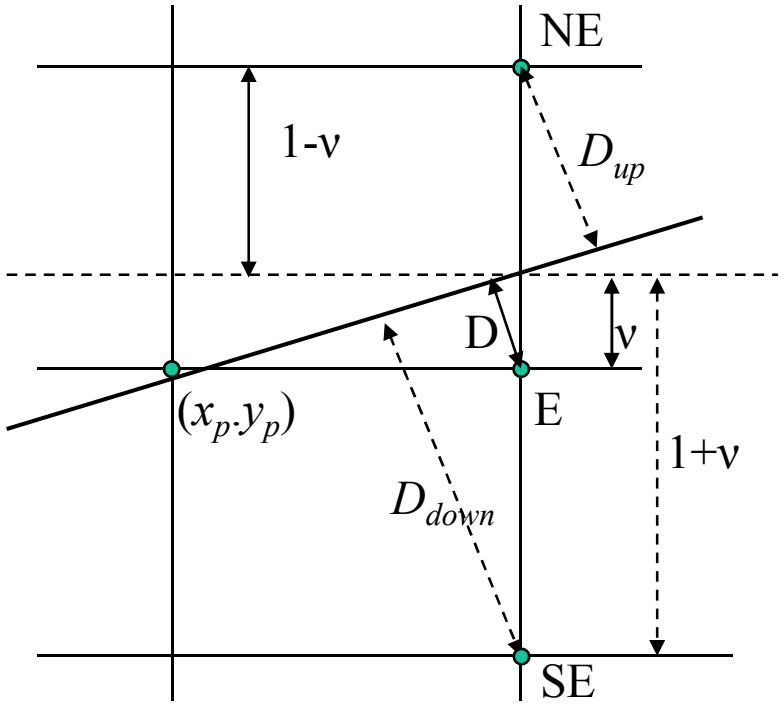




Μείωση του φαινομένου της ταύτισης, μέσω αύξησης της ανάλυσης της μνήμης οθόνης

Τεχνικές αντιταυτοποίησης (antialiasing)

- Τα pixel που παράγονται από τον αλγ. γραμμών παίρνουν δυαδικές τιμές.
- Το φαινόμενο aliasing αντιμετωπίζεται δίνοντας τιμές στο pixel του αλγ. σχεδίασης και σε γειτονικά, βάσει της απόστασης *D* τους από το κέντρο της γραμμής
- Το *D* πρέπει να υπολογίζεται αθροιστικά, βάσει ποσοτήτων του αλγ. Σχεδίασης.



$$D = v\cos\phi = \frac{vdx}{\sqrt{dx^2 + dy^2}}$$

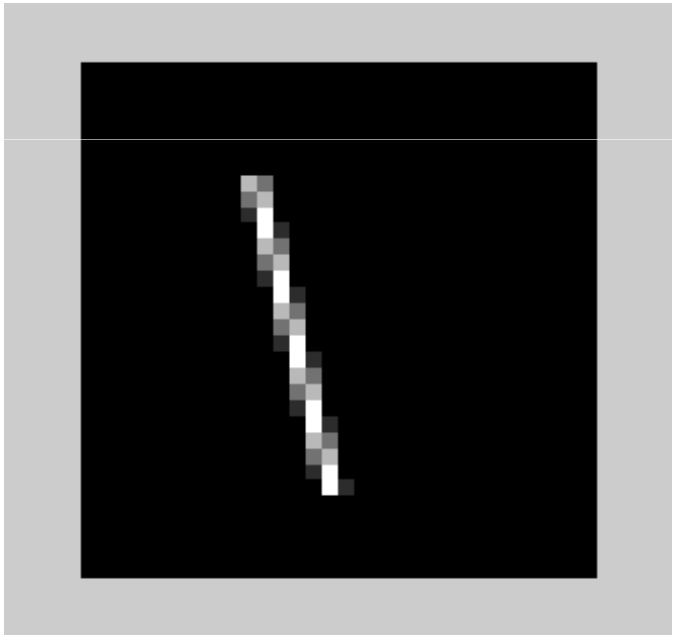
$$v = y - y_p = -\frac{ax + c}{b} - y_p \Rightarrow vdx = v(-b) = a(x_p + 1) + by_p + c = F(x_p + 1, y_p)\frac{1}{2}$$

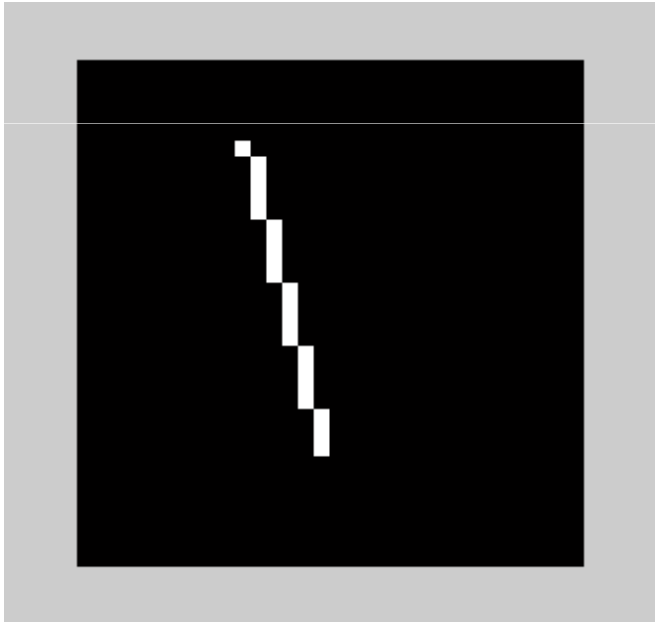
$$E \rightarrow 2vdx = F(x_p + 1, y_p) = F(M) + dx$$

$$NE \rightarrow 2vdx = F(x_p + 1, y_p + 1) = F(M) - dx$$

$$D_{down} = \frac{(1+v)dx}{\sqrt{dx^2 + dy^2}} \qquad D_{up} = \frac{(1-v)dx}{\sqrt{dx^2 + dy^2}}$$

```
dx=x1-x0;
dy=y1-y0;
d=2*dy-dx;        //αρχικοποίηση του F(M)
incE:=2*dy; incNE=2*(dy-dx);
2v_dx:=0;
invDenom:=1/(2*sqrt(dx*dx+dy*dy));
2dx_invDenom:=2*dx*invDenom;
x=x0; y=y0;
Putpixel(x,y,255);
While (x<x1) do
   if d<=0 →2n_dx:=d+dx; d:=d+incE; x++;
   if d>0 → 2n_dx:=d-dx; d:=d+incNE; x++; y++;
   putpixel(x,y,f(2v_dx*invDenom));
   putpixel(x,y+1,f(2dx_invDenom-2v_dx*invDenom));
   putpixel(x,y-1,f(2dx_invDenom+2v_dx*invDenom));
end;
```





Αντιταυτοποίηση με μεταφιλτράρισμα

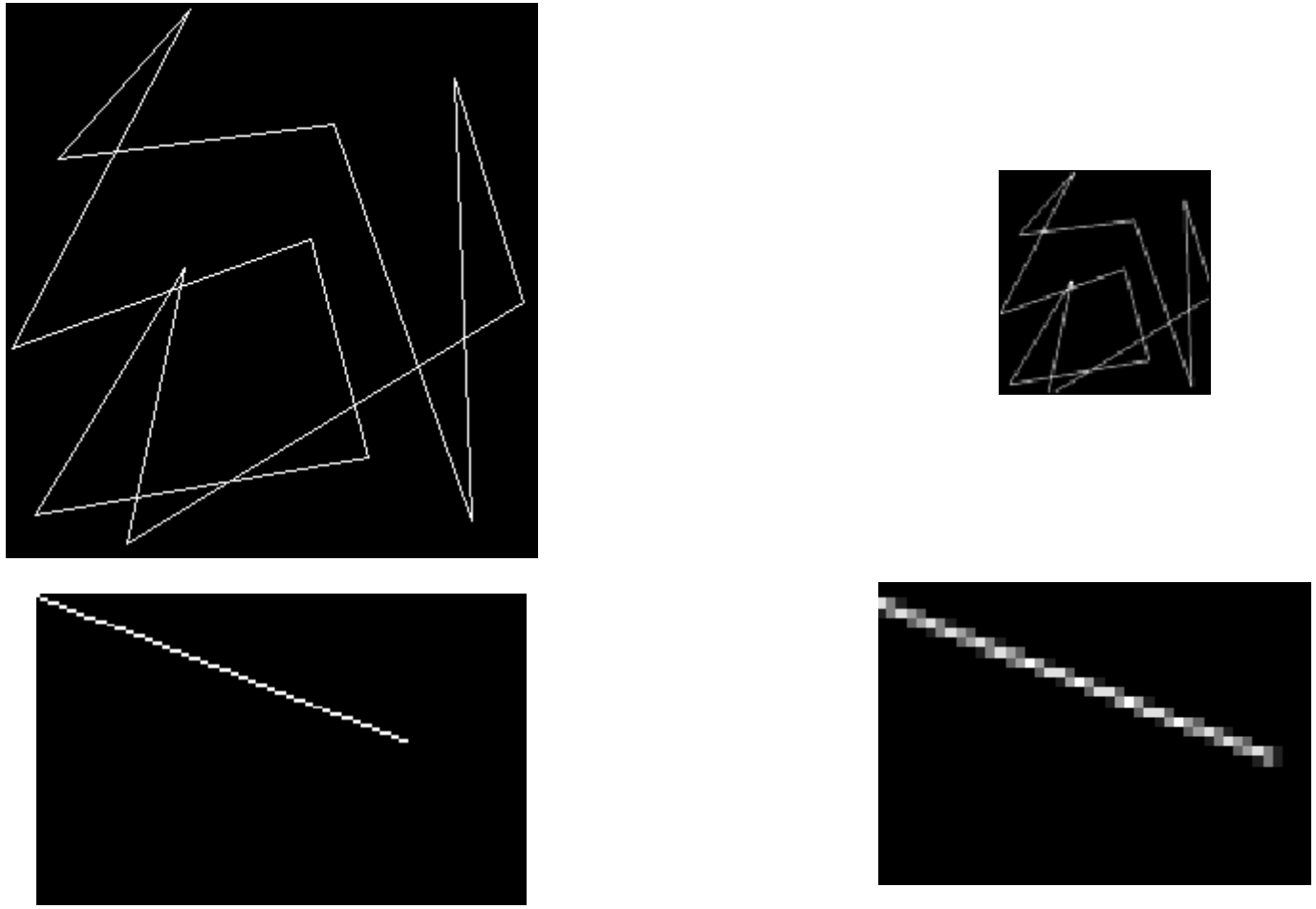
- Το μειονέκτημα της προηγούμενης μεθόδου είναι ότι δεν μπορεί να χειριστεί >1 γραμμές που περνούν από το ίδιο pixel.
- Με την τεχνική του μεταφιλτραρίσματος (metafiltering), χρησιμοποιούμε μνήμη οθόνης με διαστάσεις πολλαπλάσιες (κατά ακέραιο περιττό αριθμό) από αυτές της πλεγματικής οθόνης.
- Η σχεδίαση γίνεται στη μνήμη οθόνης και στη συνέχεια η μνήμη οθόνης αποκτά μικρότερες διαστάσεις, βάσει φιλτραρίσματος με μάσκα:

- Αν η μνήμη οθόνης I έχει a=2*k+1 φορές μεγαλύτερη κάθε μία διάσταση από την τελική εικόνα I' (πλεγματική οθόνη), τότε
- Όπου η μάσκα m φιλταρίσματος αποτελεί μία διακριτή προσέγγιση της 2D γκαουσιανής (2*k+1)x(2*k+1)
- Φροντίζουμε ώστε $\Sigma\Sigma m(p,q)=1$.
- *Βασικό μειονέκτημα της προσέγγισης*: υπολογιστικά ακριβή.

$$I'(i,j)=\sum_{p=-k}^{k}\sum_{q=-k}^{k} I(ai-p,aj-q)m(p,q)$$

$$k=1 \Rightarrow m=\frac{1}{16}\begin{pmatrix}1&2&1\\2&4&2\\1&2&1\end{pmatrix}$$

$$k=2 \Rightarrow m=\frac{1}{81}\begin{pmatrix}1&2&3&2&1\\2&4&6&4&2\\3&6&9&6&3\\2&4&6&4&2\\1&2&3&2&1\end{pmatrix}$$



(α) (β)

Παράδειγμα antialiasing με μνήμη οθόνης 256x256 (α) και πλεγματική οθόνη 128x128 (β).

Γέμισμα πολυγώνου: Αναδρομικός αλγόριθμος

- Προϋποθέσεις: εικόνα bitmap με περίγραμμα πολυγώνου με σταθερή τιμή, ή pixel background με σταθερή τιμή
- Έξοδος: τα pixel του πολυγώνου με σταθερή τιμή (διαφορετική)

```
• Flood_fill(x0,y0,bkc,fc){
   – If I(x0,y0)==bkc AND I(x0,y)!=fc {
       • Flood_fill(x0+1,y0,bkc,fc);
       • Flood_fill(x0,y0+1,bkc,fc);
       • Flood_fill(x0-1,y0,bkc,fc);
       • Flood_fill(x0,y0-1,bkc,fc);
   – }
• }
```

- Λεπτομέρειες υλοποίησης:
  - Η περίμετρος δεν πρέπει να έχει τρύπες.
  - Ακόμα και αν δεν έχει τρύπες, ο αλγόριθμος είναι δυνατό να ξεφύγει αν η αναδρομή γίνει με 8 γειτονικά pixel αντί με 4.
  - Χρειάζεται προσοχή στον τρόπο που μεταφέρεται η εικόνα I σαν όρισμα.

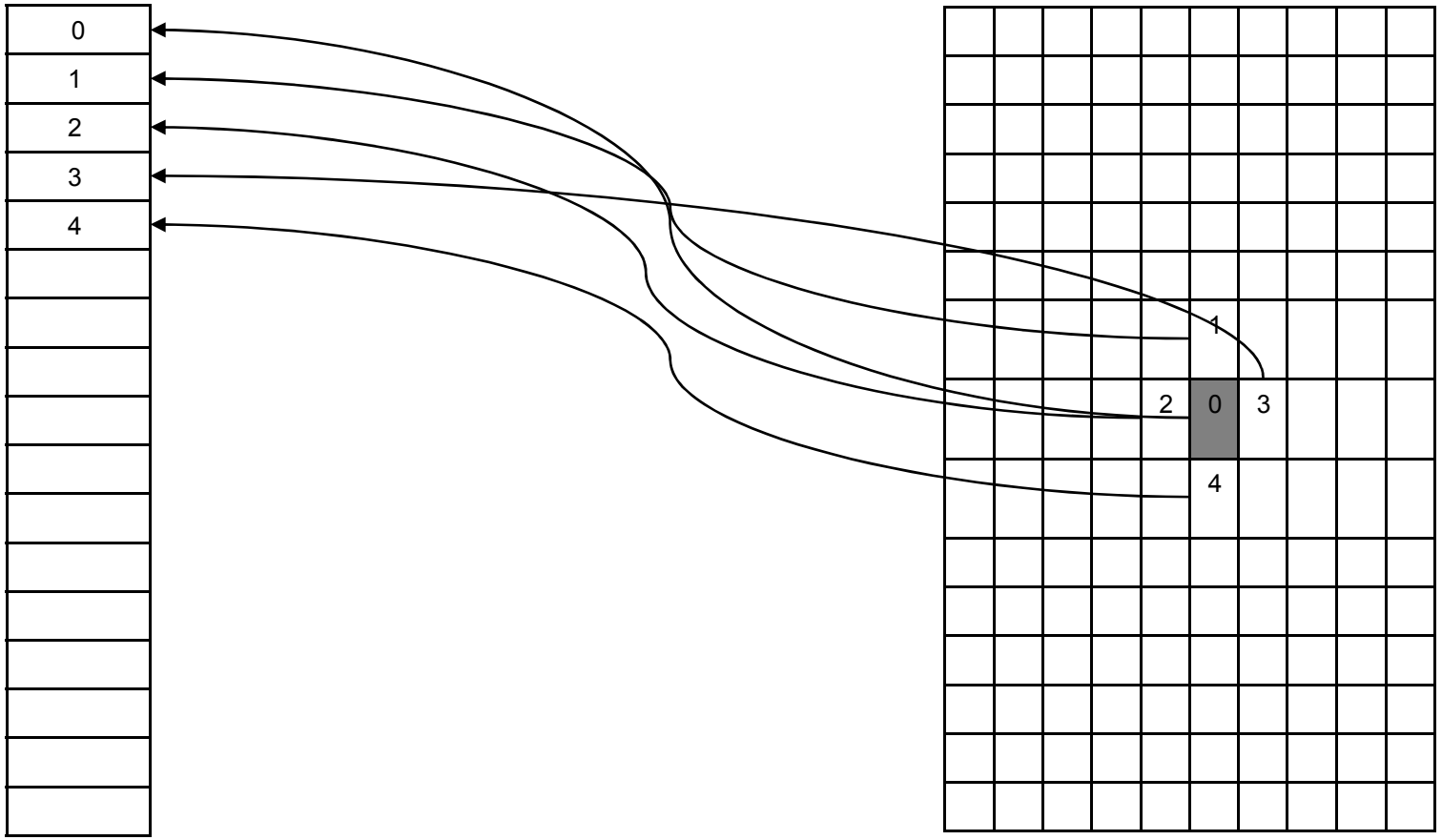
Υλοποίηση του Γεμίσματος πολυγώνου με ουρά

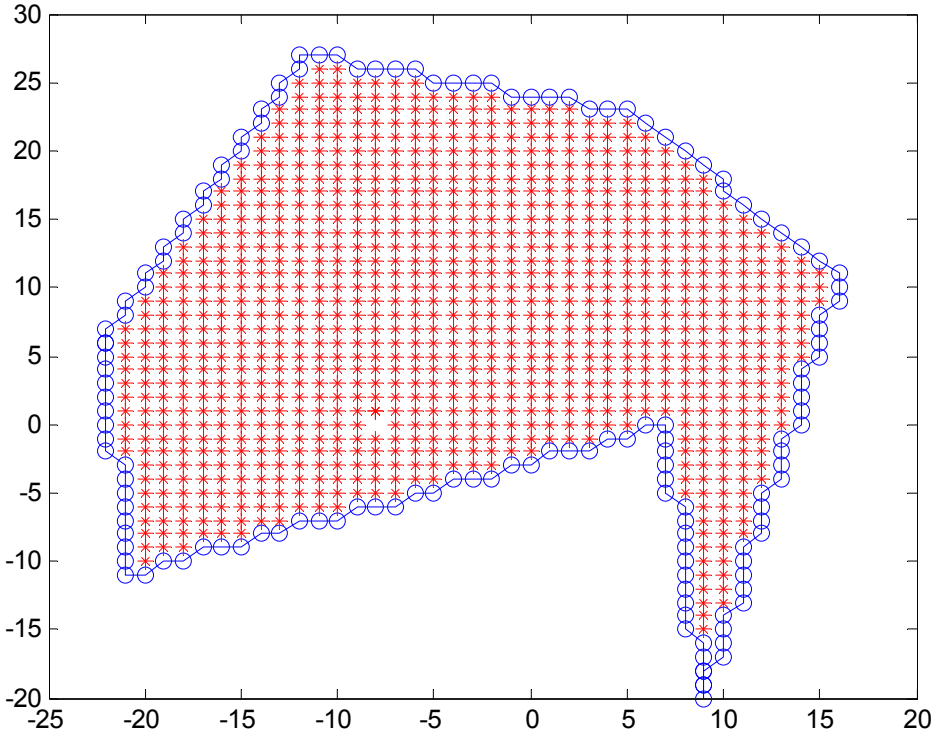
Έστω ότι επιθυμούμε να γεμίσουμε τα pixel ενός πολυγώνου

Είσοδος: τα pixel του περιγράμματος 2D array ***I***,

- Δημιουργείται άδεια ουρά Q
- Επιλέγεται σημείο p0 : p0 not in Q, p0 in polygon
- Q(1)=p0
- While (length(Q))>0
  - Label Q1 as visited I(Q(1))=color
  - Εξάγεται το Q(1) από το Q
  - For k=1 to 4 //{pk}: τα γειτονικά pixel του Q(1)
    - IF pk not in boundary and pk not visited
      - putpixel(pk,color)

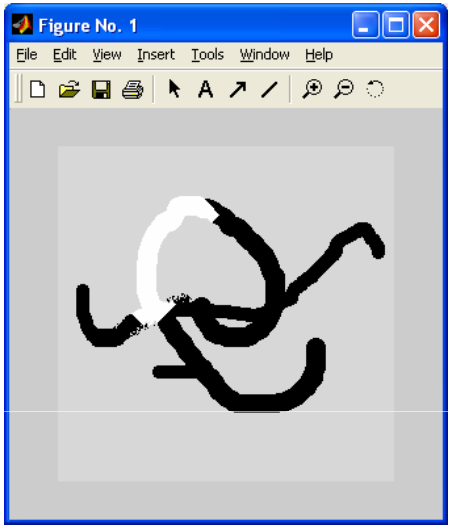
- Παρατηρήσεις:
  - Η προηγούμενη υλοποίηση δεν είναι αποτελεσματική για μεγάλο πλήθος κορυφών του περιγράμματος.
  - Οσο αυξάνει το πλήθος των pixel που βάφονται, τόσο θα καθυστερεί ο αλγόριθμος
  - Η ουρά Q αντικαθίσταται από διδιάστατο πίνακα με διαστάσεις όσο η πλεγματική οθόνη, ο οποίος αρχικοποιείται με τα pixel του περιγράμματος

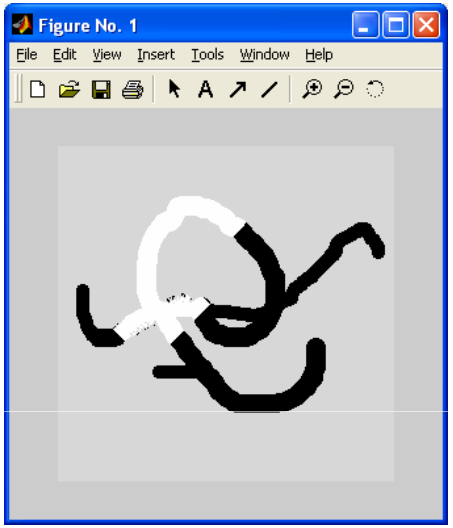

0
1
2
3
4
1
2
0
3
4

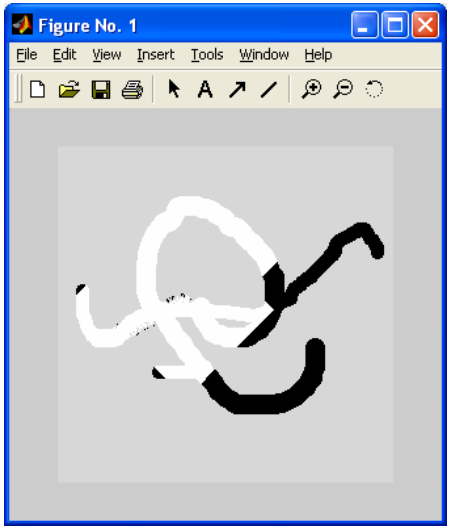


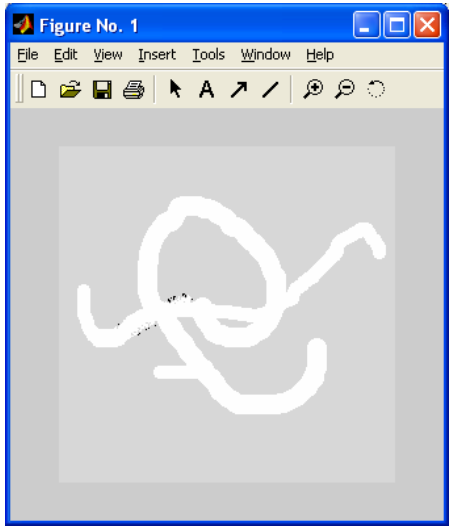
- Γέμισμα πολυγώνου με τον αλγόριθμο ουράς

Εφαρμογή του αλγόριθμού με ουρά σε συνθετική εικόνα

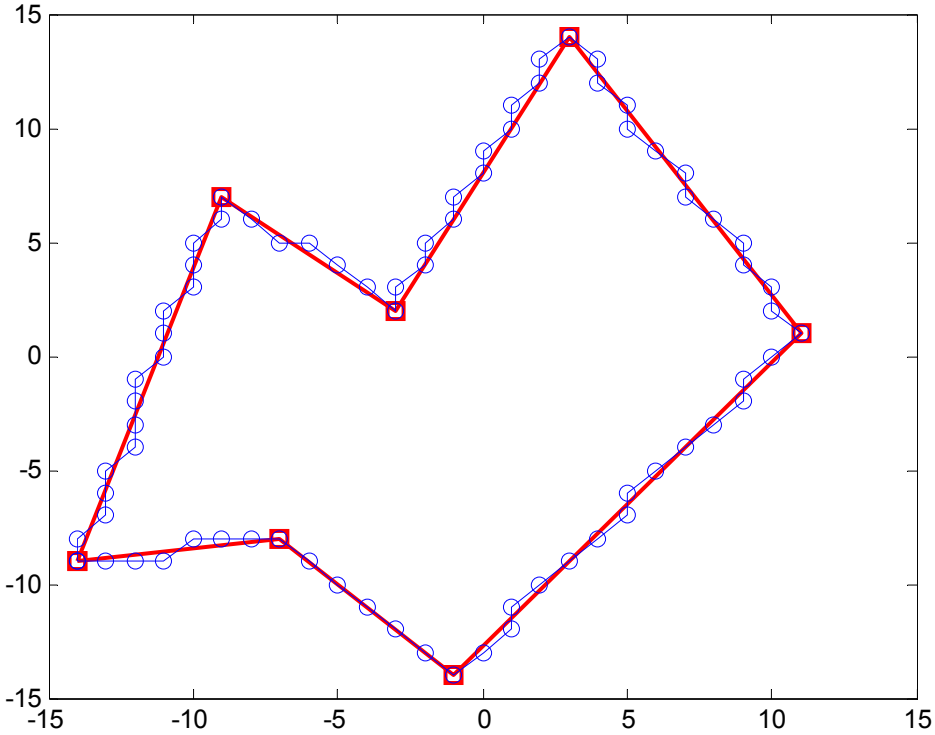








Αναπαράσταση πολυγώνου



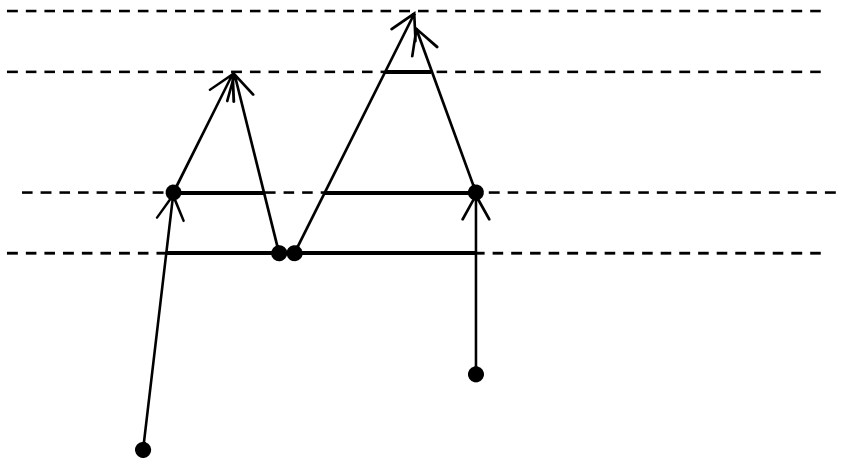
- Σαν διατεταγμένο σύνολο κορυφών (κόκκινη καμπύλη)
- Σαν διατετγμένο σύνολο όλων των σημείων του με ακέραιες συντεταγμένες

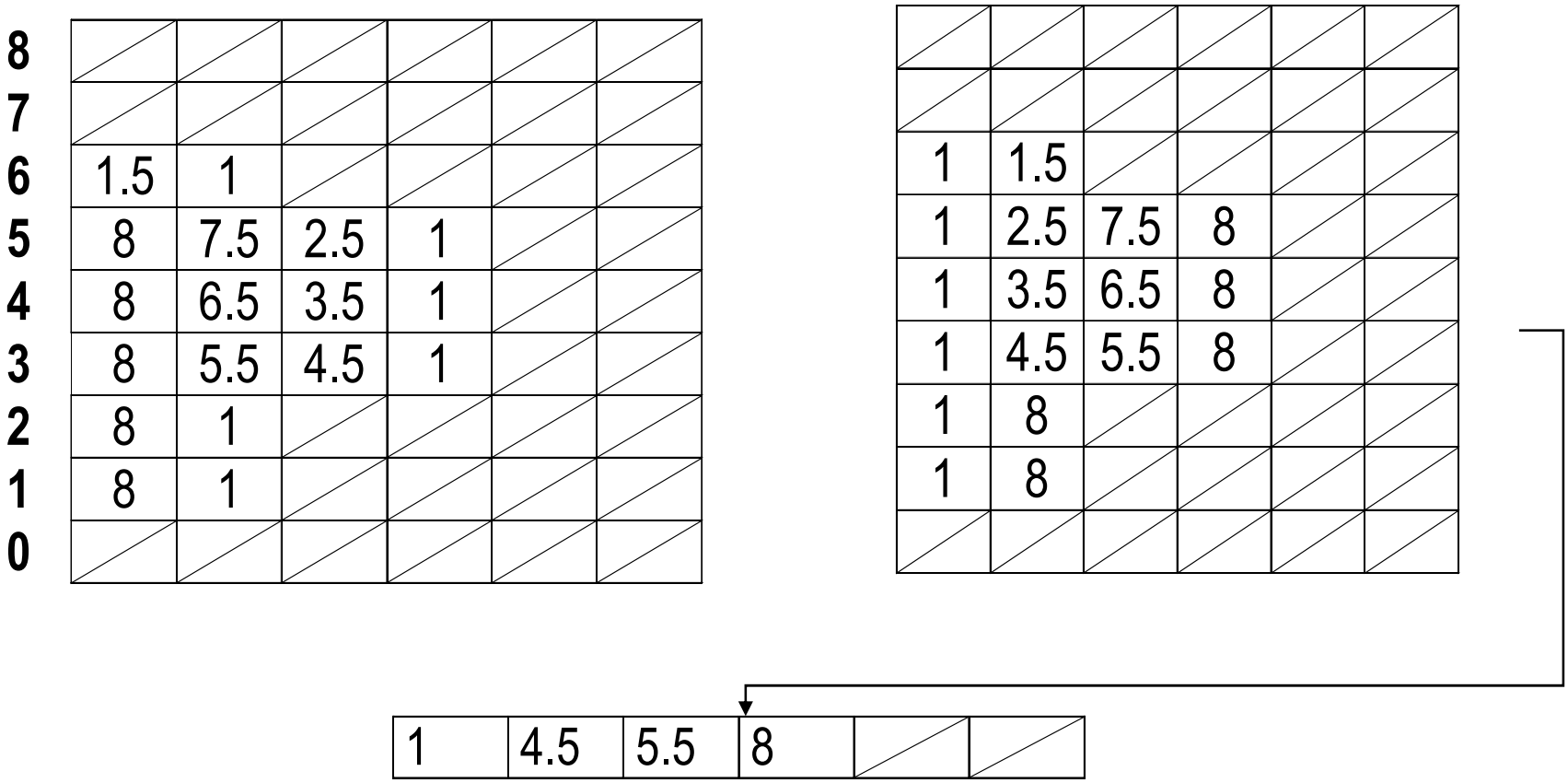
Γέμισμα πολυγώνου: αλγόριθμος *YX*

Είσοδος αλγορίθμου: διατεταγμένη σειρά από κορυφές πολυγώνου

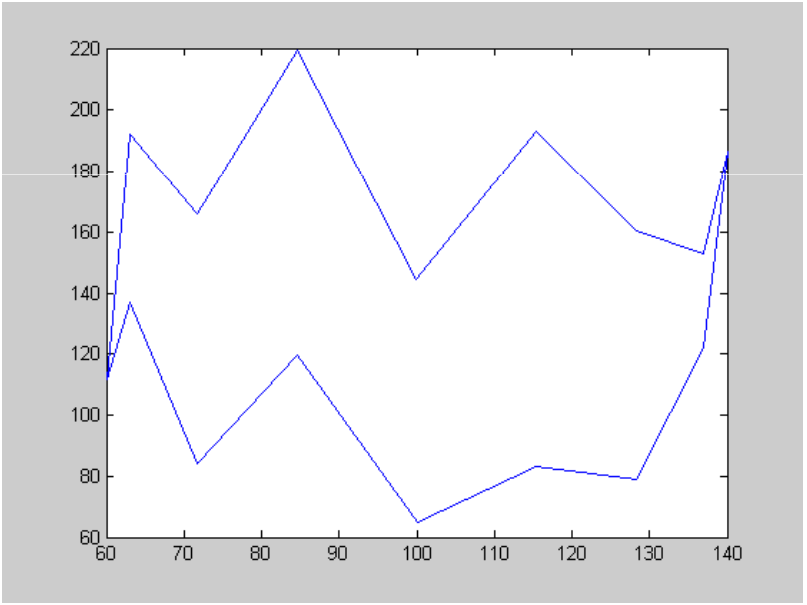
1. Ο αλγόριθμος λειτουργεί θεωρώντας μία σειρά από οριζόντιες γραμμές σάρωσης που διατρέχουν το πολύγωνο για ymin έως ymax (με βήμα 1 pixel)
2. Διατάσουμε τις γραμμές σάρωσης με φθίνουσα y
3. Για κάθε γραμμή σάρωσης y_i βρίσκουμε όλα τα σημεία τομής με τις πλευρές του πολυγώνου
4. Διατάσουμε τα σημεία τομής σε αύξουσα σειρά του x
5. Για κάθε σειρά σημείων τομής $\{x_i\}$:
   - "Ζωγραφίζουμε" όλα τα Pixel από το (x_i,y_i) έως (x_{i+1},y_i)
   - Διαγράφουμε τα x_i, x_{i+1} από τον πίνακα

- Ισοδύναμα, ορίζεται parity το οποίο παίρνει τιμή 1 με την πρώτη τομή που συναντά και σε κάθε επόμενη τομή με πλευρά πολυγώνου που συναντά το parity αλλάζει. Τα pixel ζωγραφίζονται μόνο όταν parity=1.
- Αν οι κορυφές του πολυγώνου έχουν ακέραιες συντεταγμένες: σε κάθε πλευρά, προσμετράται ως τομή μόνο η κορυφή με min(y1,y2) .
- Αν μία ακμή είναι οριζόντια, τότε δεν προσμετράται καμία κορυφή της ως τομη.




8
7
6
5
4
3
2
1
0
1.5
1
8
7.5
2.5
1
8
6.5
3.5
1
8
5.5
4.5
1
8
1
8
1
1
1.5
1
2.5
7.5
8
1
3.5
6.5
8
1
4.5
5.5
8
1
8
1
8
1
4.5
5.5
8

Παράδειγμα εφαρμογής του αλγόριθμου YX σε τυχαίο 16γωνο

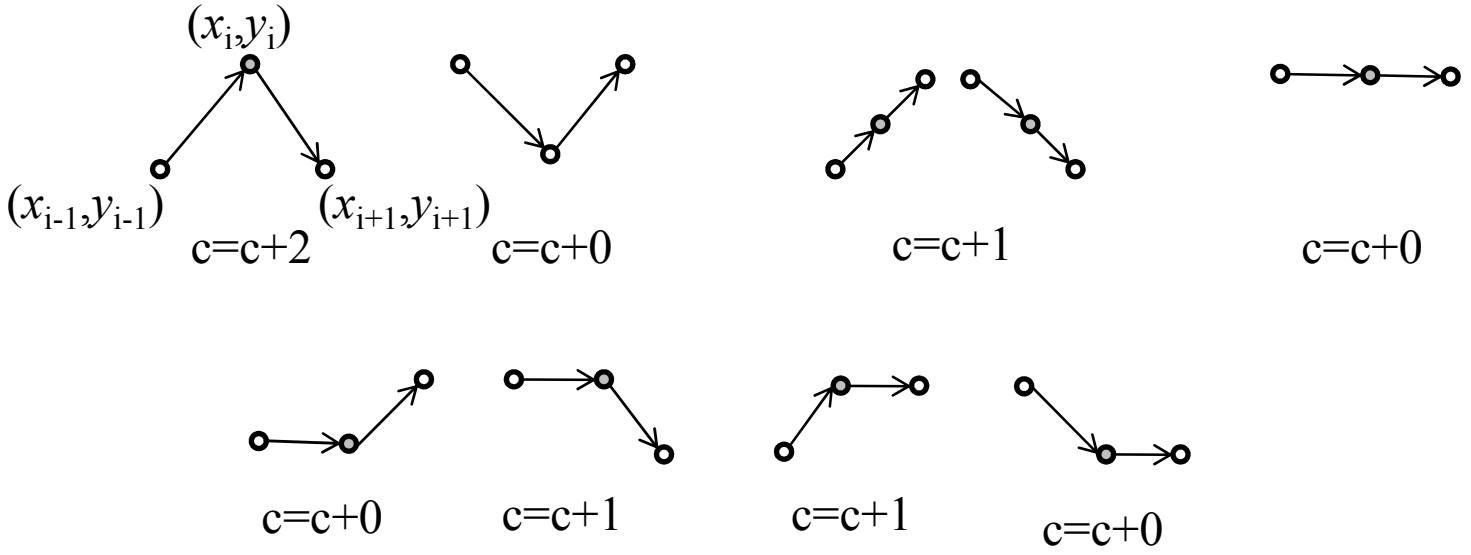


Γέμισμα πολυγώνου: αλγόριθμος *YX*

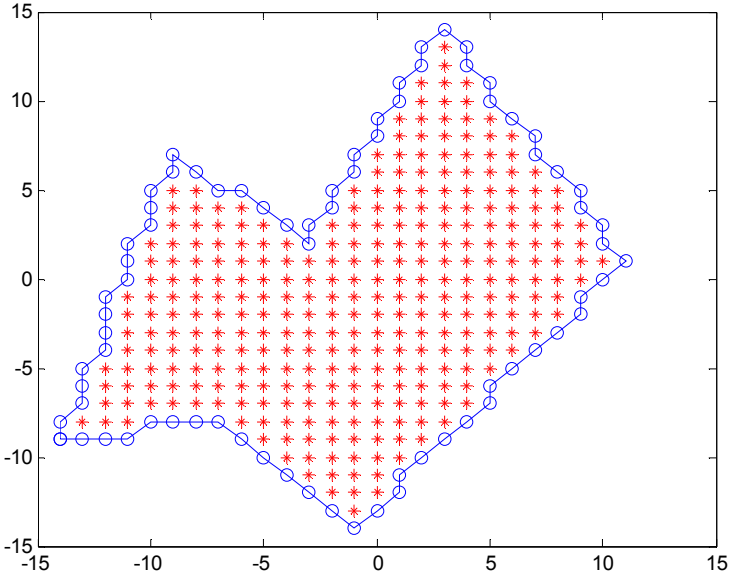
Είσοδος αλγορίθμου: διατεταγμένη σειρά από pixels που συνδέουν όλες τις κορυφές πολυγώνου

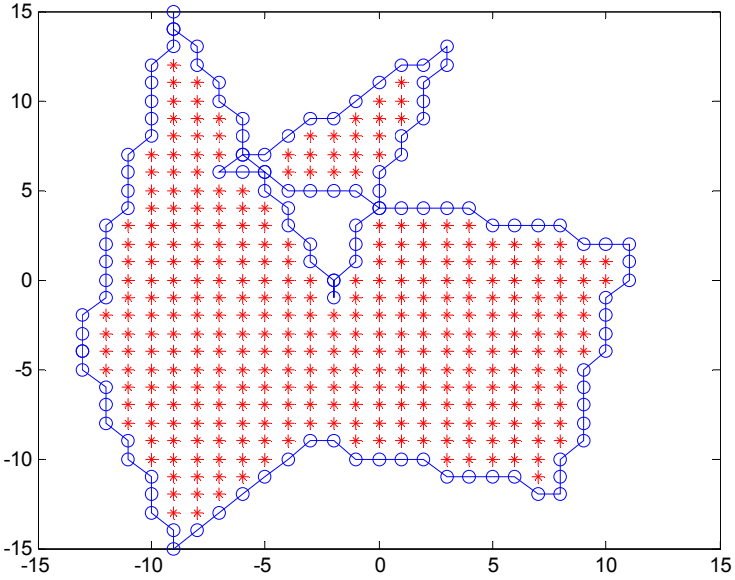
1. Ο αλγόριθμος λειτουργεί θεωρώντας μία σειρά από οριζόντιες γραμμές σάρωσης που διατρέχουν το πολύγωνο για ymin έως ymax (με βήμα 1 pixel)
2. Διατάσουμε τις γραμμές σάρωσης με φθίνουσα y
3. Για κάθε γραμμή σάρωσης y_i βρίσκουμε όλα τα σημεία του πολυγώνου με το ίδιο y_i.
4. Διατάσουμε τα σημεία τομής σε αύξουσα σειρά του x
5. Για κάθε σημείο τομής $\{x_i\}$:
   - Ελέγχουμε σε ποια κατηγορία ανήκει και ενημερώνουμε κατάλληλα την μεταβλητή parity (βλ παρακάτω)
   - Αν το parity είναι περιττό “Ζωγραφίζουμε” όλα τα Pixel από το (x_i+1,y_i) έως $(x_{i+1}-1,y_i)$

- Ολες οι πιθανές διατάξεις του τρέχοντος pixel τομής με την οριζόντια γραμμή scanline σε σχέση με το προηγούμενο και το επόμενο του πολυγώνου



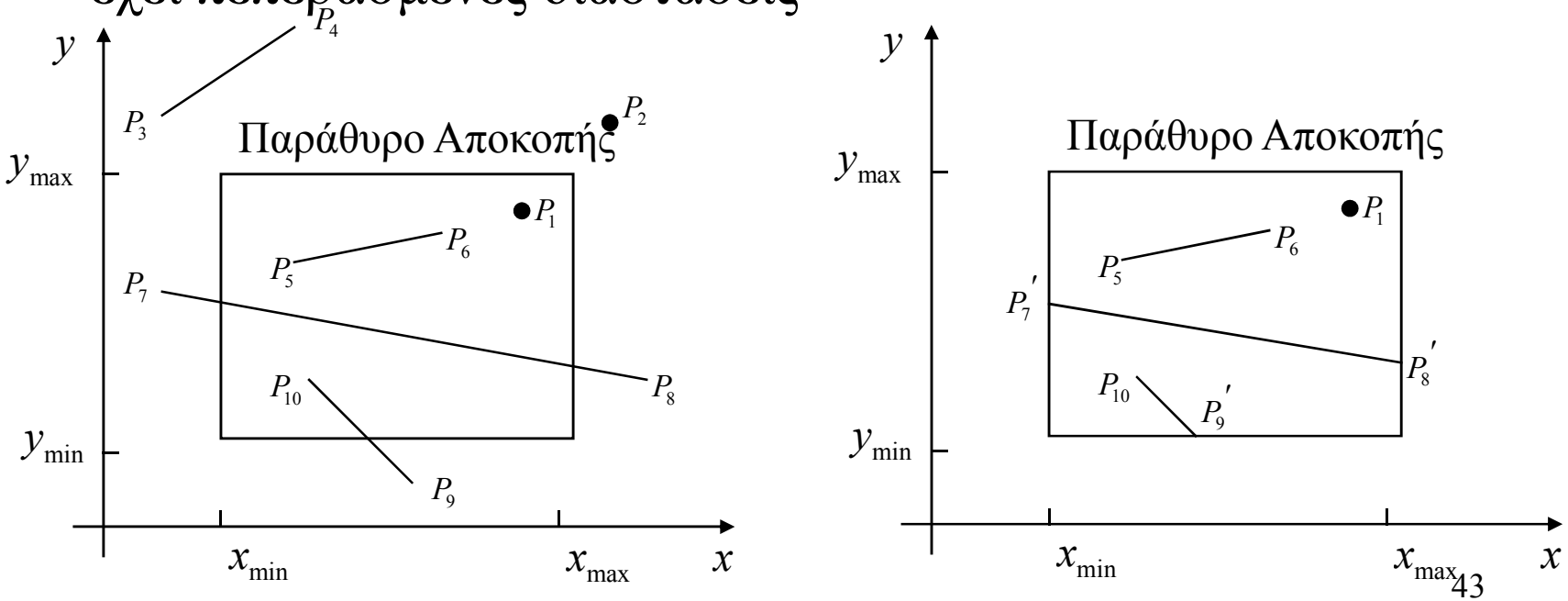
- Παράδειγμα εκτέλεσης του αλγόρίθμου για κοίλο πολύγωνο, καθώς και πολύγωνο που τέμνει τον ευατό του με σχηματισμό οπής.





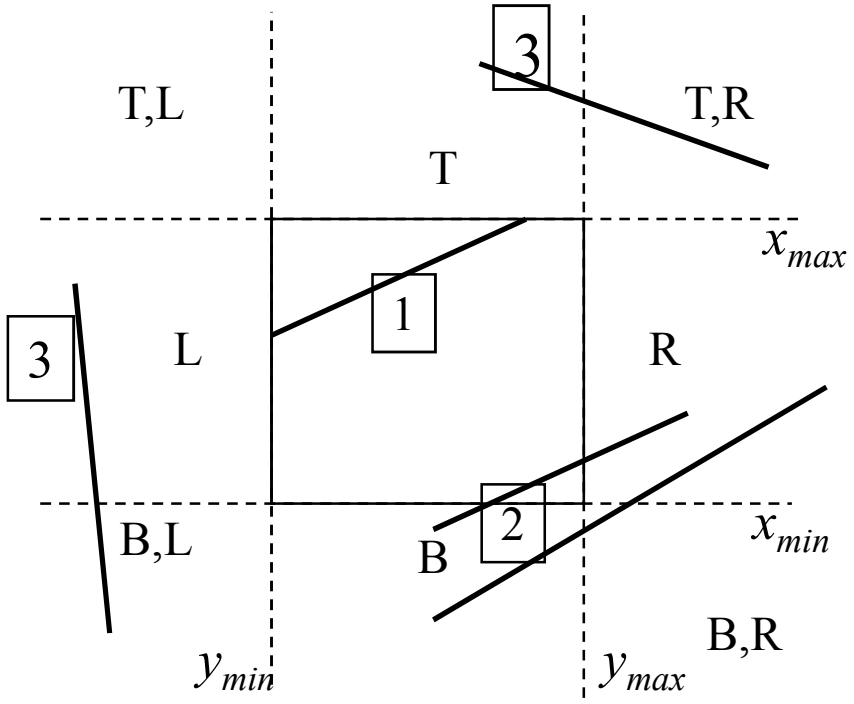
Αποκοπή (Clipping)

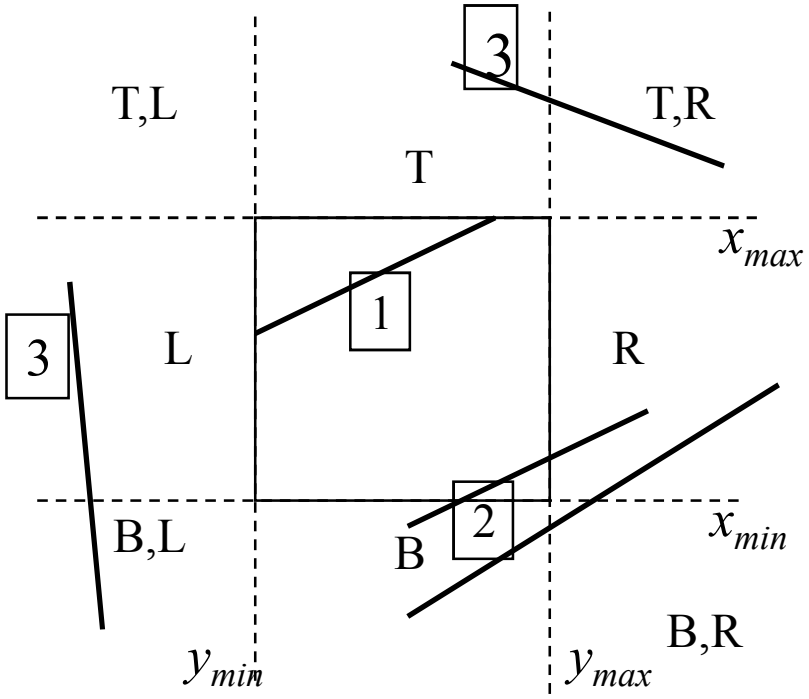
- Ορίζεται παραλληλόγραμμο παράθυρο αποκοπής (x_{min}, y_{min}, x_{max}, y_{max})
- Από οτιδήποτε έχει σχεδιαστεί, διατηρούνται μόνο τα τμήματα εντός του παραλληλογράμμου
- Εφαρμόζεται διότι η οθόνη στη οποία γίνεται η απεικόνιση έχει πεπερασμένες διαστάσεις



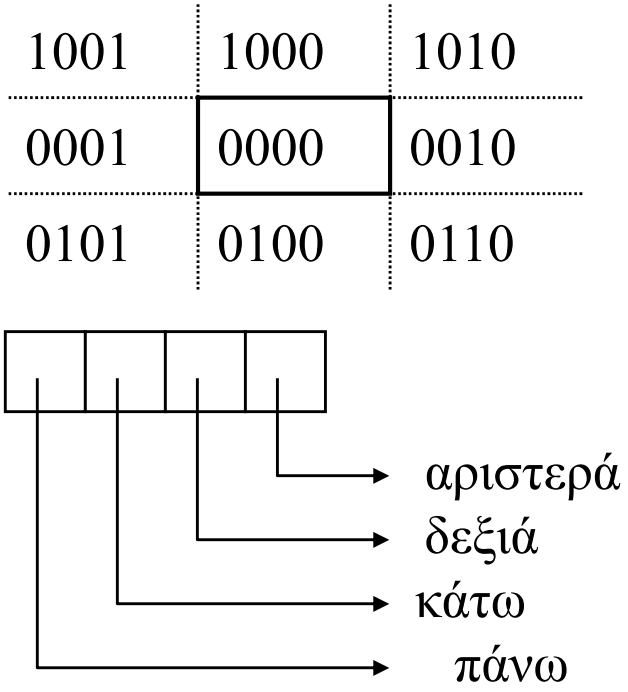
Αποκοπή (Clipping) ευθύγραμμων τμημάτων Αλγόριθμος Cohen - Sutherland

- Κωδικοποιείται το αρχικό και τελικό σημείο του ευθύγραμμου τμήματος.
- Ελέγχεται αν η περίπτωση είναι τετριμμένη
- Αν όχι, βρίσκεται η γραμμή με την οποία υπολογίζεται το σημείο τομής και, αποδίδεται κωδικός στο σημείο τομής και αντικαθιστά το άκρο που είναι εκτός .
- Ο αλγόριθμος ξανακαλείται για το νέο ευθύγραμμο τμήμα.





Σχήμα κωδικοποίησης με συμβολοσειρές



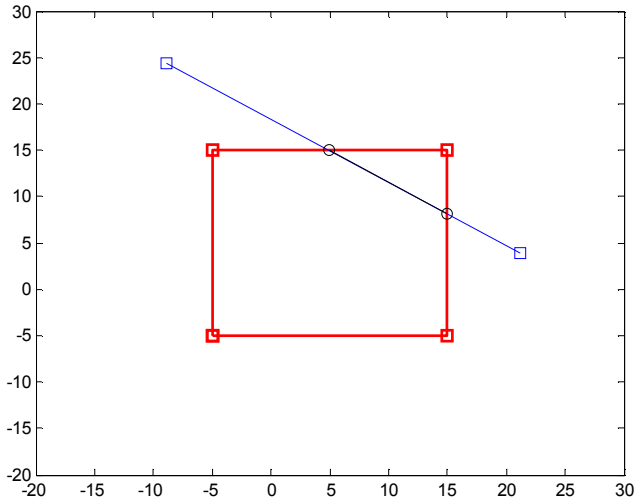
Σχήμα κωδικοποίησης με διαδικά ψηφία

Υπολογισμός του κωδικού ενός σημείου βάσει του παραλληλογράμμου αποκοπής

Οι περιπτώσεις του Αλγόριθμου Cohen - Sutherland

1. $C(P_1)=C(P_2)=$κενό→ ολόκληρο το τμήμα εντός (τετριμμένη περίπτωση)
2. $C(P_1)$ τομή $C(P_2)\neq$κενό→ ολόκληρο το τμήμα εκτός (τετριμμένη περίπτωση)
3. $C(P_1)$ =[] AND $C(P_2)\neq$[]→ υπολογισμός τομής (μη τετριμμένη περίπτωση)
4. $C(P_1)$ τομή $C(P_2)$=[]→ υπολογισμός τομής (μη τετριμμένη περίπτωση)

1. $C(P_1)=C(P_2)=$"0000" → ολόκληρο το τμήμα εντός (τετριμμένη περίπτωση)
2. $C(P_1)\&C(P_2)<>0$→ ολόκληρο το τμήμα εκτός (τετριμμένη περίπτωση)
3. $C(P_1)$ ="0000" AND $C(P_2)\neq$"0000"→ υπολογισμός τομής (μη τετριμμένη περίπτωση)
4. $C(P_1)\&C(P_2)=0$→ υπολογισμός τομής (μη τετριμμένη περίπτωση)



Παράδειγμα εφαρμογής του αλγόριθμου

Ψευδοκώδικας Cohen - Sutherland

Είσοδος: άκρα ευθύγραμμου τμήματος και συντεταγμένες παραθύρου αποκοπής

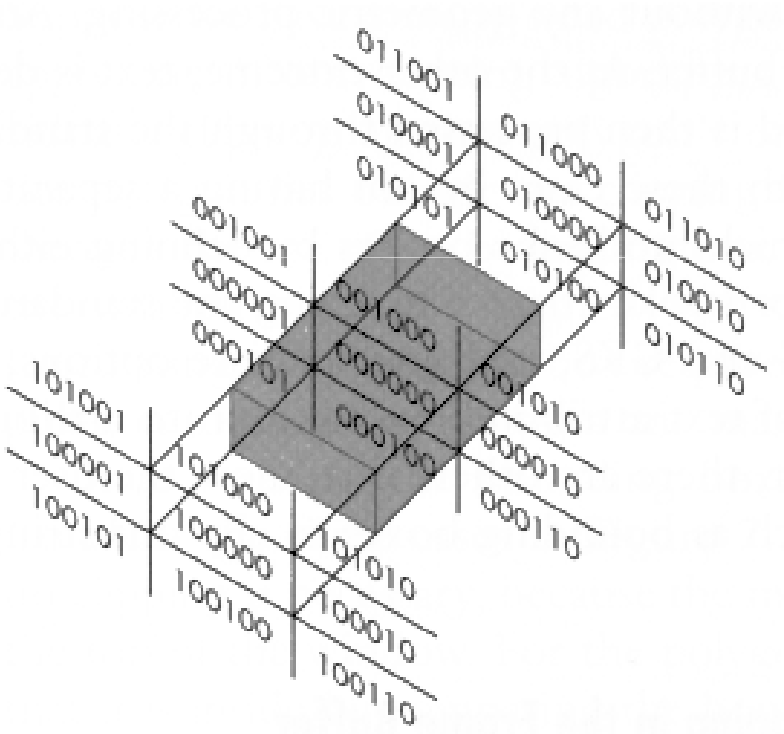
Εξοδος: άκρα αποκομένου ευθύγραμμου τμήματος,

δυαδική μεταβλητή σε περίπτωση απόριψης όλου του τμήματος

```
Υπολόγισε code1, code2 για τα άκρα του ευθ. τμ.
reject=0;
while 1
   if (code1==0 && code2==0) break;
   elseif (code1 & code2 ~=0) reject=1; break;
   else
        επιλογή ως τρέχον άκρο ενός από τα άκρα του ευθ. τμ. με current_code
        if current_code &8 >0   [x,y]=  τομή του ευθ. τμ. με y=ymax
          elseif current_code & 4>0   [x,y]= τομή του ευθ. τμ. με y=ymin
            elseif current_code & 2>0   [x,y]= τομή του ευθ. τμ. με x=xmax
                elseif bitand(current_code,1)>0   [x,y]= τομή του ευθ. τμ. με x=xmin
        end
   end
   Αντικατάσταση του τρεχόντος άκρου από την τομή, υπολογισμός νέου code
end
```

Αποκοπή (Clipping) ευθύγραμμων τμημάτων 3D Αλγόριθμος Cohen - Sutherland

- Ορίζεται παραλληλόγραμμο παράθυρο αποκοπής (x_{min}, y_{min}, x_{max}, y_{max})
- Από οτιδήποτε έχει σχεδιαστεί, διατηρούνται μόνο τα τμήματα εντός του παραλληλογράμμου
- Κωδικοποίηση αρχικού και τελικού σημείου με 6 bits.
- Επίλυση του προβλήματος τομής ευθείας που ορίζεται από 2 σημεία 3D με επίπεδο



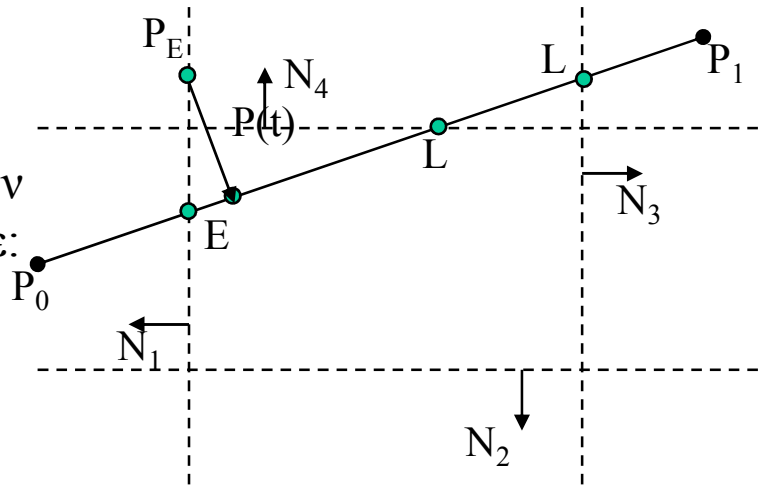
$$x = x_1 + t\left(x_2 - x_1\right)$$

$$y = y_1 + t\left(y_2 - y_1\right)$$

$$z = z_1 + t\left(z_2 - z_1\right)$$

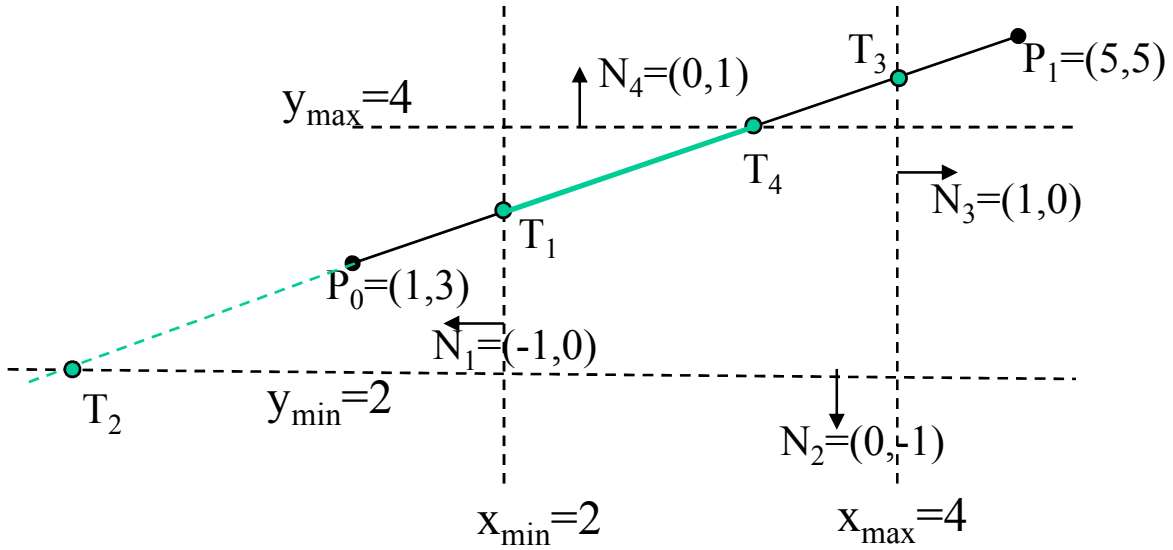
Αποκοπή (Clipping) γραμμών με κυρτό πολύγωνο: Αλγόριθμος Liang - Bersky

- Εκφράζουμε το P_0P_1 παραμετρικά (t)
- Για κάθε ευθύγραμμο τμήμα:
  - Υπολογίζουμε τα κάθετα διανύματα με κατεύθυνση προς τα έξω: $\mathbf{N_i}=(-(y_i - y_{i+1}), (x_i-x_{i+1}))$
  - Επιλέγουμε ένα σημείο P_E επί του ευθύγραμου τμήματος και υπολογίζουμε την τιμή της t για το σημείο τομής, σύμφωνα με: $P_EP(t).N_1=0$
  - Ταξινομούμε το σημείο τομής σε είσοδο E και έξοδο L, βάσει του πρόσημου $N_1.P_0P_1$
    - $\mathbf{N_i}.P_0P_1 >0 \rightarrow$ ENTER
    - $\mathbf{N_i}.P_0P_1 <0 \rightarrow$ LEAVE



- Βρίσκουμε τα σημεία τομής βάσει της συνθήκης $t_E<t_L$

- Ταξινομούμε τα σημεία τομής σε είσοδο E και έξοδο L, βάσει του πρόσημου $N_1.P_0P_1$
- Βρίσκουμε τα σημεία τομής βάσει της συνθήκης $t_E<t_L$
- Είσοδος: $t_{start}=\max(t_1, t_2,0)=\max(1/4, -1/2,0)=1/4$
- Εξοδος: $t_{start}=\min(t_3, t_4,1)=\min(1/2, 3/4,1)=1/2$
- Αν $\max(t_E)>=\min(t_L)$ τότε δεν υπάρχει τομή.



```
For i:=1 to clipping_edges
   calculate Ni;
   Select random Point PP
Για κάθε ευθ. Τμήμα
   tE:=0; tL:=1;
   PE:=0; PL:=0;
   Για κάθε πιθανή τομή i
        if Ni.D<>0 →
               calculate t;
               if Ni.D<0→PE:=1 else PL=1;
               if (PE)→tE=max(tE,t);
               if (PL)→tL=min(tL,t);
        if (tE>tL)→ break else
               PE:=P(tE);
               PL:=P(tL);
```

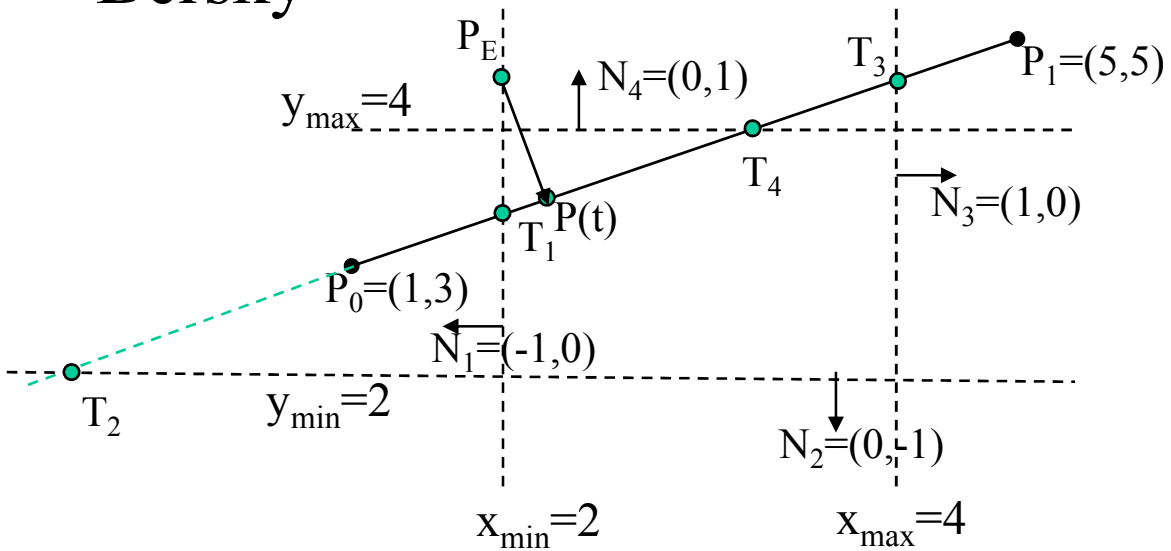
Παράδειγμα εφαρμογής του Αλγόριθμου Liang - Bersky

Τομή με $x=x_{min}=2$.

Επιλέγουμε $P_E=(2,0)$

$$t_1=-\frac{(P_0-P_E)\mathbf{N_1}}{(P_1-P_0)\mathbf{N_1}}=-\frac{(-1,3)\cdot(-1,0)}{(4,2)\cdot(-1,0)}=\frac{1}{4}$$

$$(P_1-P_0)\mathbf{N_1}=(4,2)\cdot(-1,0)<0\Rightarrow ENTER$$



Τομή με $y=y_{min}=2$.

Επιλέγουμε $P_E=(0,2)$

$$t_2=-\frac{(P_0-P_E)\mathbf{N}_2}{(P_1-P_0)\mathbf{N}_2}=-\frac{(1,1)\cdot(0,-1)}{(4,2)\cdot(0,-1)}=-\frac{1}{2}$$

$$(P_1-P_0)\mathbf{N}_2=(4,2)\cdot(0,-1)<0\Rightarrow ENTER$$

Τομή με $x=x_{max}=4$.

Επιλέγουμε $P_E=(4,0)$

$$t_3=-\frac{(P_0-P_E)\mathbf{N}_3}{(P_1-P_0)\mathbf{N}_3}=-\frac{(-3,0)\cdot(1,0)}{(4,2)\cdot(1,0)}=\frac{3}{4}$$

$$(P_1-P_0)\mathbf{N}_2=(4,2)\cdot(1,0)>0\Rightarrow LEAVE$$

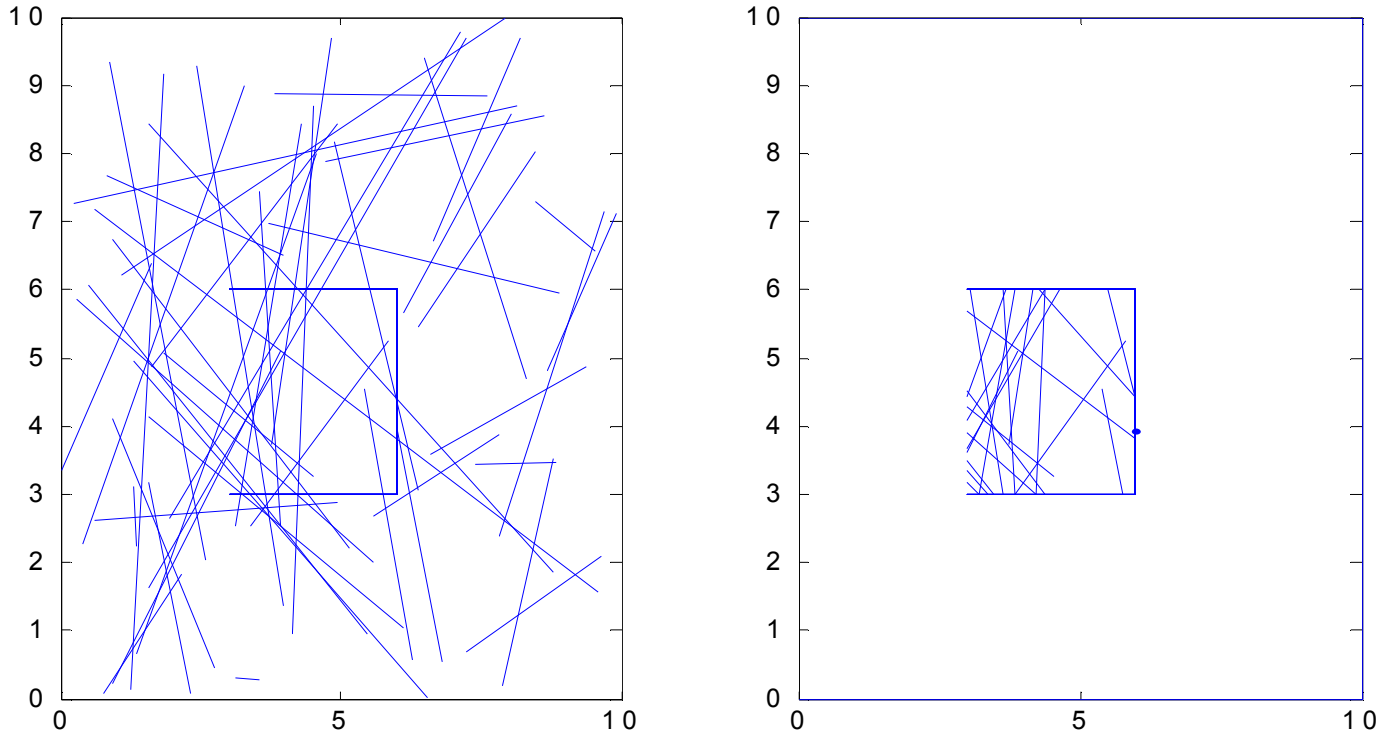
Τομή με $y=y_{max}=4$.

Επιλέγουμε $P_E=(0,4)$

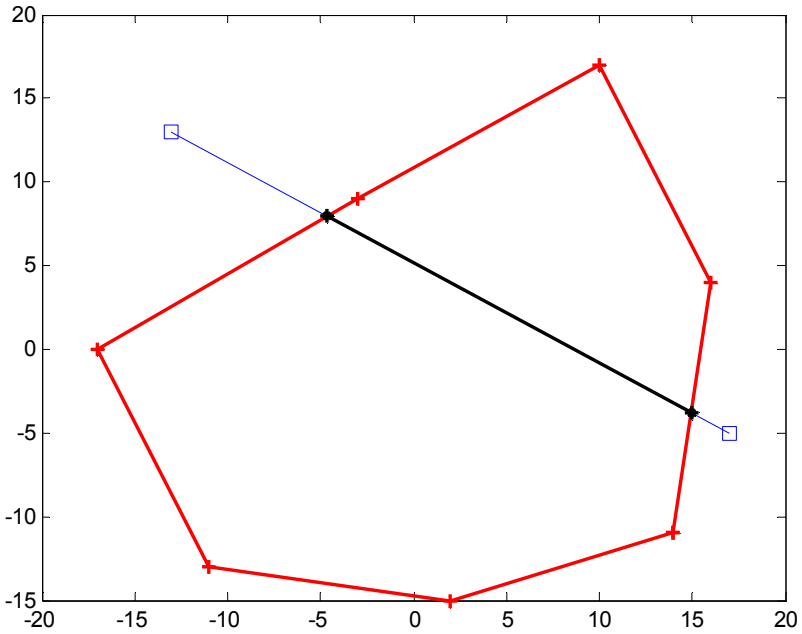
$$t_4=-\frac{(P_0-P_E)\mathbf{N}_2}{(P_1-P_0)\mathbf{N}_2}=-\frac{(1,-1)\cdot(0,1)}{(4,2)\cdot(0,1)}=\frac{1}{2}$$

$$(P_1-P_0)\mathbf{N}_2=(4,2)\cdot(0,1)>0\Rightarrow LEAVE$$

Παράδειγμα εφαρμογής του Liang – Bersky: clipping random lines



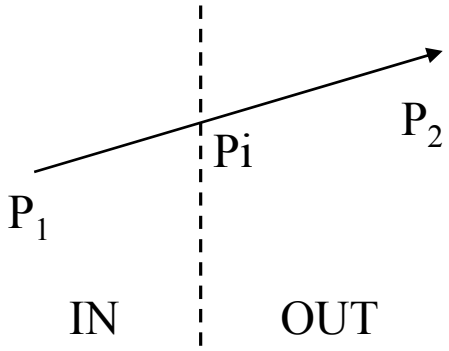
- Ο αλγόριθμος μπορεί να εφαρμοστεί χωρίς αλλαγές σε οποιοδήποτε κυτρό πολύγωνο.



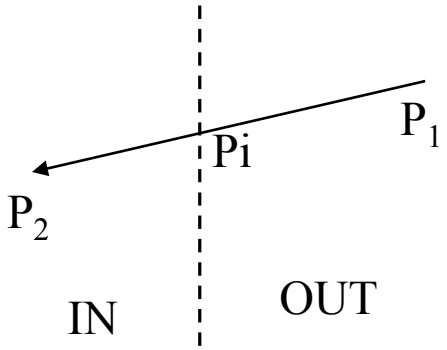
Αποκοπή (Clipping) πολυγώνων: Αλγόριθμος Cohen - Sutherland

- Δεδομένου ενός τυχαίου πολυγώνου (clipped), κυρτού ή κοίλου, να βρεθεί η τομή του με ένα τυχαίο κυρτό πολύγωνο (clipping polygon)

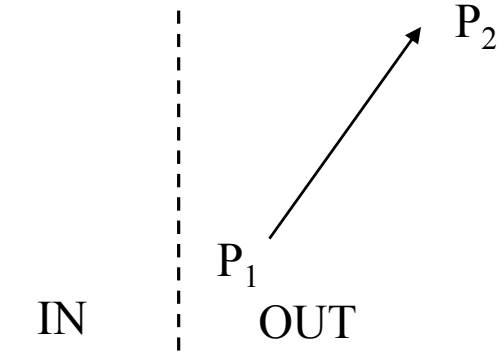
- Ο αλγόριθμος ελέγχει για κάθε τμήμα P_1P_2, ποια από τις περιπτώσεις ισχύει:



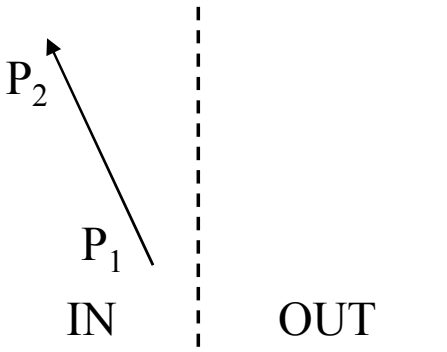
Αποθήκευση τελικού σημείου



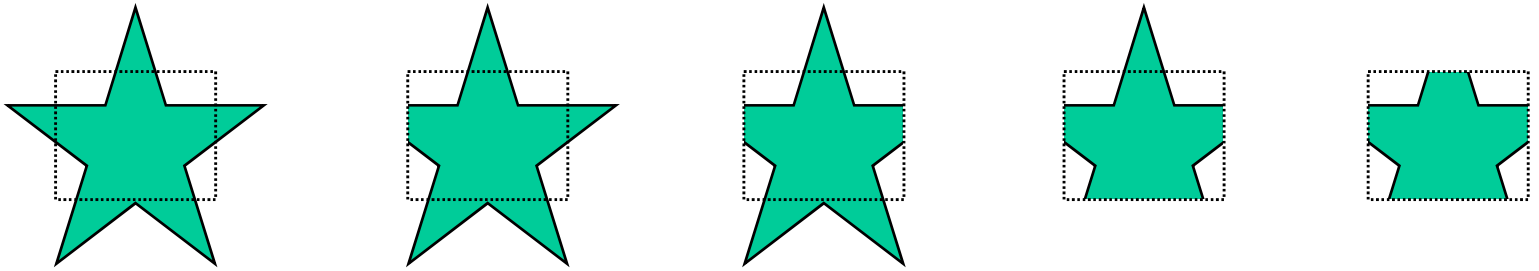
Αποθήκευση τελικού σημείου και τομής



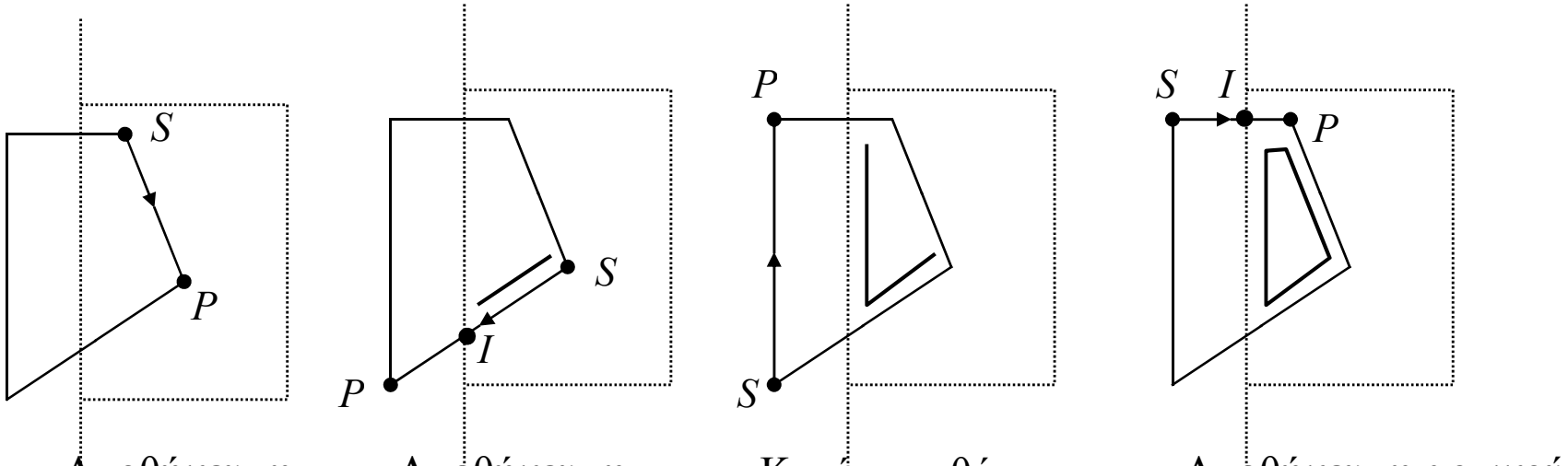
Καμία αποθήκευση σημείου



Αποθήκευση τελικού σημείου



Original Polygon | Clip Left | Clip Right | Clip Bottom | Clip Top



Αποθήκευση τελικού σημείου (a) | Αποθήκευση τομής (b) | Καμία αποθήκευση σημείου (c) | Αποθήκευση αρχικού σημείου και τομής (d)

Ψευδοκώδικας Αλγορίθμου Sutherland - Hodgman

Είσοδος: initial_polygon P, μήκους N
clipboundary C, μήκους K
Εξοδος: newP πολύγωνο τομής

Συναρτήσεις που χρησιμοποιούνται:

σημείο τομής 2 ευθειών που καθορίζονται από 2 ζεύγη ευθύγραμμων τμ.

Προσοχή, δεν απαιτείται η τομή να είναι εντός των ευθ. Τμημ.

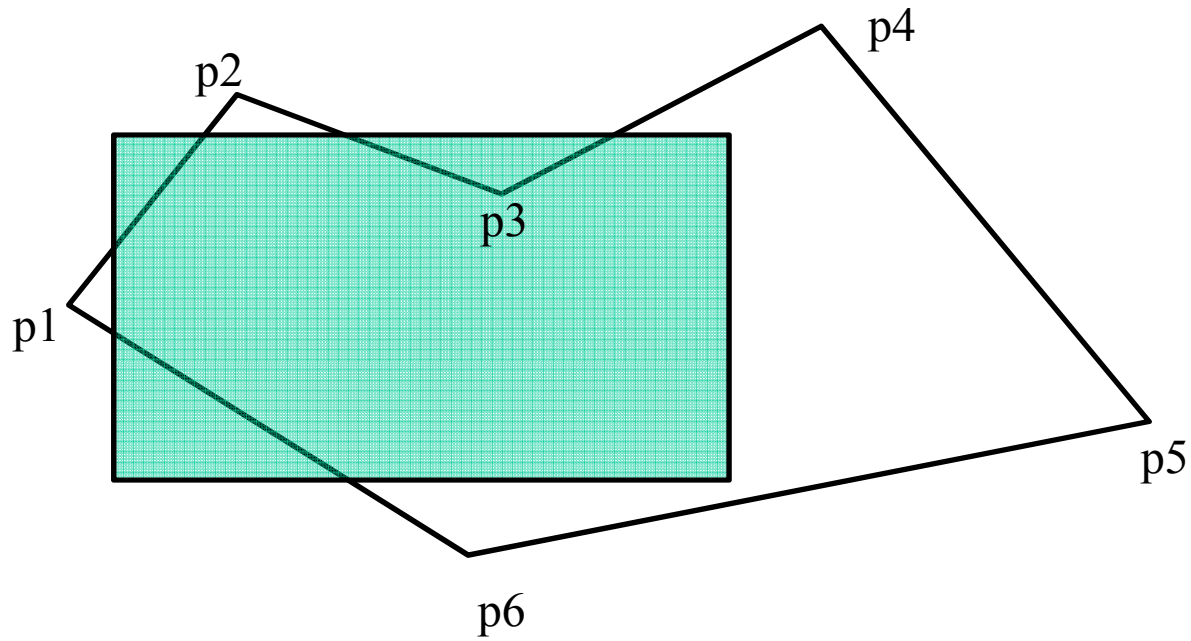
inP(C_kC_{k+1}, Pi): true αν το Pi βρίσκεται στο ημιεπίπεδο της C_kC_{k+1} που είναι εσωτερικό ως προς το πολύγωνο P.

```
poly_clip(initial_polygon, clipboundary)
   for k=1:K
      for i=1:N
         Pi: τρέχον σημείο
         Pi+1: επόμενο σημείο
         if inP(Pi)
            if inP(Pi+1) insert(newP,Pi+1)
               else
                  T=σημείο τομής του PiPi+1 με το CBkCBk+1 .
                  insert(newP,T)
         if not(inP(Pi))
            if inP(Pi+1)
               T=σημείο τομής του PiPi+1 με το CBkCBk+1 .
               insert(newP,T)
               insert(newP,Pi+1)
      P=newP
```

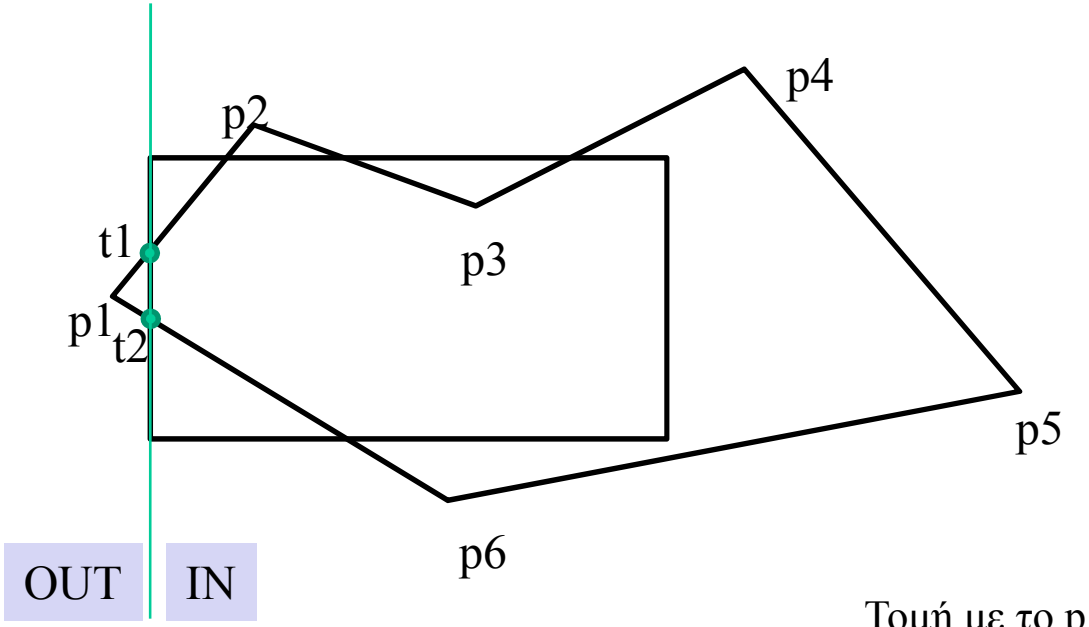
Ψευδοκώδικας Αλγορίθμου Sutherland - Hodgman

```
{ Cohen_Sutherland(initial_polygon, clipboundary)
For j=1 to N
   if i>1 →P1=initial_polygon(i-1) else
        P1=initial_polygon(N);
   P2=initial_polygon(i);
   if IN(P2) →
        IF IN(P1) →
                put(p2,vertexarray);
        else
                i:=intersect(P1,P2, clipboundary);
                put(p2,vertexarray);
                put(i,vertexarray);
   else
        if IN(P1) →
                i:=intersect(P1,P2, clipboundary);
                put(i,vertexarray);
Return(vertexarray);}
// CALL Cohen_Sutherland
for i=1 to k
   t=Cohen_Sutherland(initial_polygon, clipboundary(i));
   initial_polygon=t;
}
```

Παράδειγμα εφαρμογής του Αλγορίθμου Cohen - Sutherland

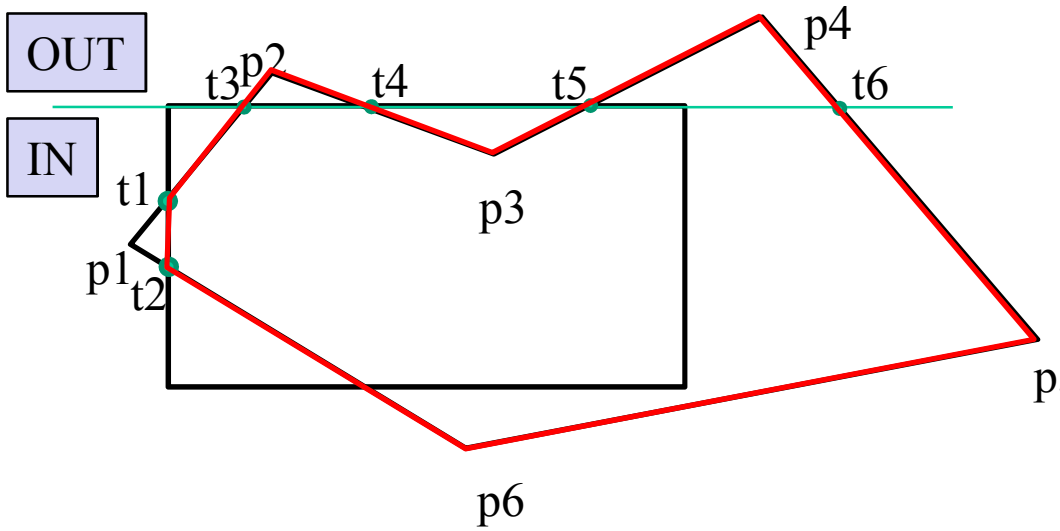


- Εστω αποκόπτον παραλληλόγραμμο (clipping polygon) και τυχαίο πολύγωνο προς αποκοπή (p1p2p3p4p5p6). Να γίνει εφαρμογή του αλγόριθμου Cohen – Sutherland.



Αρχικό πολύγωνο
[p1,p2,p3,p4,p5,p6]

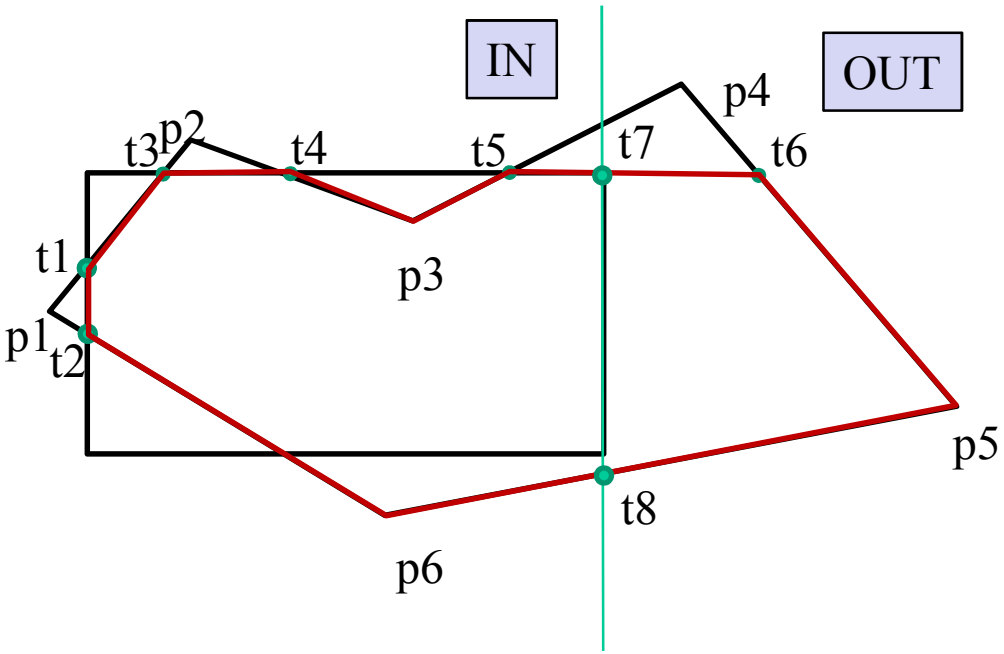
Τομή με το p1p2	[t1,p2]
Τομή με το p2p3	[t1,p2,p3]
Τομή με το p3p4	[t1,p2,p3,p4]
Τομή με το p4p5	[t1,p2,p3,p4,p5]
Τομή με το p5p6	[t1,p2,p3,p4,p5,p6]
Τομή με το p6p1	[t1,p2,p3,p4,p5,p6,t2]



Πολύγωνο από τομή με την ευθεία 1

[t1,p2,p3,p4,p5,p6,t2]

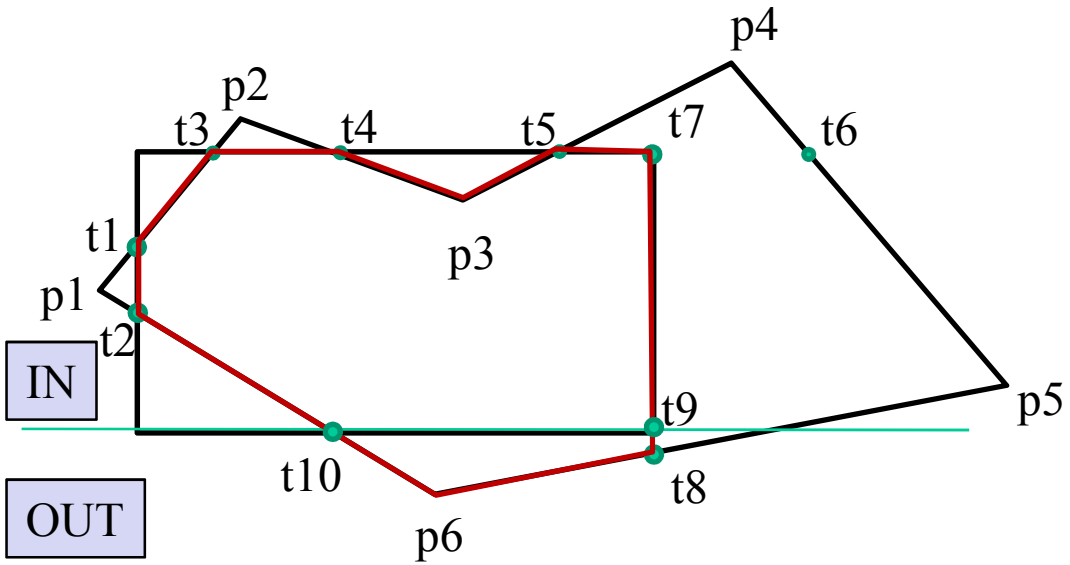
Τομή με το t1p2	[t1,t3]
Τομή με το p2p3	[t1,t3,t4,p3]
Τομή με το p3p4	[t1,t3,t4,p3,t5]
Τομή με το p4p5	[t1,t3,t4,p3,t5,t6,p5]
Τομή με το p5p6	[t1,t3,t4,p3,t5,t6,p5,p6]
Τομή με το p6t2	[t1,t3,t4,p3,t5,t6,p5,p6,t2]



Πολύγωνο από τομή με την ευθεία 2

[t1,t3,t4,p3,t5,t6,p5,p6,t2]

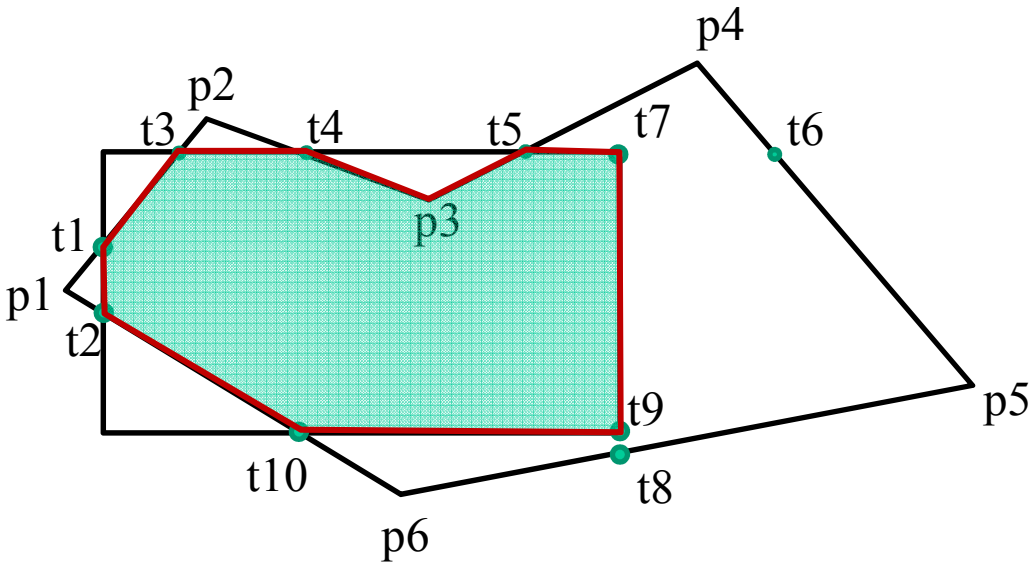
Τομή με το t1t3	[t1,t3]
Τομή με το t3t4	[t1,t3,t4]
Τομή με το t4p3	[t1,t3,t4,p3]
Τομή με το p3t5	[t1,t3,t4,p3,t5]
Τομή με το t5t6	[t1,t3,t4,p3,t5,t7]
Τομή με το t6p5	[t1,t3,t4,p3,t5,t7]
Τομή με το p5p6	[t1,t3,t4,p3,t5,t7,t8,p6]
Τομή με το p6t2	[t1,t3,t4,p3,t5,t7,t8,p6,t2]



Πολύγωνο από τομή με την ευθεία 3

[t1,t3,t4,p3,t5,t7,t8,p6,t2]

Τομή με το t1t3	[t1,t3]
Τομή με το t3t4	[t1,t3,t4]
Τομή με το t4p3	[t1,t3,t4,p3]
Τομή με το p3t5	[t1,t3,t4,p3,t5]
Τομή με το t5t7	[t1,t3,t4,p3,t5,t7]
Τομή με το t7t8	[t1,t3,t4,p3,t5,t7,t9]
Τομή με το t8p6	[t1,t3,t4,p3,t5,t7,t9]
Τομή με το p6t2	[t1,t3,t4,p3,t5,t7,t9,t10,t2]



Τελικό Πολύγωνο [t1,t3,t4,p3,t5,t7,t9,t10,t2]

Παραδείγματα εφαρμογής Αποκοπής πολυγώνων