2ο Μάθημα – Αλγόριθμοι Σχεδίασης

Γραφικα

Ευάγγελος Σπύρου

Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Ακ. Έτος 2018-19

Σύνοψη του σημερινού μαθήματος

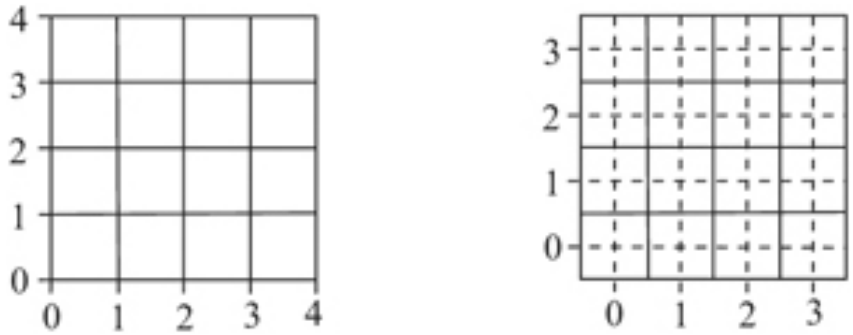
Εισαγωγή

- Οι **διδιάστατες** συσκευές απεικόνισης (π.χ. οθόνες) αποτελούνται από ένα **διακριτό** πλέγμα εικονοστοιχείων
- Καθένα από αυτά μπορεί να λάβει μια **χρωματική** τιμή
- Η **σχεδίαση** (rasterization) είναι η διαδικασία της μετατροπής διδιάστατων στοιχειωδών σχημάτων σε μια **διακριτή** παράσταση
- Με άλλα λόγια, πρέπει να βρεθούν τα **εικονοστοιχεία** που απαρτίζουν ένα στοιχειώδες σχήμα

Εισαγωγή

- Δεδομένου ότι θέλουμε να σχεδιάσουμε P στοιχειώδη σχήματα, για ένα συγκεκριμένο καρέ και υποθέτοντας ότι κάθε στοιχειώδες σχήμα αποτελείται από p εικονοστοιχεία, η **πολυπλοκότητα** της σχεδίασης είναι γενικά $O(Pp)$
- Άλλα (προηγούμενα) στάδια της σωλήνωσης γραφικών (μετασχηματισμοί και περικοπή) χρησιμοποιούν μόνο τις **κορυφές** των στοιχειωδών σχημάτων
- Γενικά, η πολυπλοκότητα αυτών των προηγούμενων σταδίων είναι $O(Pn)$, όπου n το μέσο πλήθος κορυφών από ένα **στοιχειώδες** σχήμα
- Συνήθως, $p \gg n$, οπότε πρέπει να εξασφαλιστεί ότι οι αλγόριθμοι **σχεδίασης** είναι ιδιαίτερα **αποδοτικοί**

Εισαγωγή

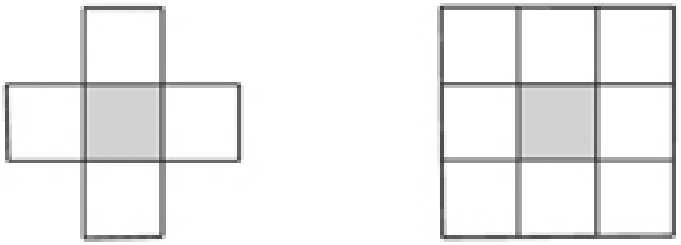


- Τα εικονοστοιχεία (pixels) μιας πλεγματικής οθόνης αποτελούν ένα διδιάστατο **κανονικό πλέγμα**
- Υπάρχουν δύο τρόποι **θεώρησης** του πλέγματος αυτού
  - **Κέντρα σε ημίσειες συντεταγμένες** (δεξιά): φανταστικές κάθετες και οριζόντιες γραμμές
  - **Κέντρα σε ακέραιες συντεταγμένες** (αριστερά): πραγματικό πλέγμα

Εισαγωγή

- Θα θεωρήσουμε τα κέντρα σε **ακέραιες** συντεταγμένες
- Όταν αναφερόμαστε σε ένα εικονοστοιχείο σαν σημείο, θα εννοούμε το **κέντρο** του
- Μια σημαντική έννοια της σχεδίασης είναι αυτή της **σύνδεσης**: Πότε είναι συνδεδεμένο ένα σχήμα που αποτελείται από **εικονοστοιχεία**;
- Για παράδειγμα, αν ένας αλγόριθμος σχεδίασης μιας καμπύλης **μεταβεί** από ένα εικονοστοιχείο στο **διαγώνιο** γείτονά του, είναι **συνδεδεμένη** η καμπύλη, ή υπάρχει κενό;

Εισαγωγή



- Υπάρχουν δύο τύποι σύνδεσης, η **τετραπλή** (αριστερά) και η **οκταπλή** (δεξιά)
- Όποιο τύπο και αν επιλέξουμε, θα πρέπει να βεβαιωθούμε ότι τον **υποστηρίζουν** οι αλγόριθμοί μας
- Εδώ θα χρησιμοποιούμε **οκταπλή** σύνδεση

Εισαγωγή

- Η δημιουργία αλγορίθμων σχεδίασης για μια κλάση **στοιχειωδών** σχημάτων έχει δύο βασικούς **στόχους**:
  01. **επιλογή** των εικονοστοιχείων που παριστάνουν ένα σχήμα με ακρίβεια
  02. **αποδοτικότητα**
- Ο πρώτος στόχος είναι απαραίτητος για τη σωστή **σύνδεση** και συνεπάγεται ότι ένας αλγόριθμος σχεδίασης:
  01. επιλέγει τα εικονοστοιχεία που παριστάνουν **καλύτερα** το σχήμα
  02. μεταβάλλει **μόνο** αυτά τα εικονοστοιχεία
  03. μεταβάλλει τις **χρωματικές** τιμές αυτών των εικονοστοιχείων **σωστά**

Εισαγωγή

- Ο δεύτερος στόχος είναι ιδιαίτερα σημαντικός, μιας και μια συνθετική σκηνή μπορεί να περιλαμβάνει πολύ **μεγάλο** αριθμό από στοιχειώδη σχήματα
  - υπάρχει συχνά απαίτηση για λειτουργία σε **πραγματικό** χρόνο
- Στο σημερινό μάθημα θα ασχοληθούμε με τις μαθηματικές αρχές και τους αλγορίθμους για τη σχεδίαση **κοινών** στοιχειωδών σχημάτων
- Ευθύγραμμα **τμήματα**, **κύκλοι**, γενικά **πολύγωνα**, **τρίγωνα** και **κλειστές** περιοχές
- Θα δούμε επίσης τις τεχνικές της προοπτικής **διόρθωσης** και της **αντιταύτισης** που σκοπό έχουν να βελτιώσουν την ποιότητα της σχεδίασης

Πεπλεγμένες μορφές

- Μεταξύ των μαθηματικών ορισμών 2Δ στοιχειωδών καμπυλών, ο **πεπλεγμένος** και ο **παραμετρικός** ορισμός είναι οι πιο χρήσιμοι στη σχεδίαση
- Μια **πεπλεγμένη** καμπύλη ορίζεται σαν μια συνάρτηση $f(x, y)$ που παράγει **τρία** πιθανά αποτελέσματα:

$$f(x,y) = \begin{cases} < 0, & \text{το σημείο } (x,y) \text{είναι εντός της καμπύλης,} \\ = 0, & \text{το σημείο } (x,y) \text{είναι πάνω στην καμπύλη,} \\ > 0, & \text{το σημείο } (x,y) \text{είναι εκτός της καμπύλης.} \end{cases}$$

- Οι όροι **εντός** και **εκτός** δεν έχουν ειδική σημασία και σε μερικές περιπτώσεις π.χ., ευθεία είναι εντελώς **συμμετρικοί**
- Έτσι, μια καμπύλη διαιρεί το επίπεδο σε δύο **μη επικαλυπτόμενες** περιοχές, εσωτερική και εξωτερική

Πεπλεγμένες μορφές – ευθεία

- Πεπλεγμένη μορφή **ευθείας**:

$$l(x, y) \equiv ax + by + c = 0$$

  - a, b, c οι **συντελεστές** της ευθείας
  - τα σημεία (x, y) **πάνω** στην ευθεία έχουν $l(x, y) = 0$
  - για ευθεία από το $\mathbf{p_1} = (x_1, y_1)$ στο $\mathbf{p_2} = (x_2, y_2)$ έχουμε $a = y_2 - y_1, b = x_1 - x_2, c = x_2 y_1 - x_1 y_2$

- Η ευθεία **διαιρεί** το επίπεδο σε δύο ημιεπίπεδα ανάλογα με την τιμή της $l(x, y)$

Πεπλεγμένες μορφές – κύκλος

- Πεπλεγμένη μορφή **κύκλου** με κέντρο $\mathbf{c} = (x_c, y_c)$, ακτίνα r:

$$c(x, y) \equiv (x - x_c)^2 + (y - y_c)^2 - r^2 = 0$$

- Ο κύκλος **διαιρεί** το επίπεδο σε δύο ημιεπίπεδα ανάλογα με την τιμή της $c(x, y)$

Παραμετρικές μορφές – ευθεία

- Η **παραμετρική** μορφή ορίζει την καμπύλη σαν συνάρτηση μιας **παραμέτρου** t, η οποία προσεγγίζει το μήκος τόξου που έχει **διανυθεί** επί της καμπύλης
- Για **ευθεία** από το $\mathbf{p_1} = (x_1, y_1)$ στο $\mathbf{p_2} = (x_2, y_2)$ θα είναι

$$\mathbf{l}(t) = (x(t), y(t))$$

όπου $x(t) = x_1 + t(x_2 - x_1), y(t) = y_1 + t(y_2 - y_1)$

- Καθώς το t παίρνει τιμές από το 0 εώς το 1, σχηματίζεται το **ευθύγραμμο τμήμα** από το $\mathbf{p_1}$ στο $\mathbf{p_2}$
- Προεκτείνοντας τις τιμές του t **εκτός** του διαστήματος αυτού, σχηματίζεται η **ευθεία** που ορίζεται από τα $\mathbf{p_1}, \mathbf{p_2}$

Παραμετρικές μορφές - κύκλος

- Παρομοίως, ο παραμετρικός ορισμός **κύκλου** με κέντρο (x_c, y_c) και ακτίνα r είναι:

$$\mathbf{c}(t) = (x(t), y(t))$$

όπου $x(t) = x_c + r\cos(2\pi t), y(t) = y_c + r\sin(2\pi t)$

- Καθώς το t παίρνει τιμές από το 0 στο 1, **διαγράφεται** ο κύκλος
- Αν προεκταθούν οι τιμές του t **πέραν** των ορίων, ο κύκλος **επαναδιαγράφεται**

Σχεδιασμός στοιχειωδών σχημάτων

- Οι συναρτήσεις που ορίζουν στοιχειώδη σχήματα, συχνά χρειάζεται να πάρουν τιμές πάνω στο **πλέγμα** των εικονοστοιχείων
- Ο υπολογισμός της τιμής τους για κάθε εικονοστοιχείο ανεξάρτητα **δεν** είναι αποδοτικός
  - π.χ., για την τιμή πεπλεγμένης συνάρτησης **ευθείας** χρειάζονται **2** πολλαπλασιασμοί και **2** προσθέσεις ανά εικονοστοιχείο
  - αντίστοιχα, για **κύκλο** χρειάζονται **3** πολλαπλασιασμοί και **4** προσθέσεις ανά εικονοστοιχείο

Σχεδιασμός στοιχειωδών σχημάτων

- Λόγω της **κανονικότητας** του πλέγματος, δίνεται η δυνατότητα για **μείωση** του κόστους, αν εκμεταλλευτεί κανείς τις **πεπερασμένες** διαφορές των συναρτήσεων
  - π.χ., η **πρώτη** έμπροσθεν διαφορά μιας συνάρτησης f στο x_i ορίζεται ως $\delta f_i = f_{i+1} - f_i$ με $f_i = f(x_i)$
  - παρομοίως και **αναδρομικά** ορίζονται και οι υπόλοιπες διαφορές, π.χ., η δεύτερη:

$$\delta^2 f_i = \delta f_{i+1} - \delta f_i$$

και η k-οστή

$$\delta^k f_i = \delta^{k-1} f_{i+1} - \delta^{k-1} f_i$$

- Για μια **πολυωνυμική** συνάρτηση βαθμού n, **όλες** οι διαφορές από την n-οστή και πάνω είναι **σταθερές**, όλες οι διαφορές από την $n+1$-οστή και πάνω είναι 0

Σχεδιασμός στοιχειωδών σχημάτων

- Θεωρώντας την πεπλεγμένη εξίσωση της ευθείας, υπολογίζουμε τις έμπροσθεν διαφορές της για ένα **μοναδιαίο βήμα** στο x
  - δηλαδή από το εικονοστοιχείο x στο $x + 1$
- Αφού η εξίσωση είναι **πρώτου** βαθμού στο x, χρειάζεται να υπολογίσουμε μόνο την **πρώτη** (σταθερή) έμπροσθεν διαφορά της κατά x:

$$\delta_x l(x, y) = l(x + 1, y) - l(x, y) = \alpha$$

όπου δ_x η έμπροσθεν διαφορά κατά την παράμετρο x

- **Παρομοίως** $\delta_y l(x, y) = b$
- Έτσι οι τιμές της συνάρτησης της ευθείας μπορούν να υπολογισθούν **αυξητικά** από εικονοστοιχείο σε εικονοστοιχείο

Σχεδιασμός στοιχειωδών σχημάτων

- Για να βρούμε την τιμή της στο εικονοστοιχείο $(x+1, y)$ από την τιμή της $l(x, y)$ στο (x, y), απλά **υπολογίζουμε** την $l(x, y) + \delta_x l(x, y) = l(x, y) + \alpha$
- Για να πάμε από το εικονοστοιχείο (x, y) στο $(x, y+1)$, απλά **υπολογίζουμε** την $l(x, y) + \delta_y l(x, y) = l(x, y) + b$
- Κάθε **αυξητικός** υπολογισμός τιμής της συνάρτησης ευθείας κοστίζει μόνο μια **πρόσθεση**
- Με **αντίστοιχο** τρόπο μπορούμε να δουλέψουμε και για τον **κύκλο**

Σχεδιασμός στοιχειωδών σχημάτων

- Για να **σχεδιάσουμε** ένα στοιχειώδες τμήμα, πρέπει να βρούμε τα εικονοστοιχεία που το **απαρτίζουν**
- Ένας τρόπος για να το επιτύχουμε αυτό είναι να ορίσουμε μια **λογική** συνάρτηση η οποία να αποφαίνεται εάν ένα εικονοστοιχείο (x, y) ανήκει στο στοιχειώδες σχήμα, ή όχι
- Για το σκοπό αυτό μπορεί να χρησιμοποιηθεί μια **πεπλεγμένη** συνάρτηση
- Π.χ., για την **απόσταση** του (x, y) από την ευθεία που περιγράφεται από την πεπλεγμένη συνάρτηση, έχουμε:

$$\frac{|l(x, y)|}{\sqrt{a^2 + b^2}}$$

Σχεδιασμός στοιχειωδών σχημάτων

- Ένας **έλεγχος** για την υπαγωγή του (x, y) στη σχεδίαση της ευθείας θα μπορούσε να είναι

$$|l(x, y)| < e$$

όπου το e σχετίζεται με το απαιτούμενο **πάχος** της ευθείας

- Δυστυχώς, ο **τυφλός** υπολογισμός τέτοιων συναρτήσεων πάνω στο πλέγμα των εικονοστοιχείων είναι **χρονοβόρος**, ακόμη κι αν γίνει αυξητικά, χρησιμοποιώντας τις πεπερασμένες διαφορές των συναρτήσεων
- Αντίθετα, υπάρχουν μέθοδοι που "**παρακολουθούν**" ένα στοιχειώδες σχήμα και είναι γενικά πιο **αποδοτικές**

Σχεδίαση Ευθείας

- Για να δημιουργήσουμε έναν καλό αλγόριθμο σχεδίασης ευθείας πρέπει πρώτα να καθορίσουμε ένα κριτήριο **ορθότητας**
- Γενικά δεν είναι δυνατό να επιλέξουμε εικονοστοιχεία ακριβώς **πάνω** στη μαθηματική ευθεία, λόγω του ότι το πλέγμα των εικονοστοιχείων έχει περιορισμένη ανάλυση
- Άρα θα πρέπει να κάνουμε μια **προσέγγιση**

Αλγόριθμοι Σχεδίασης Ευθειών

Οι επιθυμητές ιδιότητες ενός αλγόριθμου σχεδίασης ευθείας είναι:

01. **επιλογή** pixels που βρίσκονται όσο πιο **κοντά** γίνεται στη μαθηματική πορεία της ευθείας
02. όσο το δυνατόν **σταθερό** πάχος, **ανεξάρτητο** από μήκος/κλίση, ελάχιστο ίσο με 1 pixel
03. **μη** ύπαρξη κενών
04. **αποδοτικότητα** (οι αλγόριθμοι χρησιμοποιούνται συχνά)

Αλγόριθμος 1

Απλός αλγόριθμος, βασίζεται στην αλγεβρική εξίσωση της ευθείας

- Σχεδίαση ενός ευθύγραμμου τμήματος στο 1ο **οκταμόριο**
- Τα υπόλοιπα αντιμετωπίζονται με **συμμετρία**
- Για **οποιοδήποτε** σημείο $P(x, y)$ ενός ευθύγραμμου τμήματος
  - ισχύει:
    $$y = s \cdot x + b$$
  - όπου:
    $$s = \frac{y_n - y_1}{x_n - x_1} = \frac{\Delta y}{\Delta x}$$
  - και:
    $$b = \frac{y_1 x_n - y_n x_1}{x_n - x_1}$$

Αλγόριθμος 1

Με βάση τα παραπάνω, ο αλγόριθμος μπορεί να **υλοποιηθεί** ως:

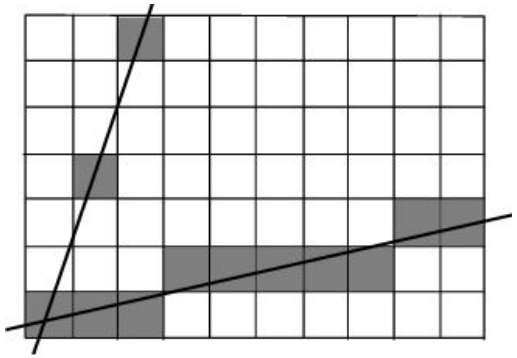
```
line1(x1,y1,xn,yn,colour)
  int x1,y1,xn,yn,colour
  float s,b,y; int x

  s=(yn-y1)/(xn-x1)
  b=(y1*xn-yn*x1)/(xn-x1)

  for (x=x1;x<=xn;x++)
    y=s*x+b
    setpixel(x,round(y),colour)
```

Προσοχή!

Αν ο αλγόριθμος χρησιμοποιηθεί για τον σχεδιασμό ευθείας με κύριο άξονα τον y θα δημιουργήσει κενά!



- Δείτε ότι στο **πρώτο** οκταμόριο δεν υπάρχουν κενά!

Αλγόριθμος 1

Μειονέκτημα

- Περιέχει πολλαπλασιασμό **μέσα** στην επανάληψη

Λύση

- Μπορεί να **αποφευχθεί** παρατηρώντας ότι σε κάθε επανάληψη

$$x_{i+1} = x_i + 1$$

- Άρα

$$y_{i+1} = sx_{i+1} + b = sx_i + b + s = y_i + s$$

- Ο πολλαπλασιασμός μπορεί να αντικατασταθεί από **πρόσθεση**!

Αλγόριθμος 2

Αυξητικός αλγόριθμος, ο υπολογισμός στο βήμα i βασίζεται στο βήμα $i-1$, αποφεύγεται ο υπολογισμός του b

```
line2 (x1,y1,xn,yn,colour)
  int x1,y1,xn,yn,colour
  float s,y; int x
  s=(yn-y1)/(xn-x1)
  y=y1
  for (x=x1;x<=xn;x++)
    setpixel(x,round(y),colour)
    y=y+s
```

Αλγόριθμος 2

Μειονέκτημα

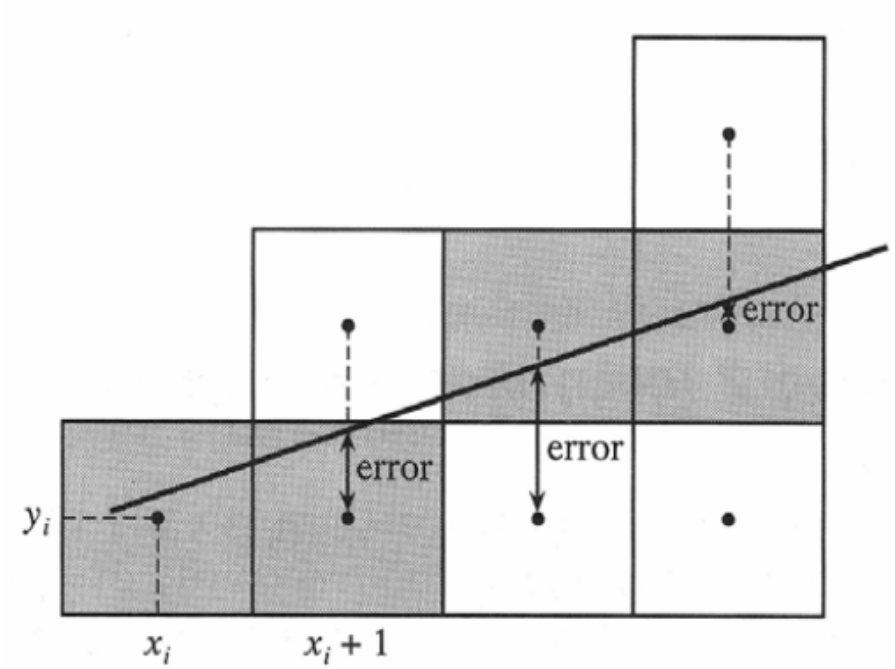
- Περιέχει **πράξη στρογγύλευσης** (round)

Λύση

- Να **χωριστεί** η *πραγματική* μεταβλητή y σε ακέραιο και δεκαδικό μέρος
- Η νέα μεταβλητή y θα είναι **ακέραια** και το δεκαδικό μέρος φυλάσσεται στην error

Αλγόριθμος 3

Θεωρώντας ότι έχει επιλεγεί το pixel (x_i, y_i), η error καθορίζει την **επιλογή** μεταξύ του $(x_i + 1, y_i)$ και $(x_i + 1, y_i + 1)$



Αλγόριθμος 3

```
line3 (x1,y1,xn,yn,colour)
  int x1,y1,xn,yn,colour
  float s,error; int x,y
  s=(yn-y1)/(xn-x1)
  y=y1
  error=0
  for (x=x1;x<=xn;x++)
    setpixel(x,y,colour)
    error=error+s
    if (error>=0.5)
      y++
      error--
```

Αλγόριθμος 4 - Bresenham

- Με κατάλληλη **κλιμάκωση** των s, error και της συνθήκης επιλογής του pixel οι πραγματικές μεταβλητές αντικαθίστανται από **ακέραιες**
- Αυτό **δεν** επηρεάζει την επιλογή του pixel
- Πολλαπλασιάζοντας με $dx = x_n - x_1$, τα s, error γίνονται **ακέραιοι**
- **Θέτουμε** $s = dy$
- Η **σύγκριση** για την επιλογή του επόμενου pixel γίνεται error$\geq dx/2$

Αλγόριθμος 4 - Bresenham

- Το error είναι πάντα **ακέραιο**, άρα αντί για $dx/2$ χρησιμοποιείται το $\lfloor dx/2 \rfloor$
- Το $\lfloor dx/2 \rfloor$ υπολογίζεται με **ολίσθηση**
- Το $\lfloor dx/2 \rfloor$ **αφαιρείται** από την αρχική τιμή του error
- Η **σύγκριση** για την επιλογή του επόμενου pixel γίνεται error≥ 0

Αλγόριθμος 4 - Bresenham

Ο ακόλουθος αλγόριθμος είναι γνωστός ως αλγόριθμος του **Bresenham**

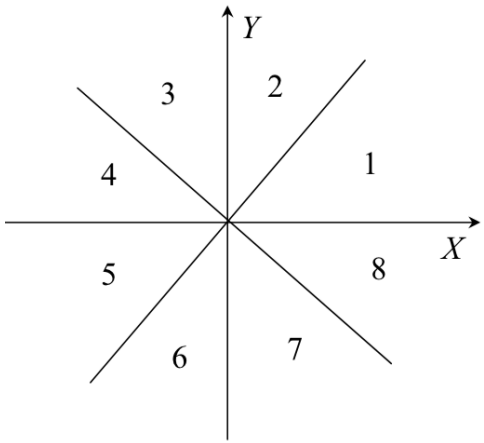
```
line4 (x1,y1,xn,yn,colour)
  int x1,y1,xn,yn,colour
  int error,x,y,dx,dy
  dx=xn-x1; dy=yn-y1
  error=-dx/2; y=y1
    for (x=x1; x<=xn; x++)
    setpixel(x,y,colour)
    error=error+dy
    if (error>=0)
      y++
      error=error-dx
```

Αλγόριθμος 4 - Bresenham

- Ο αλγόριθμος λειτουργεί για ευθύγραμμα τμήματα για τα οποία **ισχύει** $x_1 < x_n$ και έχουν **κλίση** μεταξύ 0 και 1
- Τα ανωτέρω ευθύγραμμα τμήματα βρίσκονται στο **1ο οκταμόριο**
- Ο αλγόριθμος εύκολα **επεκτείνεται** για τα υπόλοιπα οκταμόρια με εναλλαγές των x, y
- Σημειώνουμε ότι **ανταποκρίνεται** στις αρχικές απαιτήσεις ενός καλού αλγορίθμου σχεδίασης ευθείας
  01. επιλέγει τα **πλησιέστερα** εικονοστοιχεία στην μαθηματική πορεία της ευθείας (ισοδύναμος με τον `line1`)
  02. η έννοια του κύριου άξονα εξασφαλίζει **σταθερό πάχος** και **μη ύπαρξη κενών** (με 8-πλη σύνδεση)
  03. είναι ιδιαίτερα **αποτελεσματικός**, χρησιμοποιεί μόνο ακέραιες μεταβλητές και απλές πράξεις

Αλγόριθμος 4 - Bresenham

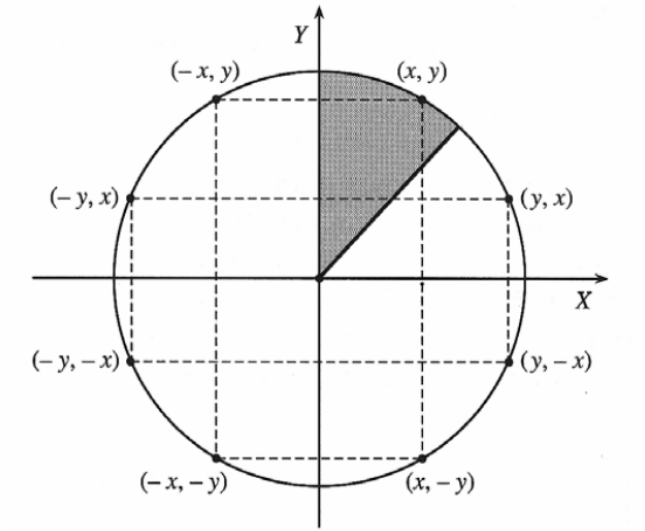
Σχέση οκταμορίων-μεταβλητών κίνησης



Οκταμόριο	Άξονας ταχυτ. Κίνησης	Άλλος άξονας
1	x	Αυξάνεται
2	y	»
3	y	Μειώνεται
4	x	»
5	x	Αυξάνεται
6	y	»
7	y	Μειώνεται
8	x	»

- Το οκταμόριο προσδιορίζεται με **μεταφορά** του ευθύγραμμου τμήματος, ώστε το (x_1, y_1) να **συμπέσει** με την αρχή των αξόνων
- Τότε το ευθύγραμμο τμήμα θα βρίσκεται μέσα σε **ένα** οκταμόριο

Αλγόριθμος Σχεδίασης Κύκλου



- Ο κύκλος χρησιμοποιείται σε πολλές εφαρμογές σαν **στοιχειώδες** σχήμα
- Κατά τη σχεδίαση κύκλου λαμβάνεται υπόψη η **8**-πλή **συμμετρία**
- Αρκεί να εξεταστεί η σχεδίαση σε ένα οκταμόριο π.χ. το **δεύτερο**

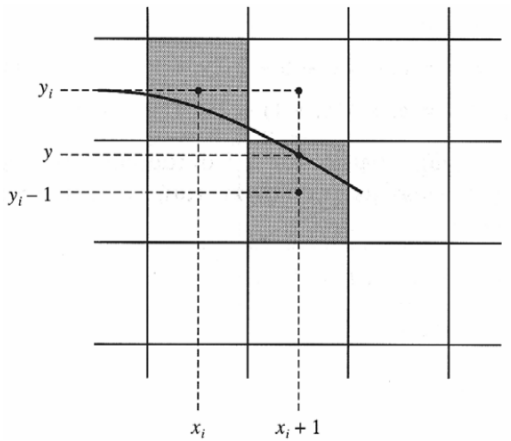
Αλγόριθμος Σχεδίασης Κύκλου

Για κύκλο με κέντρο το O και ακτίνα r, αν προσδιοριστεί σημείο (x, y), σχεδιάζονται τα 8 **συμμετρικά**

```
circlesymmetry (x,y,colour)
  int x,y,colour
  setpixel(x,y,colour)
  setpixel(y,x,colour)
  setpixel(y,-x,colour)
  setpixel(x,-y,colour)
  setpixel(-x,-y,colour)
  setpixel(-y,-x,colour)
  setpixel(-y,x,colour)
  setpixel(-x,y,colour)
```

Αλγόριθμος Σχεδίασης Κύκλου

- Για τον υπολογισμό των pixels του κύκλου που βρίσκονται στο 2ο οκταμόριο χρησιμοποιείται ο αλγόριθμος του **Bresenham**
- Έστω ότι έχει επιλεγεί το (x_i, y_i) ως το πλησιέστερο στην **ιδεατή** πορεία του κύκλου
- Στο 2ο οκταμόριο άξονας **ταχύτερης** κίνησης είναι ο x
- $(x_i + 1, y_i)$ ή $(x_i + 1, y_i - 1)$;



Αλγόριθμος Σχεδίασης Κύκλου

- Ο Bresenham έδειξε ότι μια καλή μεταβλητή **απόφασης** για την επιλογή του **κατάλληλου** pixel στο βήμα i είναι η

$$e_i = d_1 - d_2$$

όπου

$$d_1 = y_i^2 - y^2$$

και

$$d_2 = y^2 - (y_i - 1)^2$$

Αλγόριθμος Σχεδίασης Κύκλου

- **Άν** $e_i \geq 0$, επιλέγεται το $(x_i + 1, y_i - 1)$
- **διαφορετικά** επιλέγεται το $(x_i + 1, y_i)$
- $e_i = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$
- **και** $e_{i+1} = e_i + 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i)$
- Για τον **υπολογισμό** του e_{i+1}:
  - **Αν** $e_i < 0 \Rightarrow y_{i+1} = y_i \Rightarrow e_{i+1} = e_i + 4(x_i + 1) + 2$
  - **Αν**
    $e_i \geq 0 \Rightarrow y_{i+1} = y_i - 1 \Rightarrow e_{i+1} = e_i + 4(x_i + 1) + 2 - 4(y_i - 1)$
- Η αρχική τιμή e_1 **υπολογίζεται** ως
  $e_1 = 2 + r^2 + (r - 1)^2 - 2r^2 = 3 - 2r$

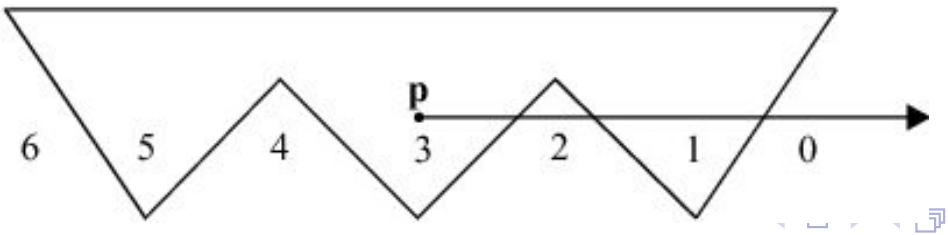
Αλγόριθμος Σχεδίασης Κύκλου

Για **κέντρο** το O χρησιμοποιείται η `circlesymmetry`. Αν το κέντρο είναι διαφορετικό, π.χ. το (x_c, y_c) τότε **αρκεί** να γίνει κλήση της ως `circlesymmetry(x+xc,y+yc,colour)`

```
circle (r,colour)
  int r,colour
  int x,y,e
  x=0; y=r
  e=3-2*r
  while (x<=y)
    circlesymmetry(x,y,colour)
    x++
    if (e>=0) y--; e=e-4*y
    e=e+4*x+2
```

Έλεγχοι εσωτερικού σημείου

- Το πιο διαδεδομένο στοιχείο για την κατασκευή μοντέλων **επιφανειών** είναι το **πολύγωνο** (τρίγωνο)
- Οι αλγόριθμοι σχεδίασης περιμέτρου και εσωτερικού ενός πολυγώνου βασίζονται στον έλεγχο για το εάν ένα **εικονοστοιχείο** βρίσκεται **εντός** του πολυγώνου
- Η **διαπίστωση** αν το (x, y) είναι **εντός** πολυγώνου (έλεγχος εσωτερικού σημείου) γίνεται ως εξής:
  - μετράται το **πλήθος** των σημείων τομής της γραμμής σάρωσης της οθόνης από τη θέση (x, y) μέχρι την (∞, y)
  - αν είναι **περιττός**, τότε το (x, y) είναι **εντός**
  - αν είναι **άρτιος**, τότε είναι **εκτός**



Σχεδίαση Πολυγώνου

- Αρχικά, θα περιγράψουμε αλγορίθμους που είναι κατάλληλοι για τη σχεδίαση **γενικών** πολυγώνων
- Στη συνέχεια, θα ασχοληθούμε με εξειδικευμένους αλγορίθμους για **τρίγωνα**
- Πρακτικά, το τρίγωνο αποτελεί το πιο **διαδεδομένο** στοιχείο κατασκευής **επιφανειών**
- Οι αλγόριθμοι σχεδίασης τριγώνων είναι πιο **απλοί** και υλοποιούνται σε επεξεργαστές γραφικών
- Για πολύγωνα μπορεί να χρησιμοποιηθούν οι αλγόριθμοι που θα δούμε στη συνέχεια, είτε αυτά να **μετατραπούν** σε τρίγωνα με αλγόριθμους **τριγωνοποίησης**
- Οι αλγόριθμοι **γεμίσματος** λειτουργούν με βάση τα περιεχόμενα του καταχωρητή εικόνας (frame buffer) και είναι χρήσιμοι σε εφαρμογές **2Δ** σχεδίασης

Βασικός αλγόριθμος σχεδίασης πολυγώνου (ΥΧ)

Αυτός ο απλός αλγόριθμος βασίζεται στον έλεγχο ισοτιμίας:

01 Για κάθε **πλευρά** προσδιορίζονται όλα τα σημεία **τομής** της με τις γραμμές σάρωσης: κατασκευάζεται **λίστα** (x, y) των σημείων τομής

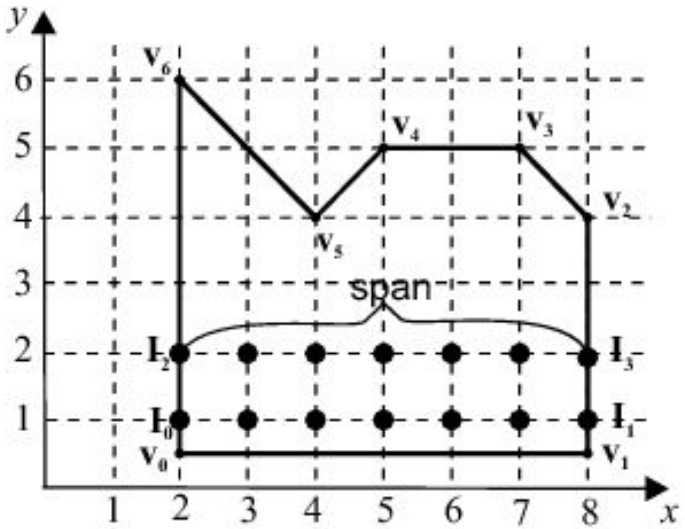
02 **Ταξινόμηση** της προηγούμενης λίστας με κύριο κλειδί το y και δευτερεύον το x

03 **Απομάκρυνση** κατά ζεύγη των στοιχείων από τη λίστα και **επιλογή** (άναμμα) των εικονοστοιχείων του ευθύγραμμου τμήματος που προσδιορίζεται από κάθε ζεύγος τομών με το χρώμα του πολυγώνου

- για τα **γειτονικά** σημεία τομής (x_1, y_1) και (x_2, y_2) ισχύει $y_1 = y_2$ και $x_1 \leq x_2$, άρα ο **χρωματισμός** γίνεται για τα pixels με $y = y_1$ και $x_1 \leq x \leq x_2$

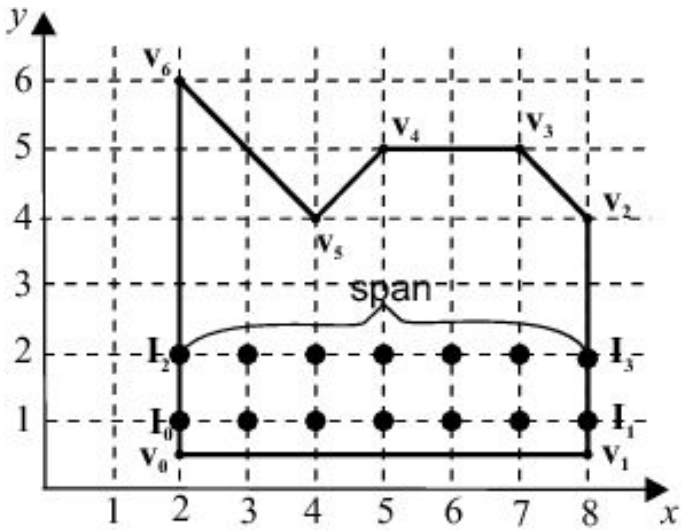
Βασικός αλγόριθμος σχεδίασης πολυγώνου (YX)

Ένα ζεύγος διαδοχικών τομών ονομάζεται **τμήμα** και αντιπροσωπεύει μια σειρά εικονοστοιχείων που είναι **εσωτερικά** του πολυγώνου, σύμφωνα με τον έλεγχο ισοτιμίας



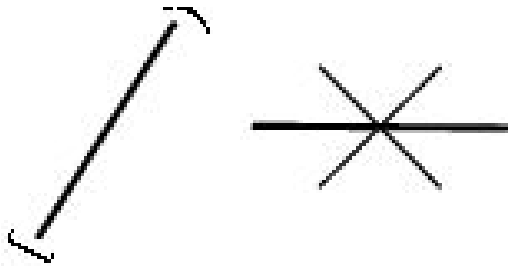
Ιδιάζοντα σημεία

Προσοχή στις **προβληματικές** περιπτώσεις:



- **Κορυφές** βρίσκονται **πάνω** σε **γραμμές** σάρωσης
- **Πόσα** σημεία τομής θεωρούνται για κάθε κορυφή; $2, 1$ ή 0;
- **Καμία** από αυτές τις επιλογές δεν είναι κατάλληλη για **όλες** τις περιπτώσεις!

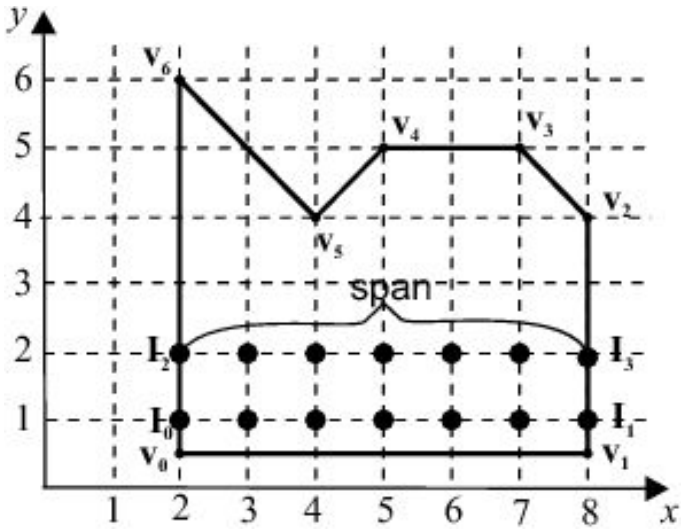
Ιδιάζοντα σημεία



- Μια απλή λύση είναι η ακόλουθη:
  1. Κάθε **μη-οριζόντια** πλευρά θεωρείται **ανοιχτή** στο άκρο με τη μεγαλύτερη y-τιμή, **κλειστή** στο άκρο με τη μικρότερη y-τιμή: φυλάσσεται η **τομή** μιας γραμμής σάρωσης με το κάτω άκρο μιας πλευράς
  2. Οι οριζόντιες πλευρές **αγνοούνται**

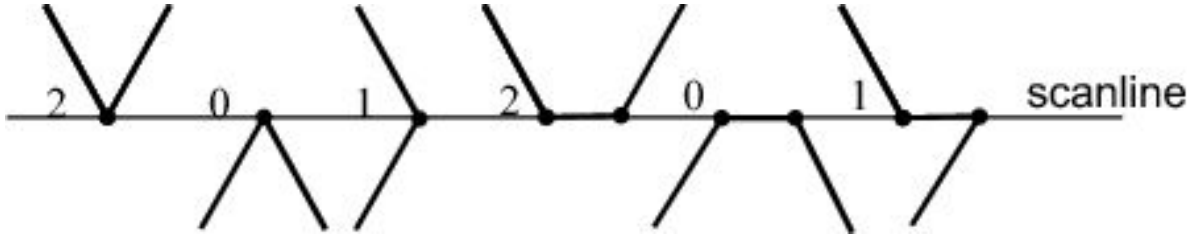
Ιδιάζοντα σημεία

Άρα στο προηγούμενο παράδειγμα:



- Η $\mathbf{v_2}$ θα μετρήσει σαν 1 τομή (με την $\mathbf{v_2v_3}$
- Η $\mathbf{v_5}$ σαν **δύο** τομές
- Η $\mathbf{v_6}$ σαν **μηδέν** τομές
- Οι $\mathbf{v_3}$ και $\mathbf{v_4}$ σαν **μηδέν** τομές αφού η $\mathbf{v_3v_4}$ είναι οριζόντια

Ιδιάζοντα σημεία



- Το **αποτέλεσμα** του κανόνα φαίνεται στο σχήμα
- Έτσι, **λύνεται** το πρόβλημα των ιδιαζόντων σημείων, αλλά το πολύγωνο σχεδιάζεται **ασύμμετρα** στην κατεύθυνση y
  - οριζόντιες ακμές στο **πάνω** μέρος του πολυγώνου και κορυφές που είναι τοπικά **μέγιστα** δεν σχεδιάζονται
  - οριζόντιες ακμές στο **κάτω** μέρος του πολυγώνου και κορυφές που είναι τοπικά **ελάχιστα** σχεδιάζονται

Αλγόριθμος σχεδίασης κατά γραμμές σάρωσης

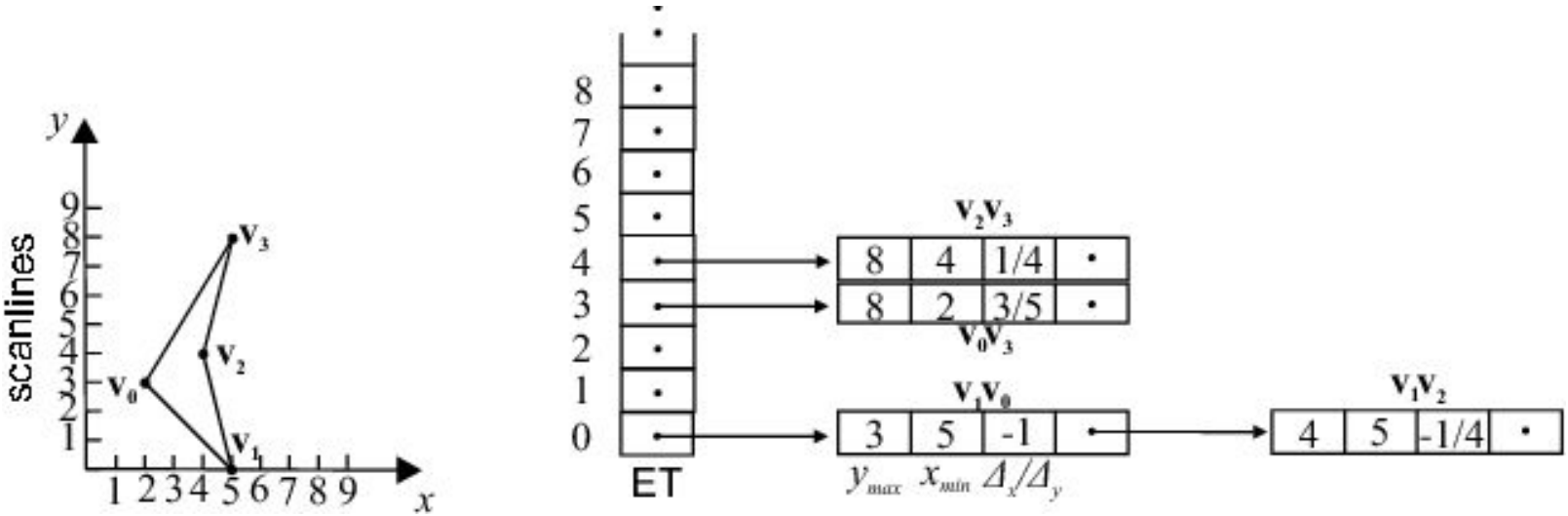
- Ο **βασικός** αλγόριθμος σχεδίασης που είδαμε, **δεν** είναι αποτελεσματικός
- Αυτό, γιατί η εύρεση του σημείου **τομής** είναι μια υπολογιστικά **ακριβή** πράξη
- Ο αλγόριθμος σχεδίασης πολυγώνου **κατά γραμμές σάρωσης** που θα δούμε **βελτιώνει** τον βασικό αλγόριθμο, καθώς:
  1. τα σημεία τομής γραμμής σάρωσης με πλευρά του πολυγώνου υπολογίζονται **γρήγορα**, για διαδοχικές γραμμές σάρωσης με βάση την κλίση της πλευράς
  2. το **κόστος** της ταξινόμησης **μειώνεται** με τη χρήση της **συνάφειας** του πολυγώνου
  3. λαμβάνονται υπόψη **μόνο** οι **τομές** με γραμμές **σάρωσης** που βρίσκονται **μεταξύ** των y-τιμών του πολυγώνου

Λίστα ενεργών ακμών

- Μπορούμε να φυλάμε σε μια **δομή** τις **τομές** της εκάστοτε γραμμής σάρωσης με τις ακμές του πολυγώνου
- Αυτή η δομή πρέπει να ενημερώνεται **αυξητικά**, κατά τη μετάβαση από μια γραμμή σάρωσης στην επόμενη
- Ονομάζεται λίστα ενεργών ακμών (**ΛΕΑ**)
- Η συνάφεια ακμών σημαίνει ότι μια ακμή αλλάζει με **προβλέψιμο** τρόπο
  - το σημείο τομής ακμής / γραμμής σάρωσης μπορεί να υπολογισθεί αυξητικά από μια γραμμή σάρωσης στην επόμενη, προσθέτοντας το **αντίστροφο** της κλίσης της ευθείας που ορίζει η ακμή $(1/s = \Delta x/\Delta y)$
- Χρησιμοποιείται ένας **πίνακας ακμών** για την ταξινόμηση με δοχεία των ακμών του πολυγώνου – κάθε δοχείο αντιστοιχεί σε μια **γραμμή** σάρωσης

Λίστα ενεργών ακμών

Ένα πολύγωνο και ο πίνακας ακμών (ΠΑ) του



Μια **εγγραφή** με τα απαραίτητα δεδομένα μιας ακμής εισάγεται στο δοχείο που αντιστοιχεί στην **ελάχιστη** y συντεταγμένη της

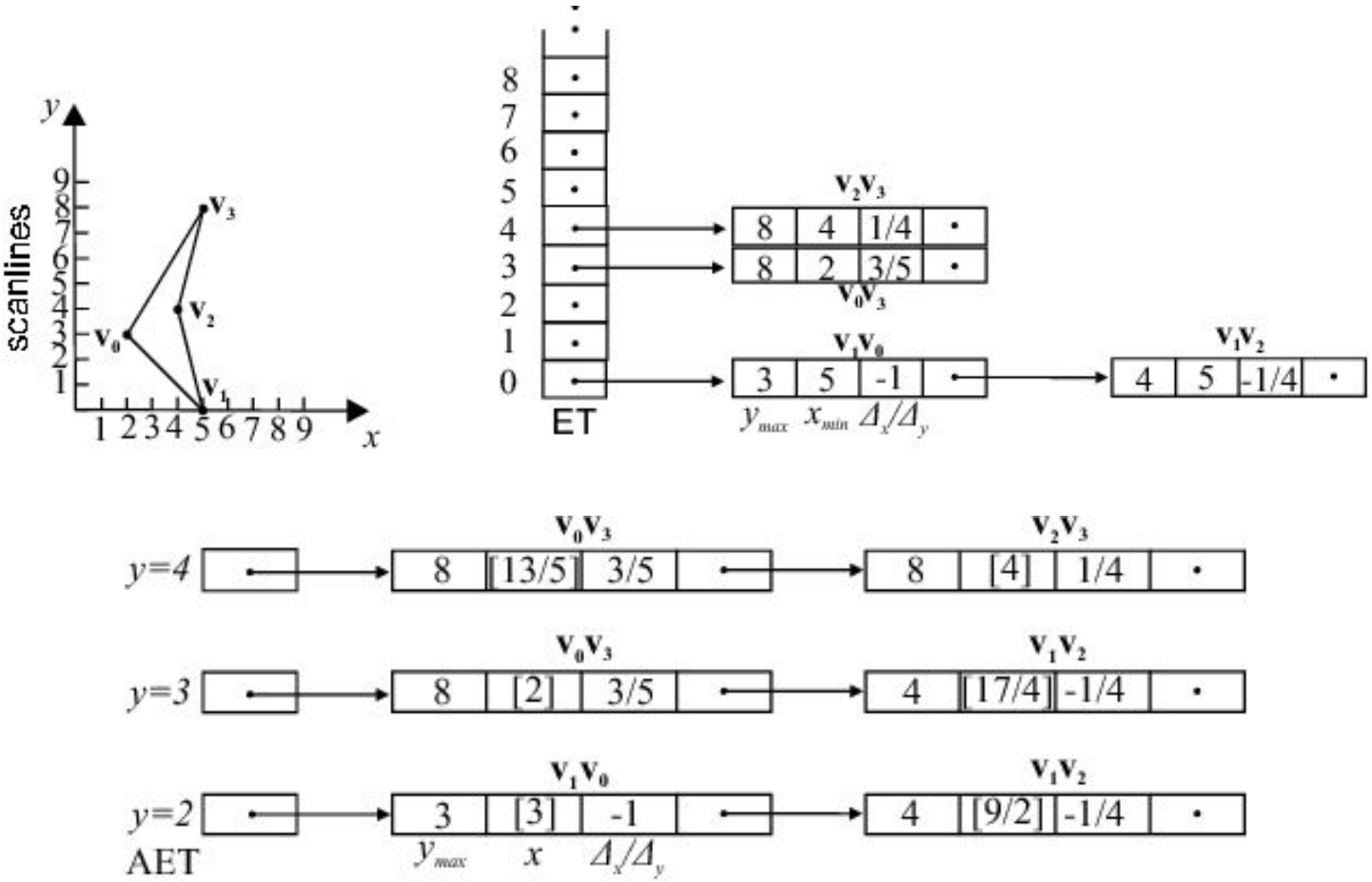
Αλγόριθμος σχεδίασης κατά γραμμές σάρωσης

01 **Δημιουργία ΠΑ** για το πολύγωνο που περιέχει εγγραφές $(y_{max}, x_{min}, 1/s)$ για κάθε ακμή – η εγγραφή μιας ακμής εισάγεται στο δοχείο που αντιστοιχεί στην y_{min}

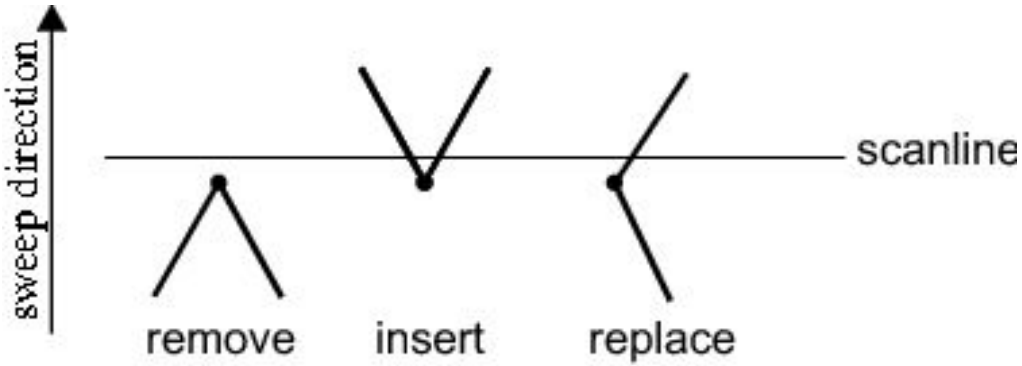
02 Για κάθε **γραμμή σάρωσης** y που τέμνει το πολύγωνο (από το ελάχιστο προς το μέγιστο y):

- α. **Ενημέρωση** των τομών της ΛΕΑ για την τρέχουσα γραμμή σάρωσης: $x = x + 1/s$
- β. **Εισαγωγή** ακμών από το δοχείο y του ΠΑ στην ΛΕΑ
- γ. **Αφαίρεση** ακμών από τη ΛΕΑ για τις οποίες ισχύει $y_{max} \leq y$
- δ. **Επαναταξινόμηση** της ΛΕΑ κατά x
- ε. **Εξαγωγή** τμημάτων (ζευγών εγγραφών) από τη ΛΕΑ και **επιλογή** των εικονοστοιχείων που αυτά ορίζουν

Παραδείγματα καταστάσεων της ΛΕΑ



Εξαγωγή ακμών από τοπικά μέγιστα



- Οι ακμές που βρίσκονται στη ΛΕΑ **αλλάζουν** στις κορυφές του πολυγώνου όπως φαίνεται στο σχήμα
  - Ένα τοπικό μέγιστο **εξάγει** δύο ακμές
  - Ένα τοπικό ελάχιστο **εισάγει** δύο ακμές
  - Οι άλλες κορυφές **αντικαθιστούν** μια ακμή με μια άλλη

Σχεδίαση με βάση τα κρίσιμα σημεία

- Στον προηγούμενο αλγόριθμο παρατηρήσαμε ότι **νέες** ακμές εισάγονται στη ΛΕΑ μόνο σε κορυφές που είναι **τοπικά ελάχιστα**
- Ο μοναδικός σκοπός του ΠΑ είναι η **φύλαξη** ακμών προς εισαγωγή στη ΛΕΑ στην κατάλληλη γραμμή σάρωσης
- Ο αλγόριθμος κρίσιμων σημείων **αποφεύγει** τη χρήση του ΠΑ και της υπολογιστικά ακριβής δημιουργίας του, κάνοντας χρήση των τοπικών ελαχίστων
- Οι κορυφές πολυγώνου που είναι τοπικά ελάχιστα ως προς την y συντεταγμένη τους ονομάζονται **κρίσιμα** σημεία

Αλγόριθμος Κρίσιμων Σημείων

01 **Εύρεση** και **φύλαξη** κρίσιμων σημείων του πολυγώνου

02 Για **κάθε** γραμμή σάρωσης y_i από την ελάχιστη ως τη μέγιστη τιμή του πολυγώνου, **επαναλαμβάνεται** η εξής διαδικασία:

α. Για κάθε **κρίσιμο** σημείο $\mathbf{c}$ με y μεταξύ των y_i και $y_i - 1$, ακολουθείται η **περίμετρος** του πολυγώνου προς τις δύο κατευθύνσεις που έχουν αφετηρία το $\mathbf{c}$. Η όδευση **σταματά** αν τμηθεί η γραμμή σάρωσης y ή βρεθεί τοπικό μέγιστο. Για κάθε τομή δημιουργείται μια εγγραφή στη ΛΕΑ $(v, \pm 1, x)$ που περιέχει τον αριθμό v της κορυφής έναρξης της τέμνουσας ακμής, την κατεύθυνση παρακολούθησης ($+1$ ωρολογιακά, -1 αντι-ωρολογιακά), x συντ/μένη σημείου τομής

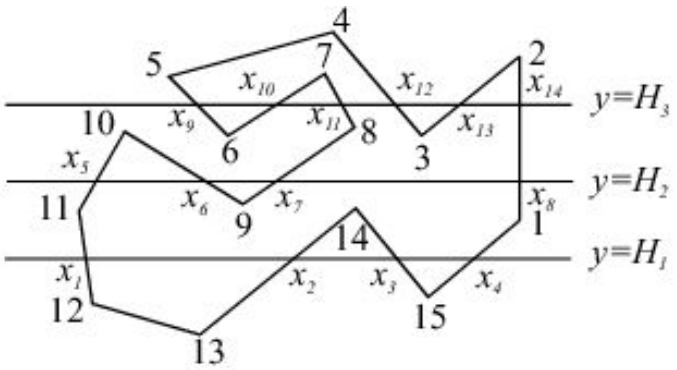
β. Για κάθε εγγραφή της ΛΕΑ που προϋπήρχε του προηγούμενου βήματος, γίνεται **όδευση** κατά μήκος της περιμέτρου, σύμφωνα με το δείκτη κατεύθυνσης της εγγραφής, μέχρι είτε να βρεθεί σημείο **τομής** μιας πλευράς της περιμέτρου με την γραμμή σάρωσης y, είτε ένα τοπικό **μέγιστο** – στην πρώτη περίπτωση, η εγγραφή **ανανεώνεται** με τη x-συντεταγμένη της νέας τομής, στη δεύτερη **διαγράφεται**

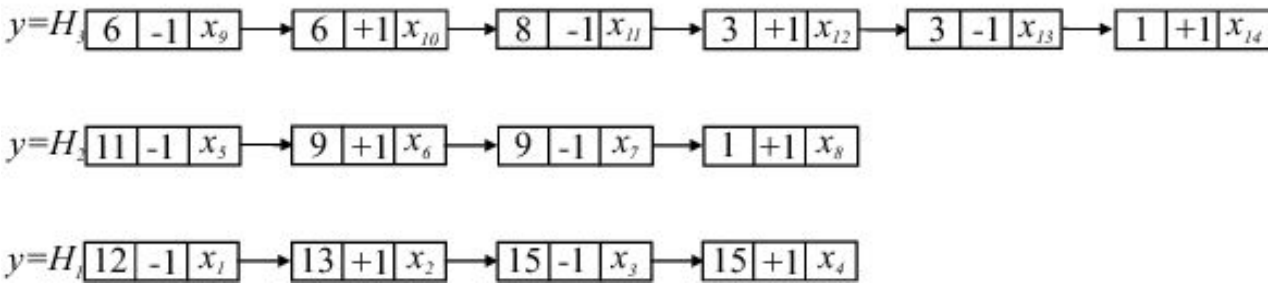
γ. **Επαταξινόμηση** των στοιχείων της ΛΕΑ κατά x, όπου χρειάζεται

δ. **Εξαγωγή** τμημάτων (ζευγών εγγραφών) από τη ΛΕΑ και **επιλογή** των εικονοστοιχείων που αυτά ορίζουν

Αλγόριθμος Κρίσιμων Σημείων

Σχεδίαση πολυγώνου και περιεχόμενα της ΛΕΑ για τρεις διαδοχικές γραμμές σάρωσης





Αλγόριθμοι σχεδίασης τριγώνων

- Το **τρίγωνο** είναι το **απλούστερο** πολύγωνο
- Είναι εξ ορισμού **επίπεδο** και **κυρτό**
- Είναι το πιο **συνηθισμένο** συστατικό των μοντέλων
- Υπάρχουν αλγόριθμοι για τη **μετατροπή** των περισσότερων τύπων επιφανειών σε **τριγωνικά πλέγματα**
- Έτσι, τα τρίγωνα χρήζουν **ειδικής** μεταχείρισης

Απλός αλγόριθμος σχεδίασης τριγώνου

- Ένας τρόπος εύρεσης των εικονοστοιχείων που αποτελούν την επιφάνεια του τριγώνου είναι η εκτέλεση **ελέγχου εσωτερικού σημείου** για **κάθε** εικονοστοιχείο του περιβάλλοντος κιβωτίου τριγώνου
- Αφού το τρίγωνο είναι ένα **κυρτό** πολύγωνο, ο έλεγχος εσωτερικού σημείου μπορεί να γίνει υπολογίζοντας τις τιμές των τριών γραμμικών συναρτήσεων που ορίζονται από τις **ακμές** του
- Για κάθε εικονοστοιχείο $\mathbf{p}$ του περιβάλλοντος κιβωτίου, αν οι τρείς συναρτήσεις δίνουν αποτελέσματα με το ίδιο πρόσημο, τότε το $\mathbf{p}$ είναι εντός του τριγώνου
- Αυτός ο έλεγχος μπορεί να **εξειδικευθεί** σε θετικό ή αρνητικό πρόσημο, αν είναι γνωστή η **φορά** ορισμού των κορυφών

Απλός αλγόριθμος σχεδίασης τριγώνου

```
triangle1(vertex v0, v1, v2, color c); {
    line l0, l1, l2;
    float e0, e1, e2, e0t, e1t, e2t;

    mkline(v0, v1, &l0), mkline(v1, v2, &l1), mkline(v2, v0, &l2);
    bb_xmin = min(v0.x, v1.x, v2.x), bb_xmax = max(v0.x, v1.x, v2.x);
    bb_ymin = min(v0.y, v1.y, v2.y), bb_ymax = max(v0.y, v1.y, v2.y);

    e0 = l0.a * bb_xmin + l0.b * bb_ymin + l0.c;
    e1 = l1.a * bb_xmin + l1.b * bb_ymin + l1.c;
    e2 = l2.a * bb_xmin + l2.b * bb_ymin + l2.c;

    for (y=bb_ymin; y<=bb_ymax; y++) {
        e0t = e0; e1t = e1; e2t = e2;
        for (x = bb_xmin; x<=bb_xmax; x++) {
            if (sign(e0)==sign(e1)==sign(e2)) setpixel(x,y,c);
            e0 = e0 + l0.a, e1 = e1 + l1.a, e2 = e2 + l2.a; }
        e0 = e0t + l0.b, e1 = e1t + l1.b, e2 = e2t + l2.b; }
}
```

Απλός αλγόριθμος σχεδίασης τριγώνου

- Αν το περιβάλλον κιβώτιο είναι **μεγάλο** σε σχέση με την επιφάνεια του τριγώνου, τότε ο `triangle1` δεν θα είναι αποδοτικός
  - θα υπολογίζει τις γραμμικές συναρτήσεις για ένα **μεγάλο** πλήθος εξωτερικών εικονοστοιχείων
- Ένας άλλος τρόπος είναι ο **περίπατος ακμών**
  - **τρεις** αλγόριθμοι σχεδίασης ευθείας τύπου **Bresenham** χρησιμοποιούνται για τον περίπατο των ακμών του τριγώνου
  - ο περίπατος γίνεται κατά γραμμή σάρωσης με **συγχρονισμό** των αλγορίθμων σχεδίασης ευθείας
  - έτσι τα άκρα ενός τμήματος εσωτερικών εικονοστοιχείων υπολογίζοται για **κάθε** γραμμή σάρωσης που τέμνει το τρίγωνο και τα εικονοστοιχεία του τμήματος **ενεργοποιούνται**

Αλγόριθμος γεμίσματος

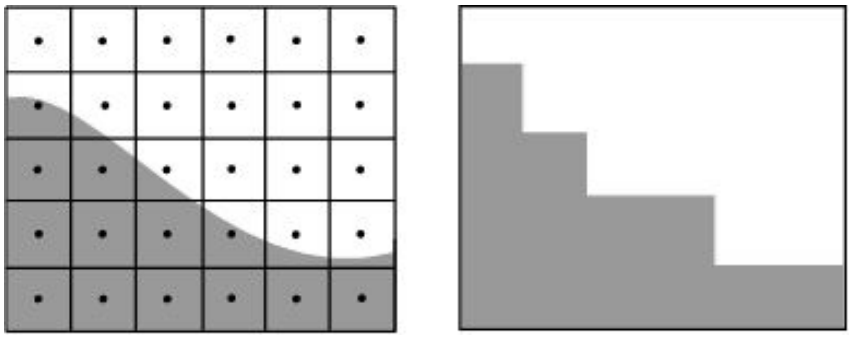
- Ένας απλός τρόπος **ενεργοποίησης** των εικονοστοιχείων που καλύπτει ένα **κλειστό** πολύγωνο (ή μια οποιαδήποτε κλειστή καμπύλη) είναι να σχεδιαστεί πρώτα η **περίμετρός** του και έπειτα να **γεμισθεί**, ξεκινώντας από ένα αρχικό **εσωτερικό** σημείο
- Οι αλγόριθμοι **γεμίσματος** χρησιμοποιούν τα περιεχόμενα του καταχωρητή εικόνας και είναι κατάλληλοι για ορισμένες **2Δ** εφαρμογές:
  1. Σχεδίαση της **περιμέτρου** του πολυγώνου
  2. Εύρεση ενός αρχικού **εσωτερικού** εικονοστοιχείου
  3. **Αναδρομική** ενεργοποίηση των εσωτερικών εικονοστοιχείων ξεκινώντας από το αρχικό προς όλες τις κατευθύνσεις, ώσπου να **συναντηθεί** η περίμετρος

Οπτικά προβλήματα ταύτισης

- Οι αλγόριθμοι σχεδίασης που είδαμε παίρνουν μια **δυαδική απόφαση** σχετικά με το αν ένα εικονοστοιχείο ανήκει σε έναν στοιχειώδες σχήμα ή όχι
- Η δυαδική αυτή απόφαση βασίζεται στη σχέση του **κέντρου** του εικονοστοιχείου με το σχεδιαζόμενο σχήμα (εντός/εκτός)
  - δηλαδή το εικονοστοιχείο θεωρείται **σημείο**
- Όμως, τα εικονοστοιχεία δεν είναι μαθηματικά σημεία, αφού καταλαμβάνουν μια μικρή **επιφάνεια**
- Η παραπάνω απλούστευση οδηγεί σε μια σειρά από **οπτικά προβλήματα ταύτισης**

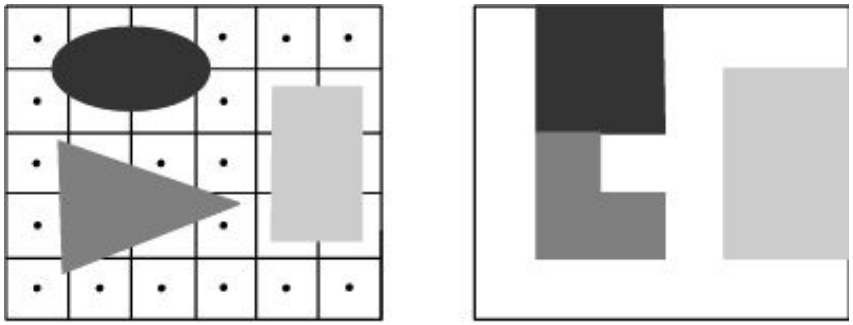
Οπτικά προβλήματα ταύτισης

01 Ευθείες και ακμές πολυγώνων (και γενικά, τα περιγράμματα αντικειμένων) μπορεί να έχουν **οδοντωτή** εμφάνιση



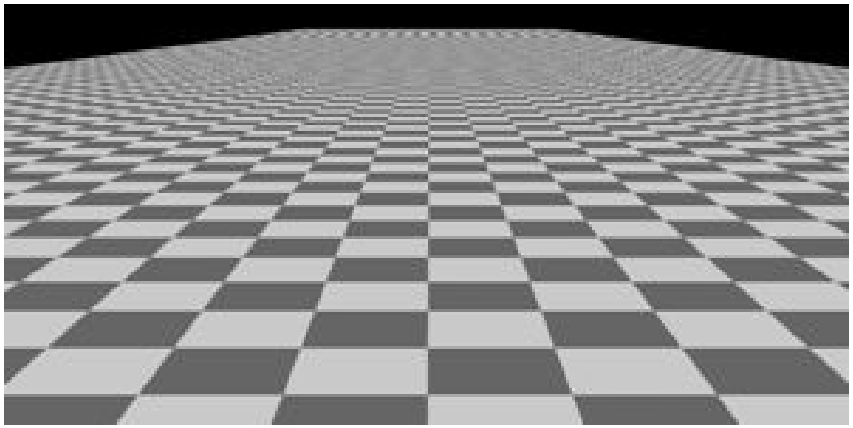
Οπτικά προβλήματα ταύτισης

02 Μικρά αντικείμενα μπορεί να σχεδιαστούν σε **λάθος μέγεθος** ή **λάθος σχήμα**. Ακόμα χειρότερα, κινούμενα μικρά αντικείμενα μπορεί να **εξαφανιζονται** από καρέ σε καρέ, ανάλογα με το αν πέφτουν πάνω σε κέντρα εικονοστοιχείων ή όχι



Οπτικά προβλήματα ταύτισης

03 Λεπτομέρειες όπως η **υφή** μπορεί να σχεδιαστούν **λανθασμένα**



Θεωρία δειγματοληψίας και ταύτιση

- Στη θεωρία δειγματοληψίας το πρόβλημα της ταύτισης είναι **εκτενώς** μελετημένο
- Εμφανίζεται όταν το δειγματοληπτούμενο σήμα περιέχει συχνότητες **μεγαλύτερες του μισού** της συχνότητας δειγματοληψίας (θεώρημα **Nyquist**)
- Στα γραφικά, το δειγματοληπτούμενο σήμα είναι το **μαθηματικό μοντέλο** της εικόνας (που αποτελείται π.χ. από ευθείες και πολύγωνα)
- Η συχνότητα δειγματοληψίας είναι η **ανάλυση** του πλέγματος των εικονοστοιχείων
- Τα **κέντρα** των εικονοστοιχείων είναι τα σημεία δειγματοληψίας

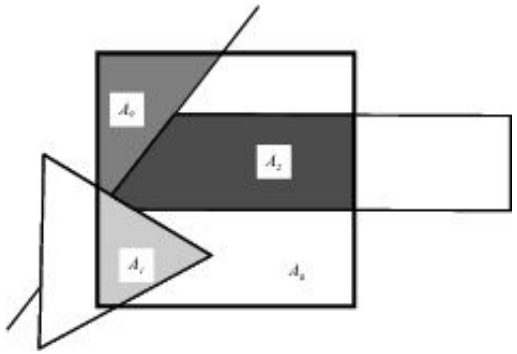
Μέθοδοι αντιταύτισης

- Δεν είναι απλό να μετρηθεί με ακρίβεια η **μέγιστη** συχνότητα του μαθηματικού μοντέλου μιας εικόνας
- Μπορούν να χρησιμοποιηθούν μέθοδοι **αντιταύτισης**, βασιζόμενες στη θεωρία της δειγματοληψίας
- Ουσιαστικά, η αντιταύτιση **ανταλλάσσει** χρωματική ανάλυση για χωρική ανάλυση
- Είναι το αντίθετο της **αυτοτυπίας** (halftoning) που θα μελετήσουμε σε επόμενο μάθημα
- Ανάλογα με τον **τρόπο** που αντιμετωπίζονται οι υψηλές συχνότητες, οι μέθοδοι αντιταύτισης μπορούν να διακριθούν σε δύο κατηγορίες:
  01. **Προ-φιλτράρισμα**: εξάγει τις υψηλές συχνότητες **πριν** τη δειγματοληψία
  02. **Μετα-φιλτράρισμα**: εξάγει τις υψηλές συχνότητες **μετά** τη δειγματοληψία

Αντιταύτιση με προ-φιλτράρισμα – πολύγωνα

- Ο Catmull πρότεινε τη θεώρηση του εικονοστοιχείου ως ένα **τετράγωνο παράθυρο**
- Συγκεκριμένα, στα όρια του παραθύρου **αποκόπτονται** τα πολύγωνα που έχουν **επικάλυψη** με αυτό
- Μετά την απομάκρυνση των κρυμμένων τμημάτων, υπολογίζεται η ορατή επιφάνεια κάθε πολυγώνου σαν **ποσοστό** της συνολικής επιφάνειας του εικονοστοιχείου
- Αυτή είναι η **συνεισφορά** του χρώματος του πολυγώνου στο χρώμα του εικονοστοιχείου
- Ο αλγόριθμος του Catmull μπορεί να θεωρηθεί το **ιδανικό** με το οποίο συγκρινονται οι αλγόριθμοι αντιταύτισης, αλλά γενικά **δεν** είναι πρακτικός λόγω του μεγάλου υπολογιστικού του κόστους

Ο αλγόριθμος του Catmull

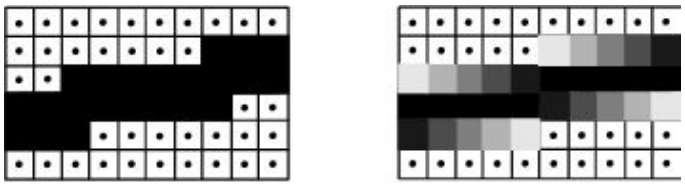


01 **Αποκοπή** όλων των πολυγώνων στα όρια του εικονοστοιχείου, έστω $P_0, \ldots, P_{n-1}$ τα εναπομείναντα τμήματα πολυγώνων

02 **Απομάκρυνση κρυμμένων τμημάτων**: ταξινομούντα τα πολύγωνα $P_0, \ldots, P_{n-1}$ κατά βάθος και την αποκοπή τους στην περιοχή που σχηματίζεται, αφαιρώντας τα πολύγωνα από την εκάστοτε εναπομένουσα επιφάνεια του εικονοστοιχείου κατά σειρά βάθους και προκύπτουν τελικά τα ορατά τμήματα των πολυγώνων $P_0, \ldots, P_{m-1}$ με επιφάνειες $A_0, \ldots, A_{m-1}$

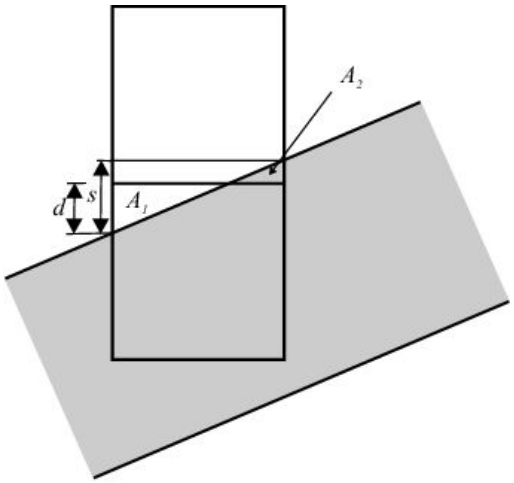
03 Υπολογισμός του **τελικού χρώματος** του εικονοστοιχείου ως $A_0C_0 + A_1C_1 + \ldots + A_{m-1}C_{m-1} + A_BC_C$, C_i το χρώμα του πολυγώνου i. Τα A_B, C_B αντιπροσωπεύουν την επιφάνεια και το χρώμα του φόντου

Σχεδιαση ευθείας με αντιταύτιση



- Οι **ευθείες** που παράγονται από τον αλγόριθμο του Bresenham έχουν **οδοντωτή** εμφάνιση
- Το ίδιο συμβαίνει και με τις **ακμές** πολυγώνων
- Αυτό είναι αποτέλεσμα της δυαδικής **απόφασης** που λαμβάνεται κατά την επιλογή του πλησιέστερου εικονοστοιχείου στη μαθηματική πορεία της ευθείας
- Καθώς έχουν **πάχος**, θα μπορούσαν να προσομοιωθούν από μακρά και λεπτά **παραλληλόγραμμα**
  - Στην περίπτωση αυτή, η δυαδική επιλογή **δεν** θα ήταν σωστή: θα έπρεπε να παίρνουν χρώμα αντίστοιχο του **ποσοστού** τους που **καλύπτεται** από την ευθεία

Σχεδιαση ευθείας με αντιταύτιση

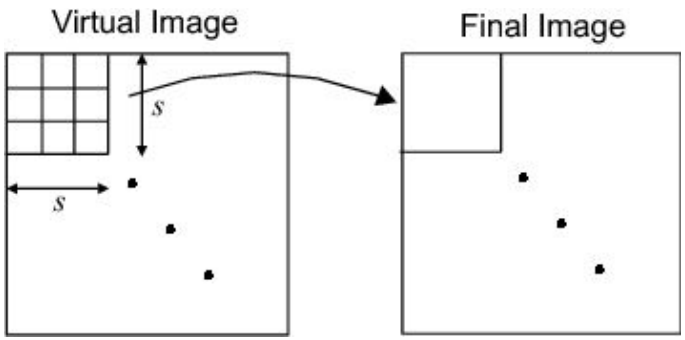


- Ας θεωρήσουμε πάλι μια ευθεία με **κλίση** $s = -\alpha/b$ στο **πρώτο** οκταμόριο και δύο εικονοστοιχεία που καλύπτει **μερικώς** σε ένα βήμα της πορείας της
- Αν το **πάνω** πάρει το χρώμα της ευθείας σε ποσοστό A_2 ενώ το **κάτω** σε $1 - A_1$ τα εικονοστοιχεία θεωρηθούν ως **μοναδιαία** τετράγωνα, τότε $A_1 = d^2/2s, A_2 = \frac{(s-d)^2}{2s}$

Αντιταύτιση με μετα-φιλτράρισμα

- Στο **μετα-φιλτράρισμα** (υπερ-δειγματοληψία) λαμβάνουμε **πάνω** από ένα δείγμα/εικονοστοιχείο
- Αυτό ισοδυναμεί με την κατασκευή της εικόνας σε **μεγαλύτερη** ανάλυση
- Τα επιπλέον δείγματα λαμβάνονται σε **ισαπέχοντα** διαστήματα που συνιστούν ένα πλέγμα **πυκνότερο** από αυτό των εικονοστοιχείων
- Στη συνέχεια, το αποτέλεσμα μετατρέπεται στην ανάλυση των εικονοστοιχείων με ένα φίλτρο **συνέλιξης** (π.χ., με μέσους όρους)
- Συνήθως, αντί για το μέσο όρο των δειγμάτων χρησιμοποιείται ένα **φίλτρο συνέλιξης** και εκτελείται η κλασική πράξη της συνέλιξης

Αντιταύτιση με μετα-φιλτράρισμα – παράδειγμα



- Για να δημιουργήσουμε μια εικόνα ανάλυσης 1024×1024, μπορούμε να πάρουμε 3072×3072 δείγματα
- Αυτά αντιστοιχούν σε 9 δείγματα/εικονοστοιχείο
- Στη συνέχεια, λαμβάνεται ο **μέσος όρος** των εννέα δειγμάτων για τον υπολογισμό του τελικού χρώματος του εικονοστοιχείου

Μειονεκτήματα μετα-φιλτραρίσματος

Υπάρχουν δύο **παράμετροι** που οδηγούν τον βασικό αλγόριθμο του μετα-φιλτραρίσματος: το **μέγεθος** s του φίλτρου συνέλιξης και η επιλογή των **βαρών** (όσο μεγαλύτερο είναι το s, τόσο καλύτερα είναι τα αποτελέσματα)

- Τα βασικά **προβλήματα** του μετα-φιλτραρίσματος είναι:
  1. η αύξηση του s **αυξάνει** και το χρόνο δημιουργίας της εικόνας, καθώς και τον απαιτούμενο χώρο για την αποθήκευση της εικόνας-είδωλο
  2. θεωρητικά, αφού οι συχνότητες μιας εικόνας δεν είναι φραγμένες, το πρόβλημα της ταύτισης θα **παραμένει** – πρακτικά, αρκεί να ικανοποιείται το ανθρώπινο **μάτι**
  3. το μετα-φιλτράρισμα δεν είναι ευαίσθητο στην **τοπική** πολυπλοκότητα της εικόνας: αυξάνεται κατά s η ανάλυση **χωρίς** αυτό να χρειάζεται πάντα και έτσι γίνονται πολλοί **άχρηστοι** υπολογισμοί

Αποκοπή

- Η πράξη της **αποκοπής** (clipping), προέκυψε από την ανάγκη να **μην** επιχειρείται η σχεδίαση αντικειμένων **εκτός** των ορίων σχεδίασης της συσκευής απεικόνισης
- Η αποκοπή ενός **αντικειμένου** (π.χ. ευθύγραμμου τμήματος, πολυγώνου) γίνεται ως προς συγκεκριμένο **αντικείμενο αποκοπής** (π.χ. ορθογώνιο παραλληλόγραμμο, πυραμίδα, κύβος)
  - Το **αντικείμενο αποκοπής** ορίζει το τμήμα του χώρου που ενδιαφέρει τον παρατηρητή σχετικά με το ποιες επιφάνειες μιας σκηνής μπορεί **αυτός** να δει
- Η αποκοπή είναι πολύ **χρήσιμη** για τους εξής λόγους
  - **Αποφυγή** της λανθασμένης (αντεστραμμένης) εμφάνισης αντικειμένων που βρίσκονται **πίσω** από τον παρατηρητή
  - **Μείωση** του όγκου των δεδομένων που προωθούνται προς την αναπαράσταση στην οθόνη

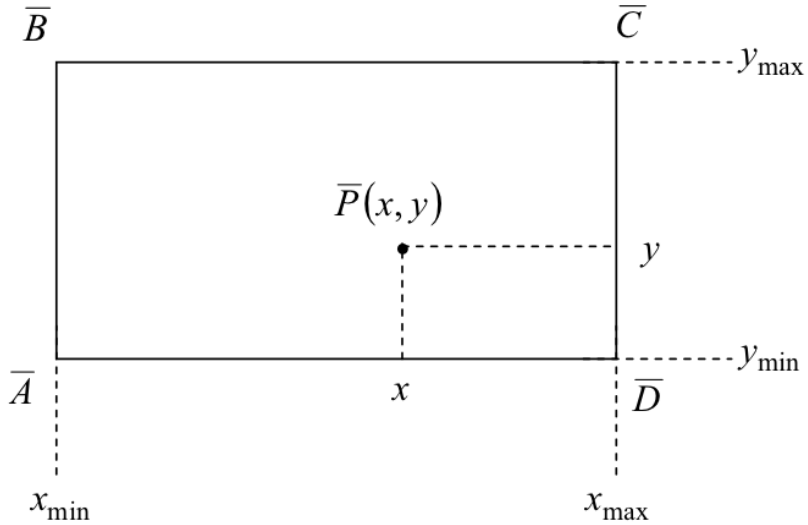
Σχήμα και αντικείμενο αποκοπής

Επειδή το τελικό μέσο παράστασης (**οθόνη**) είναι ορθογώνιο παραλληλόγραμμο, για αντικείμενο αποκοπής θα χρησιμοποιήσουμε ένα ορθογώνιο παραλληλόγραμμο το οποίο και θα αποκαλείται **παράθυρο αποκοπής** (clipping window)

- Ένα στοιχειώδες **σχήμα προς σχεδίαση** μπορεί να έχει **τρεις** πιθανές σχέσεις με το αντικείμενο αποκοπής:
  1. Να βρίσκεται ολόκληρο **εντός** του αντικειμένου αποκοπής: σχεδίαζεται **κανονικά**
  2. Να βρίσκεται ολόκληρο **εκτός** του αντικειμένου αποκοπής: **δεν** σχεδιάζεται
  3. Να **τέμνει** το αντικείμενο αποκοπής: η τομή των δύο πρέπει να **βρεθεί** και να **σχεδιασθεί**

Οι αλγόριθμοι αποκοπής ασχολούνται κυρίως με την **τρίτη** περίπτωση!

Αποκοπή Σημείου

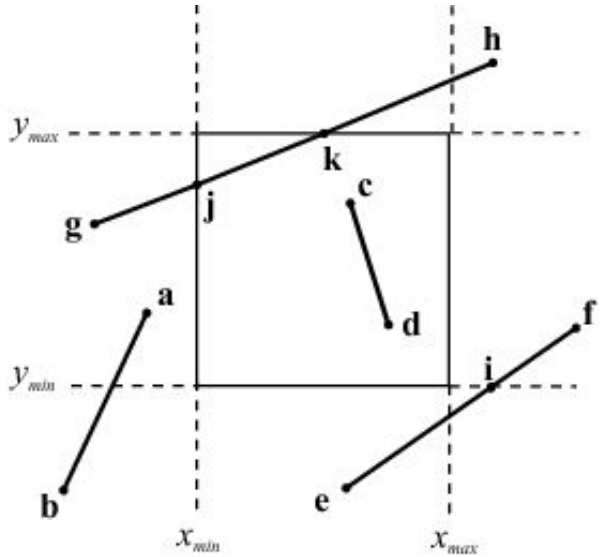


- Στην περίπτωση αυτή εξετάζεται αν το σημείο $\mathbf{p}(x, y)$ βρίσκεται **εντός** ή **εκτός** του αντικειμένου αποκοπής
- Π.χ., για ορθογώνιο παραλληλόγραμμο που ορίζεται από τα σημεία $\mathbf{a}(x_{min}, y_{min})$ και $\mathbf{c}(x_{max}, y_{max})$
  - Εάν $x_{min} \leq x \leq x_{max}$ **και** $y_{min} \leq y \leq y_{max}$, τότε το $\mathbf{p}(x, y)$ βρίσκεται **εντός** του παραθύρου, διαφορετικά **αποκόπτεται**

Αλγόριθμος Cohen-Sutherland

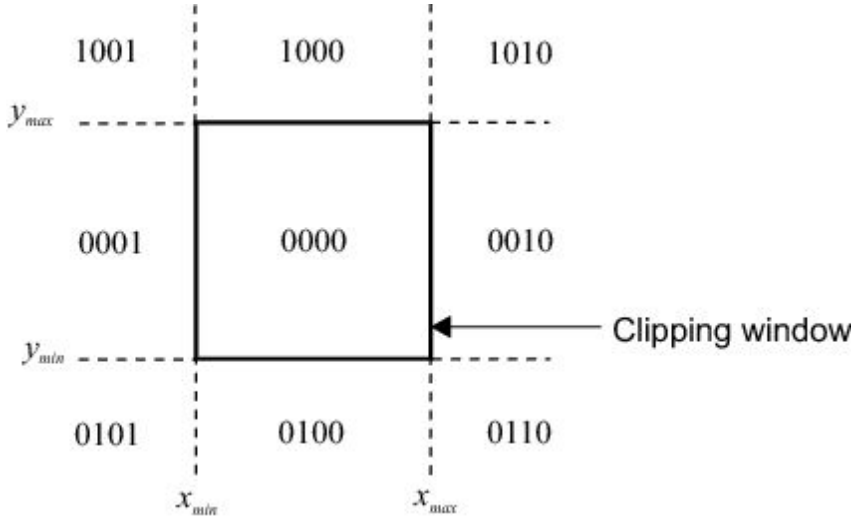
- Ο αλγόριθμος αρχικά σαν πρώτο βήμα εκτελεί έναν υπολογιστικά **φθηνό** έλεγχο για να αποφανθεί κατά πόσον το ευθύγραμμο τμήμα είναι εξ ολοκλήρου **εντός** ή **εκτός** του παραθύρου
  - Ο έλεγχος αυτός χρησιμοποιεί την επικάλυψη του ευθύγραμμου τμήματος και του παραθύρου αποκοπής στους άξονες x, y
- Εάν μπορεί να ληφθεί μια απόφαση, αποφεύγεται ο **δαπανηρός** υπολογισμός της τομής
- Διαφορετικά, πρέπει να υπολογιστεί η **τομή** του ευθύγραμμου τμήματος με **μία** από τις ευθείες που ορίζουν οι **ακμές** του παραθύρου αποκοπής
- Έτσι, το ευθύγραμμο τμήμα **διασπάται** και ο αλγόριθμος εφαρμόζεται **αναδρομικά** στα δύο **νέα** τμήματα

Αλγόριθμος Cohen-Sutherland



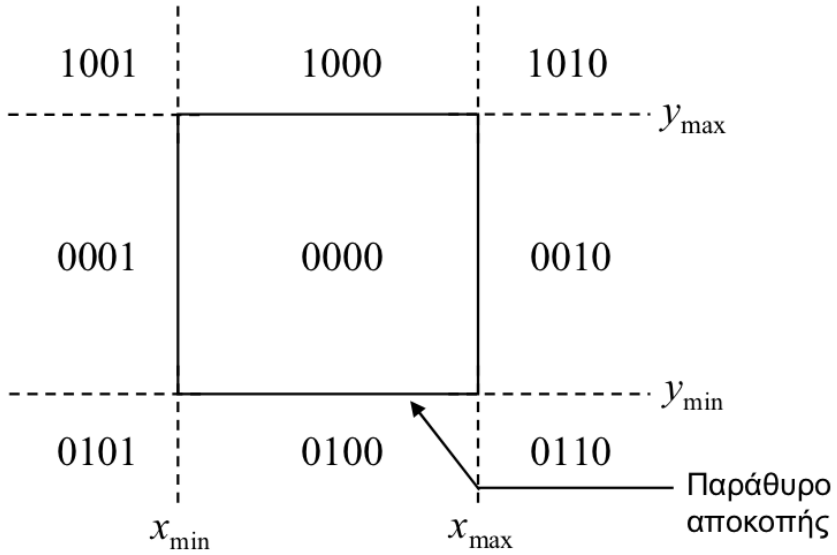
- Το **ab** είναι **εκτός**, αφού οι x συντεταγμένες των άκρων του είναι μικρότερες από x_min
- Το **cd** είναι **εντός**, αφού όλες οι συντεταγμένες των άκρων του είναι μεταξύ από x_{min}, x_{max} και y_{min}, y_{max}
- **Δεν** μπορεί να εξαχθεί συμπέρασμα για τα **gh** και **ef**

Αλγόριθμος Cohen-Sutherland



- Για να γίνουν **αποδοτικά** οι έλεγχοι, το επίπεδο του παραθύρου χωρίζεται σε **9 περιοχές**
- Σε κάθε περιοχή αντιστοιχεί ένας **κώδικας** που καθορίζεται απο τη **θέση** της περιοχής σχετικά με τα **ημιεπίπεδα** των ακμών του παραθύρου

Αλγόριθμος Cohen-Sutherland



- Κάθε ένα από τα 4 bits του κώδικα έχει τιμή 1 αν η περιοχή βρίσκεται στο ημιεπίπεδο της ακμής που **ΔΕΝ** περιέχει το παράθυρο, αλλιώς έχει την τιμή 0
  - **1ο** bit = 1: περιοχή **πάνω** από την ευθεία $y = y_{max}$
  - **2ο** bit = 1: περιοχή **κάτω** από την ευθεία $y = y_{min}$
  - **3ο** bit = 1: περιοχή **δεξιά** από την ευθεία $x = x_{max}$
  - **4ο** bit = 1: περιοχή **αριστερά** από την ευθεία $x = x_{max}$

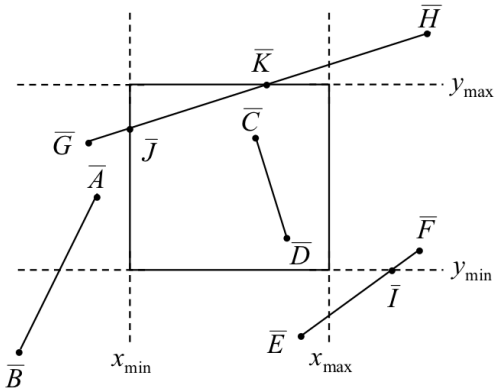
Αλγόριθμος Cohen-Sutherland

- Για κάθε άκρο (x, y) ενός ευθυγράμμου τμήματος υπολογίζεται ο **κώδικας** της περιοχής στην οποία βρίσκεται
- Αυτό γίνεται **αποδοτικά** ως εξής:
  - το **1ο** bit αντιστοιχεί στο πρόσημο της παράστασης $y_{max} - y$
  - το **2ο** bit αντιστοιχεί στο πρόσημο της παράστασης $y - y_{min}$
  - το **3ο** bit αντιστοιχεί στο πρόσημο της παράστασης $x_{max} - x$
  - το **4ο** bit αντιστοιχεί στο πρόσημο της παράστασης $x - x_{min}$
- Έχοντας υπολογίσει τους κώδικες c_1, c_2 των άκρων κάποιου ευθύγραμμου τμήματος, το **1ο βήμα της αποκοπής** είναι
  - Αν $c_1 \vee c_2 = 0000$, τότε το ευθύγραμμο τμήμα είναι **εντός** του παραθύρου
  - Αν $c_1 \wedge c2 \neq 0000$, τότε το ευθύγραμμο τμήμα είναι **εκτός** του παραθύρου

Αλγόριθμος Cohen-Sutherland

- Στην περίπτωση που δεν ισχύει κάτι από τα παραπάνω, προσδιορίζεται ένα **σημείο τομής** του ευθύγραμμου τμήματος με μία από τις ευθείες που ορίζονται από τις ακμές του **παραθύρου** αποκοπής ως εξής
  - Επιλέγουμε μια ακμή που να αντιστοιχεί σε ψηφίο με **διαφορετικές** τιμές στους κωδικούς c_1, c_2
  - Μετά, γίνεται **αναδρομική** κλήση του αλγορίθμου με το **εσωτερικό** τμήμα του ευθύγραμμου τμήματος μόνο (δηλαδή, αυτό που είχε στο ψηφίο τιμή 0)

Αλγόριθμος Cohen-Sutherland – Παράδειγμα



Άκρο	Κωδικός	Άκρο	Κωδικός
a	0001	**e**	0100
b	0101	**f**	0010
c	0000	**g**	0001
d	0000	**h**	1010

- το **ab** είναι ολόκληρο **εκτός**, αφού $0001 \wedge 0101 \neq 0000$
- το **cd** είναι ολόκληρο **εντός**, αφού $0000 \vee 0000 = 0000$
- για τα **ef**, **gh** ο απλός έλεγχος **δεν** καταλήγει, οπότε πρέπει να υπολογισθούν σημεία **τομής**
- **τέμνουμε** το **ef** με την ευθεία $y = y_{min}$, αφού το δεύτερο ψηφίο του κωδικού έχει διαφορετική τιμή στα άκρα **e**, **f**
- εκτελείται **αναδρομική** κλήση με το **if**, καθώς το δεύτερο ψηφίο του κωδικού του άκρου **f** έχει τιμή 0 (εντός)
- **παρομοίως**, και για το **gh**

Αλγόριθμος Cohen-Sutherland

```
CS (vertex p1, p2, float xmin, xmax, ymin, ymax)
     int c1, c2;
     vertex i;
     edge e;

     c1=mkcode(P1); //find P1 code
     c2=mkcode(P2); //find P2 code
     if ((c1||c2)==0) // P1P2 is inside the window
     else if ((c1&&c2)!=0) // P1P2 is outside the window
     else
          e = //window edge with (c1 bit != c2 bit)
          i = intersect_lines(e, (p1,p2));
          if outside(e,p1)  CS(i,p2,xmin,xmax,ymin,ymax);
          else CS(p1,i,xmin,xmax,ymin,ymax);
```

- Η `intersect_lines` υπολογίζει την τομή του ευθύγραμμου τμήματος από το $\mathbf{p_1}$ στο $\mathbf{p_2}$ με την ευθεία του παραθύρου αποκοπής e
- Η `outside` αποφασίζει αν ένα σημείο βρίσκεται στο εξωτερικό ή στο εσωτερικό ημιεπίπεδο της e και αν η ευθεία είναι παράλληλη με άξονα συντεταγμένων

Αλγόριθμος Liang-Barsky

- Ο αλγόριθμος Liang-Barsky επιλύει το πρόβλημα της αποκοπής ευθείων με **άμεσο** τρόπο
- **Αποφεύγει** τις πιθανές αναδρομικές κλήσεις του αλγορίθμου Cohen-Sutherland
- Έχει απόδοση **βελτιωμένη** κατά περίπου 30%
- Βασίζεται στην **παραμετρική εξίσωση ευθείας**
- Για $\mathbf{p}_1(x_1, y_1)$ και $\mathbf{p}_2(x_2, y_2)$, η παραμετρική εξίσωση των $\mathbf{p}_1, \mathbf{p}_2$ είναι

$$\mathbf{p} = \mathbf{p}_1 + t \cdot (\mathbf{p}_2 - \mathbf{p}_1), \quad t \in [0, 1]$$

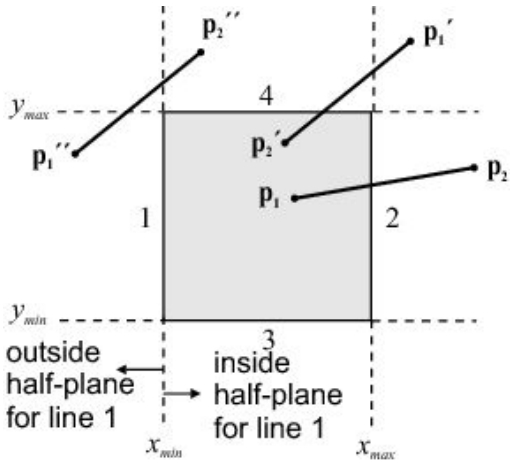
ή

$$x = x_1 + t \cdot \Delta x, \quad y = y_1 + t \cdot \Delta y$$

με

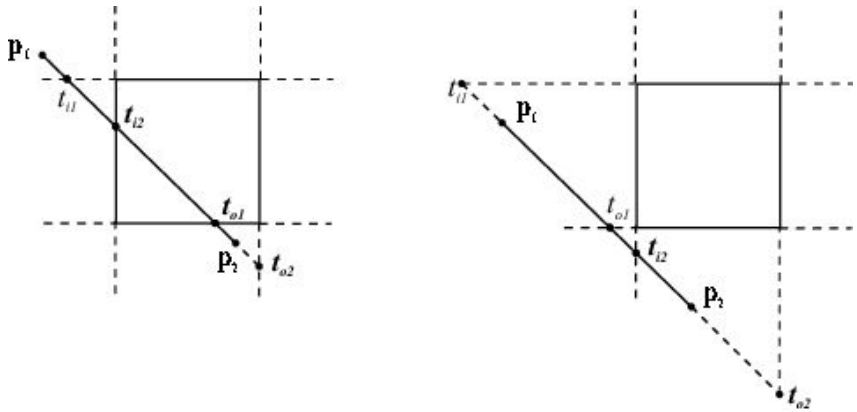
$$\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1$$

Αλγόριθμος Liang-Barsky



- Έστω **προέκταση** του παραθύρου αποκοπής που αποτελείται από τις **ευθείες** που ορίζονται από τις ακμές του
- Το κατευθυνόμενο ευθύγραμμο τμήμα από το $\mathbf{p_1}$ στο $\mathbf{p_2}$ είναι **εισερχόμενο** ή **εξερχόμενο** σε σχέση με μια ευθεία του **επεκταμένου** παραθύρου αποκοπής
  - ανάλογα με το αν η **κατεύθυνσή** του είναι από το εσωτερικό στο εξωτερικό ημιεπίπεδό της ή αντίστροφα
  - π.χ., τα $\mathbf{p_1p_2}$, $\mathbf{p_1''p_2''}$ εισερχόμενα σε σχέση με την ευθεία 1, το $\mathbf{p_1'p_2'}$ εξερχόμενο

Αλγόριθμος Liang-Barsky



- Για να είναι ένα σημείο ευθυγράμμου τμήματος $\mathbf{p}_1\mathbf{p}_2$ **εντός** του παραθύρου αποκοπής, πρέπει να βρίσκεται στο **εσωτερικό** ημι-επίπεδο **κάθε** ευθείας του παραθύρου
- "Ταξιδεύοντας" πάνω στην ευθεία από το $\mathbf{p}_1$ στο $\mathbf{p}_2$, **δεν** πρέπει να εξέλθουμε σε σχέση με κάποια ευθεία του παραθύρου, **πριν** εισέλθουμε σε κάποια άλλη
  - π.χ., η ακολουθία τομών [είσοδος, είσοδος, έξοδος, έξοδος] υποδεικνύει **τομή** με το παράθυρο αποκοπής
  - η ακολουθία [είσοδος, έξοδος, είσοδος, έξοδος] **δεν** υποδεικνύει

Αλγόριθμος Liang-Barsky

- Αν ένα ευθύγραμμο τμήμα **τέμνει** το παράθυρο αποκοπής, θα **εισέλθει** σε αυτό στην τομή του με μια ευθεία του παραθύρου ως προς την οποία είναι **εισερχόμενο**
- Θα **εξέλθει** στην τομή του με μια ευθεία του παραθύρου ως προς την οποία είναι **εξερχόμενο**
- **Εξαίρεση** αποτελούν **άκρα** του ευθύγραμμου τμήματος που βρίσκονται **εντός** του παραθύρου
- Ο αλγόριθμος Liang-Barsky υπολογίζει τη **μέγιστη** παραμετρική τιμή t_{in} των εισερχόμενων τομών και την **ελάχιστη** παραμετρική τιμή t_{out} των εξερχόμενων
- Στη συνέχεια, ελέγχει αν αυτές οι παραμετρικές τιμές αντιστοιχούν σε σημεία **πάνω** στο ευθύγραμμο τμήμα
  - αν είναι μέσα στο $[0, 1]$ – αν όχι, **αντικαθίστανται** από το 0 ή το 1, αντίστοιχα
  - αν $t_{in} \leq t_{out}$, υπάρχει **τομή** του ευθύγραμμου τμήματος και του παραθύρου αποκοπής, διαφορετικά, όχι

Αλγόριθμος Liang-Barsky

Η θεωρία του αλγορίθμου είναι η ακόλουθη:

- Για **δοσμένο** παράθυρο αποκοπής, θα ισχύει

$$x_{min} \leq x_1 + t\Delta x \leq x_{max}, \quad y_{min} \leq y_1 + t\Delta y \leq y_{max}$$

ή

$$-t\Delta x \leq x_1 - x_{min}, \quad -t\Delta y \leq y_1 - y_{min}$$

$$t\Delta x \leq x_{max} - x_1, \quad t\Delta y \leq y_{max} - y_1$$

- Οι προηγούμενες ανισότητες μπορούν να τεθούν στην **κοινή** μορφή $t \cdot p_i \leq q_i, i = 1, 2, 3, 4$, όπου
  - $p_1 = -\Delta x, \quad q_1 = x_1 - x_{min}$
  - $p_2 = \Delta x, \quad q_2 = x_{max} - x_i$
  - $p_3 = -\Delta y, \quad q_3 = y_1 - y_{min}$
  - $p_4 = \Delta y, \quad q_4 = y_{max} - y_1$

Αλγόριθμος Liang-Barsky

- Κάθε μία ανισότητα δίνει την **περιοχή τιμών** του t η οποία αντιστοιχεί στο **ορατό** μέρος του ευθύγραμμου τμήματος αναφορικά με την αντίστοιχη ακμή
- Εάν $p_i \neq 0$, τότε το **σημείο τομής** της ευθείας που ορίζουν τα $\mathbf{p}_1, \mathbf{p}_2$ με την i πλευρά του παραθύρου δίνεται για $t = \frac{q_i}{p_i}, \quad i = 1, 2, 3, 4$
- Το πρόσημο του q_i καθορίζει το **ημιεπίπεδο** της πλευράς i στο οποίο βρίσκεται το $\bar{p}_1$
  - $q_i \geq 0 \Rightarrow \mathbf{p}_1$ βρίσκεται στο **ορατό** ημιεπίπεδο
  - $q_i < 0 \Rightarrow \mathbf{p}_1$ βρίσκεται στο **μη ορατό** ημιεπίπεδο
- Εάν $p_i = 0$, τότε η ευθεία είναι **παράλληλη** στην ακμή $i, \quad = 1, 2, 3, 4$
- Εάν $p_i < 0$, τότε η προσανατολισμένη ευθεία $\mathbf{p}_1\mathbf{p}_2$ μεταβαίνει **από το μη ορατό προς το ορατό** ημιεπίπεδο της πλευράς i
- Εάν $p_i > 0$, τότε η προσανατολισμένη ευθεία $\mathbf{p}_1\mathbf{p}_2$ μεταβαίνει **από το ορατό προς το μη ορατό** ημιεπίπεδο της πλευράς i

Αλγόριθμος Liang-Barsky

- Ο αλγόριθμος υπολογίζει δύο τιμές t_1 και t_2 της παραμέτρου t, οι οποίες καθορίζουν τα δύο άκρα του τμήματος του P_1P_2 που βρίσκεται **μέσα** στο παράθυρο

$$t_1 = \max(\{\frac{q_i}{p_i} | p_i < 0, i = 1, 2, 3, 4\} \cup \{0\}) \quad (1)$$

και

$$t_2 = \min(\{\frac{q_i}{p_i} | p_i > 0, i = 1, 2, 3, 4\} \cup \{1\}) \quad (2)$$

- Τα μονοσύνολα $\{0\}, \{1\}$ εξασφαλίζουν την επιλογή των άκρων $\bar{P}_1, \bar{P}_2$ στην περίπτωση που τα σημεία τομής της ευθείας με τις πλευρές του παραθύρου βρίσκονται **εκτός** του $\bar{P}_1, \bar{P}_2$
- Εάν $t_1 > t_2$, τότε το ευθύγραμμο τμήμα βρίσκεται **εκτός** του παραθύρου, διαφορετικά οι τιμές t_1, t_2 χρησιμοποιούνται για τον υπολογισμό των σημείων τομής

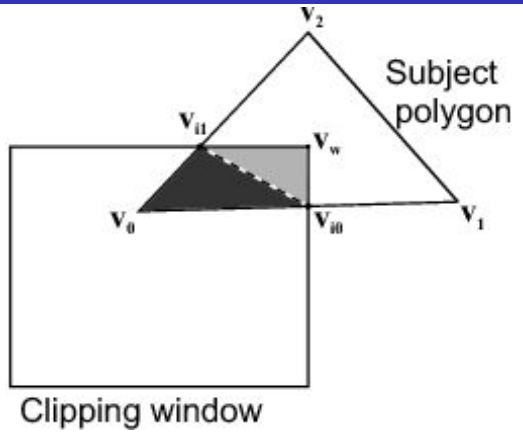
Αποκοπή Πολυγώνου

- Η **αποκοπή** 2Δ πολυγώνων είναι βασική πράξη στα γραφικά
- Οι χρήσεις της περιλαμβάνουν
  - Αλγόριθμους **αντιτάυτισης** (αποκοπή στα όρια του **εικονοστοιχείου**)
  - Αλγόριθμους **απόκρυψης**
  - Χωρικό **διαχωρισμό** πολυγώνων σε συστήματα πολυεπεξεργασίας
- Οι 3Δ αλγόριθμοι αποκοπής είναι συνήθως **γενικεύσεις** των 2Δ αλγόριθμων αποκοπής
- Ο αλγόριθμος Sutherland-Hodgman είναι κατάλληλος για την αποκοπή οποιουδήποτε πολυγώνου με **κυρτό** πολύγωνο αποκοπής
- Ο αλγόριθμος Greiner-Hormann μπορεί να εφαρμοστεί για **γενικευμένα** πολύγωνα

Αποκοπή Πολυγώνου

- Στη 2Δ αποκοπή πολυγώνου, τόσο το αντικείμενο αποκοπής, όσο και αυτό που αποκόπτεται είναι **πολύγωνα**
- Το αντικείμενο αποκοπής μερικές φορές περιορίζεται να είναι **κυρτό** ή ακόμα και ορθογώνιο παραλληλόγραμμο (**παράθυρο**)
- Γιατί **χρειαζόμαστε** αλγορίθμους αποκοπής;
- Δεν θα μπορούσαμε να θεωρήσουμε ένα πολύγωνο ως ένα σύνολο ευθύγραμμων τμημάτων (**ακμών**), τα οποία να αποκόψουμε με το πολύγωνο αποκοπής;
- Αν κόψουμε απλά ένα πολύγωνο ως σύνολο ευθύγραμμων τμημάτων μπορεί να οδηγηθούμε σε **λάθος** αποτέλεσμα

Χρησιμότητα αλγορίθμων αποκοπής

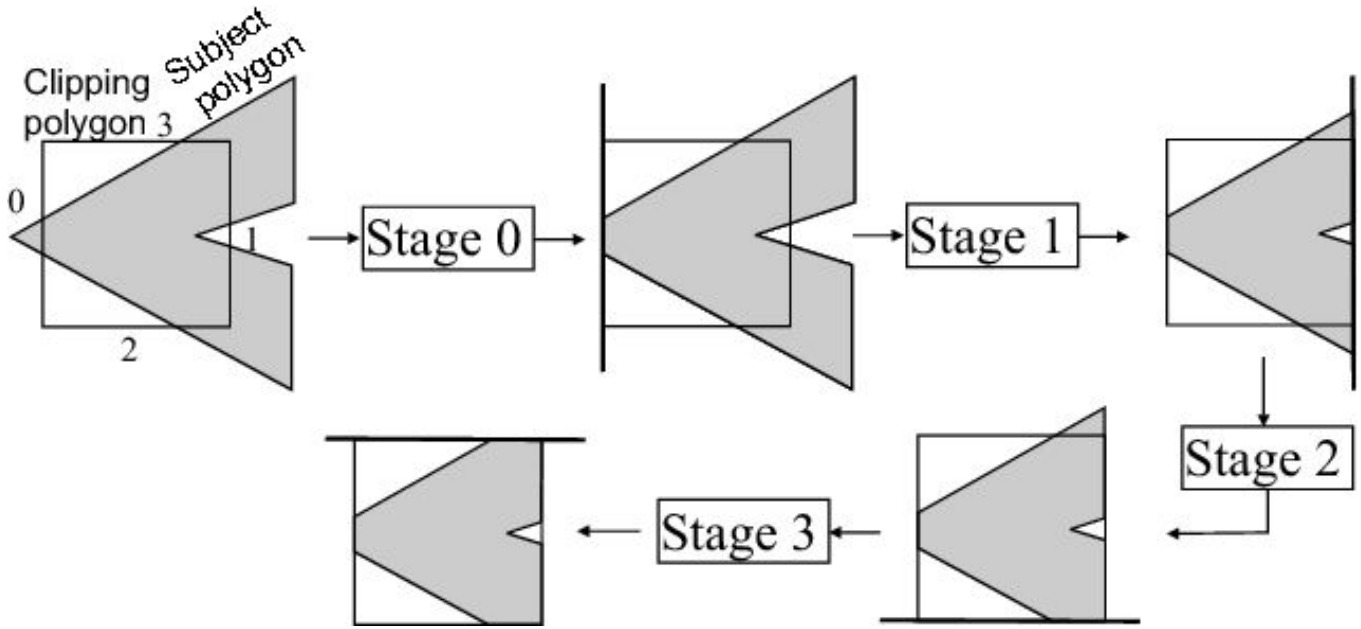


- Τα **αποτελέσματα** της αποκοπής των ακμών του τριγώνου $\mathbf{v}_0\mathbf{v}_1\mathbf{v}_2$ με το πολύγωνο αποκοπής είναι τα ευθύγραμμα τμήματα $\mathbf{v}_0\mathbf{v}_{i0}, \mathbf{v}_0\mathbf{v}_{i1}$
- Αυτά δεν παριστάνουν **κλειστό** πολύγωνο!
- Ακόμη κι αν προσθέσουμε το $\mathbf{v}_{i0}\mathbf{v}_{i1}$, το πολύγωνο **κλείνει**, αλλά και πάλι είναι **λάθος**!
- Το **σωστό** αποτέλεσμα θα ήταν το $\mathbf{v}_0\mathbf{v}_{i0}\mathbf{v}_w\mathbf{v}_{i1}$
- Τα πολύγωνα πρέπει να αντιμετωπίζονται ως **επιφάνειες** και όχι ως σύνολα ευθύγραμμων τμημάτων

Αλγόριθμος Sutherland-Hodgman

- Ο αλγόριθμος Sutherland-Hodgman μπορεί να αποκόψει ένα **κυρτό ή μη κυρτό** (άρα τυχαίο) πολύγωνο ως προς ένα **κυρτό** πολύγωνο (παράθυρο) αποκοπής
- Αν το παράθυρο έχει m **πλευρές**, ο αλγόριθμος αποτελείται από m **βήματα**
- Το βήμα i, $i = 1, 2, \ldots, m$, αφορά την αποκοπή του πολυγώνου ως προς την **ευθεία** που περιέχει την πλευρά i του παραθύρου
- Η είσοδος στο 1ο βήμα είναι το **αρχικό** πολύγωνο
- Η είσοδος στο i-οστό βήμα είναι το πολύγωνο που προκύπτει **μετά** την αποκοπή στο i-1-οστό βήμα
- Ουσιαστικά υπολογίζει την **τομή** της επιφάνειας του αποκοπτόμενου πολυγώνου με το εσωτερικό ημιεπίπεδο της ευθείας αποκοπής i

Αλγόριθμος Sutherland-Hodgman

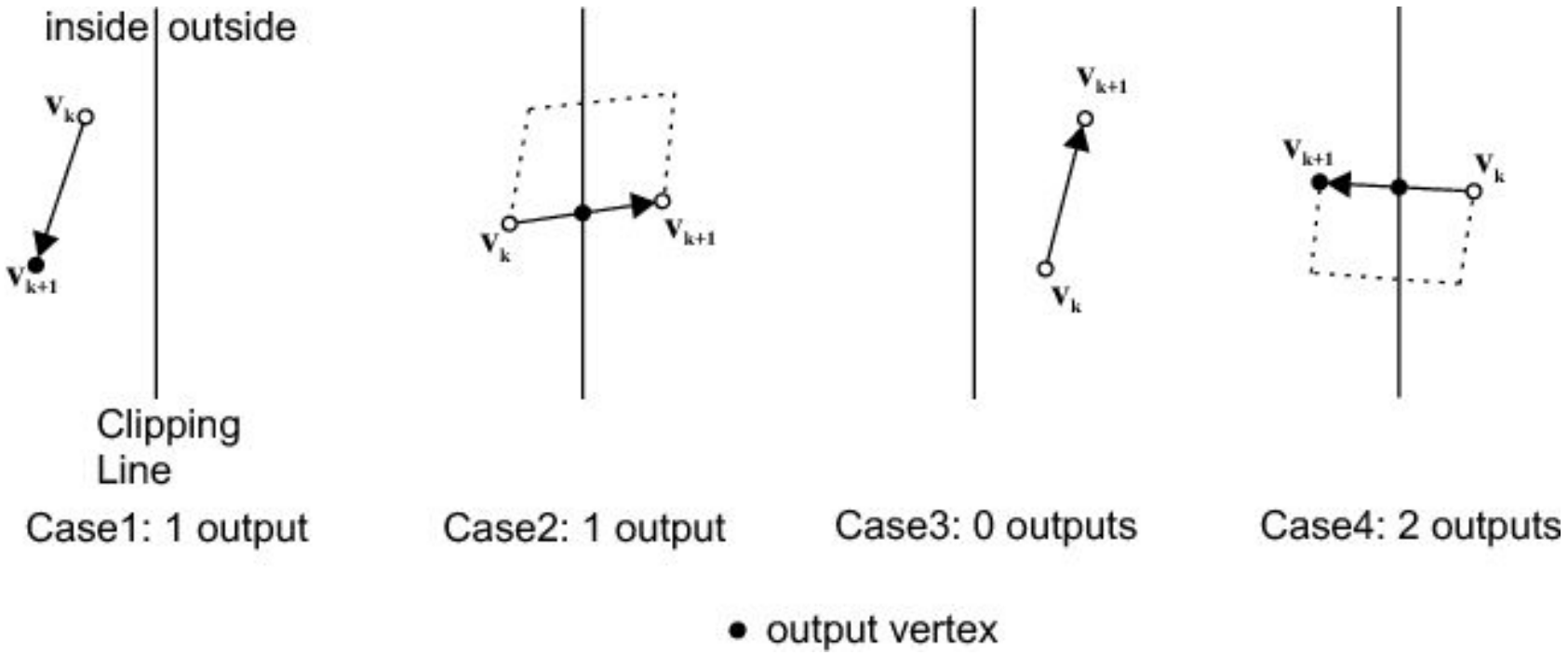


Βήματα εφαρμογής του αλγορίθμου **Sutherland-Hodgman**

Αλγόριθμος Sutherland-Hodgman

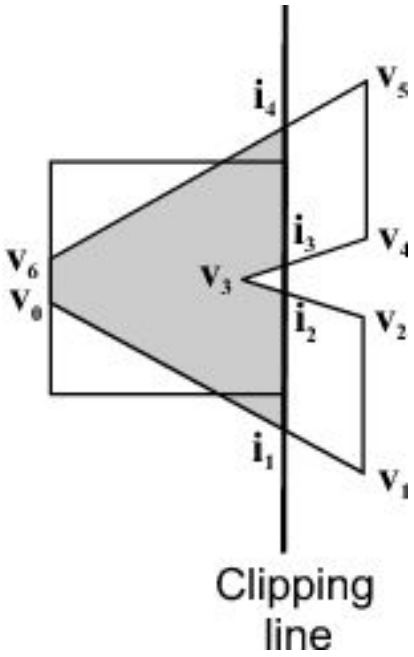
- Το πολύγωνο αρχικά ορίζεται από τις **κορυφές** του $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n-1}$
- Θεωρείται ότι η κίνηση κατά μήκος της περιμέτρου του πολυγώνου γίνεται με **θετική** φορά
- Οι κορυφές ορίζουν τις **πλευρές** $\mathbf{v}_0\mathbf{v}_1$, $\mathbf{v}_1\mathbf{v}_2$, …, $\mathbf{v}_{n-2}\mathbf{v}_{n-1}$, $\mathbf{v}_{n-1}\mathbf{v}_0$
- Για κάθε πλευρά του πολυγώνου το οποίο προέκυψε από το προηγούμενο βήμα αποκοπής, **προστίθενται** στη λίστα κορυφών του νέου πολυγώνου **0,1 ή 2 κορυφές**, ανάλογα με τη **θέση** της, σε σχέση με την **ευθεία** αποκοπής

Αλγόριθμος Sutherland-Hodgman



Περιπτώσεις αποκοπής με την εφαρμογή του αλγορίθμου **Sutherland-Hodgman**

Βήμα Αποκοπής Αλγορίθμου Sutherland-Hodgman



$\mathbf{v}_k$	$\mathbf{v}_{k+1}$	Περίπτωση	Έξοδος
$\mathbf{v}_0$	$\mathbf{v}_1$	2	$\mathbf{i}_1$
$\mathbf{v}_1$	$\mathbf{v}_2$	3	-
$\mathbf{v}_2$	$\mathbf{v}_3$	4	$\mathbf{i}_2, \mathbf{v}_3$
$\mathbf{v}_3$	$\mathbf{v}_4$	2	$\mathbf{i}_3$
$\mathbf{v}_4$	$\mathbf{v}_5$	3	-
$\mathbf{v}_5$	$\mathbf{v}_6$	4	$\mathbf{i}_4, \mathbf{v}_6$
$\mathbf{v}_6$	$\mathbf{v}_0$	1	$\mathbf{v}_0$

- 1ο βήμα του αλγορίθμου Sutherland-Hodgman για το προηγούμενο παράδειγμα
- Κάθε ακμή $\mathbf{v}_k\mathbf{v}_{k+1}$ του πολυγώνου εισόδου θεωρείται σε σχέση με την ευθεία αποκοπής

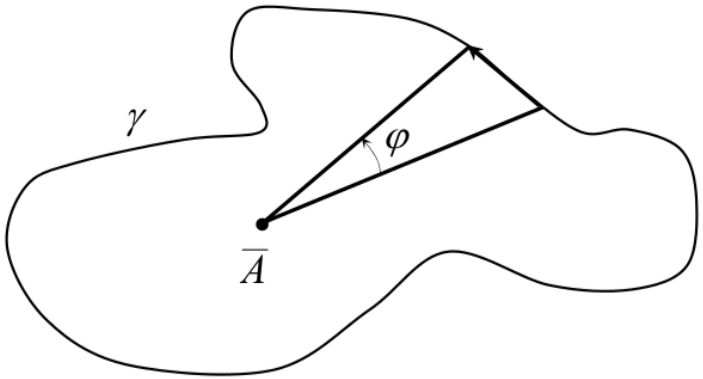
Αλγόριθμος Greiner-Hormann

- Ο αλγόριθμος των Greiner-Hormann είναι κατάλληλος για **γενικά** πολύγωνα
- Το πολύγωνο **αποκοπής** C και το πολύγωνο **προς αποκοπή** S μπορεί να είναι οποιαδήποτε **κλειστά** πολύγωνα (ακόμη και μη κυρτά, αυτοτεμνόμενα, κτλ.)
- Ο αλγόριθμος θεωρεί τα πολύγωνα **συμμετρικά** (δεν έχει σημασία ποιο είναι ποιο – υπολογίζεται η τομή των επιφανειών τους)
- Η απόφαση του εάν ένα σημείο είναι εντός ή εκτός του πολυγώνου είναι **έυκολη** στην περίπτωση που το πολύγωνο είναι κυρτό (Greiner-Hodgman)
- Για τη γενικευμένη περίπτωση, ο αλγόριθμος χρησιμοποιεί την έννοια του **αριθμού περιελίξεων**

Αριθμός περιελίξεων

- Για μια κλειστή καμπύλη γ και ένα σημείο $\mathbf{a}$, ο **αριθμός περιελίξεων** $\omega(\gamma, \mathbf{a})$ μετρά πόσες στροφές ολοκληρώνει μια ακτίνα που ξεκινά από το $\mathbf{a}$ και διαγράφει την περιφέρεια της καμπύλης γ μια φορά
- Για κάθε ολοκληρωμένη στροφή με τη **φορά** των δεικτών του ρολογιού, **μειώνεται** κατά 1
- Για κάθε ολοκληρωμένη στροφή με **αντίθετη φορά** από αυτή των δεικτών του ρολογιού, **αυξάνεται** κατά 1
- Ορίζεται ως $\omega(\gamma, \mathbf{a}) = \frac{1}{2\pi} \int d\phi$

Αριθμός περιελίξεων



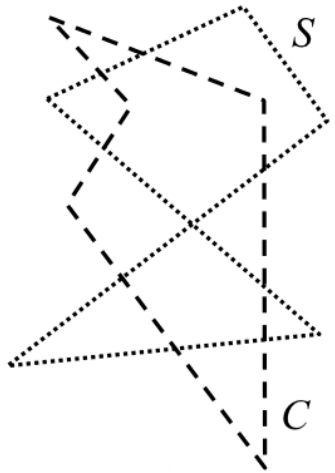
- Ο αριθμός περιελίξεων δεν αλλάζει εφόσον δεν μεταβάλλεται η **τοπολογική** σχέση του $\mathbf{a}$ με την γ
- Αν το $\mathbf{a}$ είναι **εκτός** της καμπύλης, τότε $\omega(\gamma, \mathbf{a}) = 0$
- Αν το $\mathbf{a}$ μετακινηθεί και διασταυρώσει μία φορά την γ, τότε το $\omega(\gamma, \mathbf{a})$ **αυξάνεται** ή **μειώνεται** κατά 1
- Αν το $\mathbf{a}$ βρίσκεται εντός της γ, τότε το $\omega(\gamma, \mathbf{a})$ είναι **περιττός**, αλλιώς είναι **άρτιος**

Δείκτης Περιστροφών

- Από τα παραπάνω μπορεί να εξαχθεί το συμπέρασμα ότι κάθε φορά που το μονοπάτι του $\mathbf{a}$ τέμνει την γ, περνά είτε από το **εξωτερικό στο εσωτερικό** της, είτε **αντίστροφα**
- Αυτό είναι αντίστοιχο με το μέτρημα των **τομών** της γ με την ημιευθεία από το $\mathbf{a}$ προς το **άπειρο**

Αλγόριθμος Greiner-Hormann - Παράδειγμα

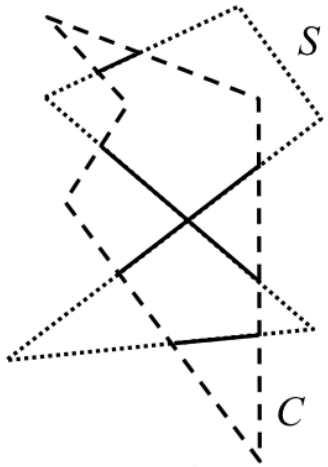
- Έστω το πολύγωνο αποκοπής C και το πολύγωνο προς αποκοπή S



- Το C είναι μη κυρτό, το S αυτοτεμνόμενο

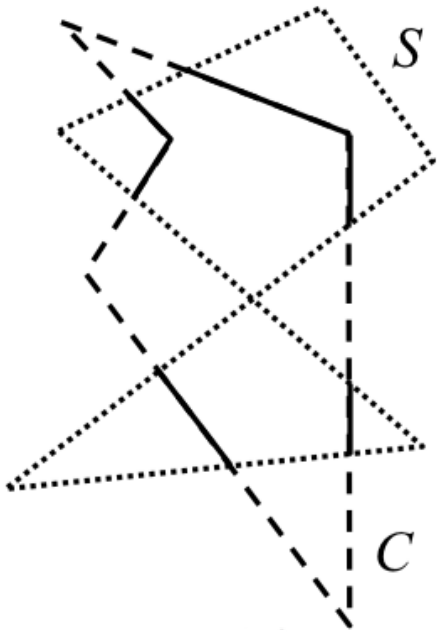
Αλγόριθμος Greiner-Hormann - Βήμα 1

- Εκκίνηση από μια κορυφή του S – παρακολουθείται η **περίμετρος** του
- Χρησιμοποιείται μια φανταστική **γραφίδα** (ενεργή ή μη ενεργή)
- Αρχικά, είναι ενεργή αν η κορυφή είναι εντός του C, μη ενεργή αν είναι εκτός του C
- Κάθε φορά που διασταυρώνεται με το C, **αλλάζει κατάσταση**
- **Διαγράφονται** τα τμήματα του S μέσα στο C



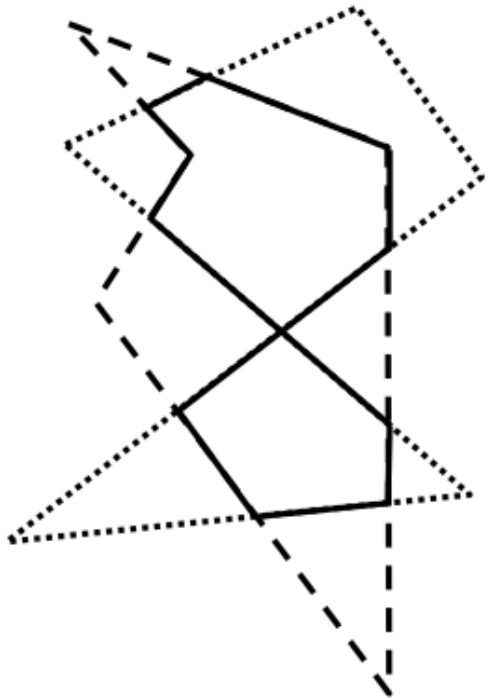
Αλγόριθμος Greiner-Hormann - Βήμα 2

- Το **ίδιο** με το βήμα 1, αλλάζει μόνο ο **ρόλος** των C, S
- Δημιουργούνται τα τμήματα του C που βρίσκονται **μέσα** στο S



Αλγόριθμος Greiner-Hormann - Βήμα 3

- **Ενώνονται** όλα τα τμήματα που προέκυψαν από τα βήματα 1 και 2 S



Αλγόριθμος Greiner-Hormann

- Το αποτέλεσμα της αποκοπής μπορεί να αποτελείται από 1 ή περισσότερα **μη συνεκτικά** (ασύνδετα) πολύγωνα
- Η αποκοπή ουσιαστικά βρίσκει την **τομή** των επιφανειών C,S ($C \cap S$)
- Ο αλγόριθμος Greiner-Hormann εύκολα **μετατρέπεται** ώστε να υπολογίζει τα $C \cup S, C - S, S - C$
  - αυτό γίνεται αλλάζοντας τις **αρχικές** καταστάσεις των γραφίδων για τα S, C (υπάρχουν 4 δυνατοί συνδυασμοί αρχικών καταστάσεων)