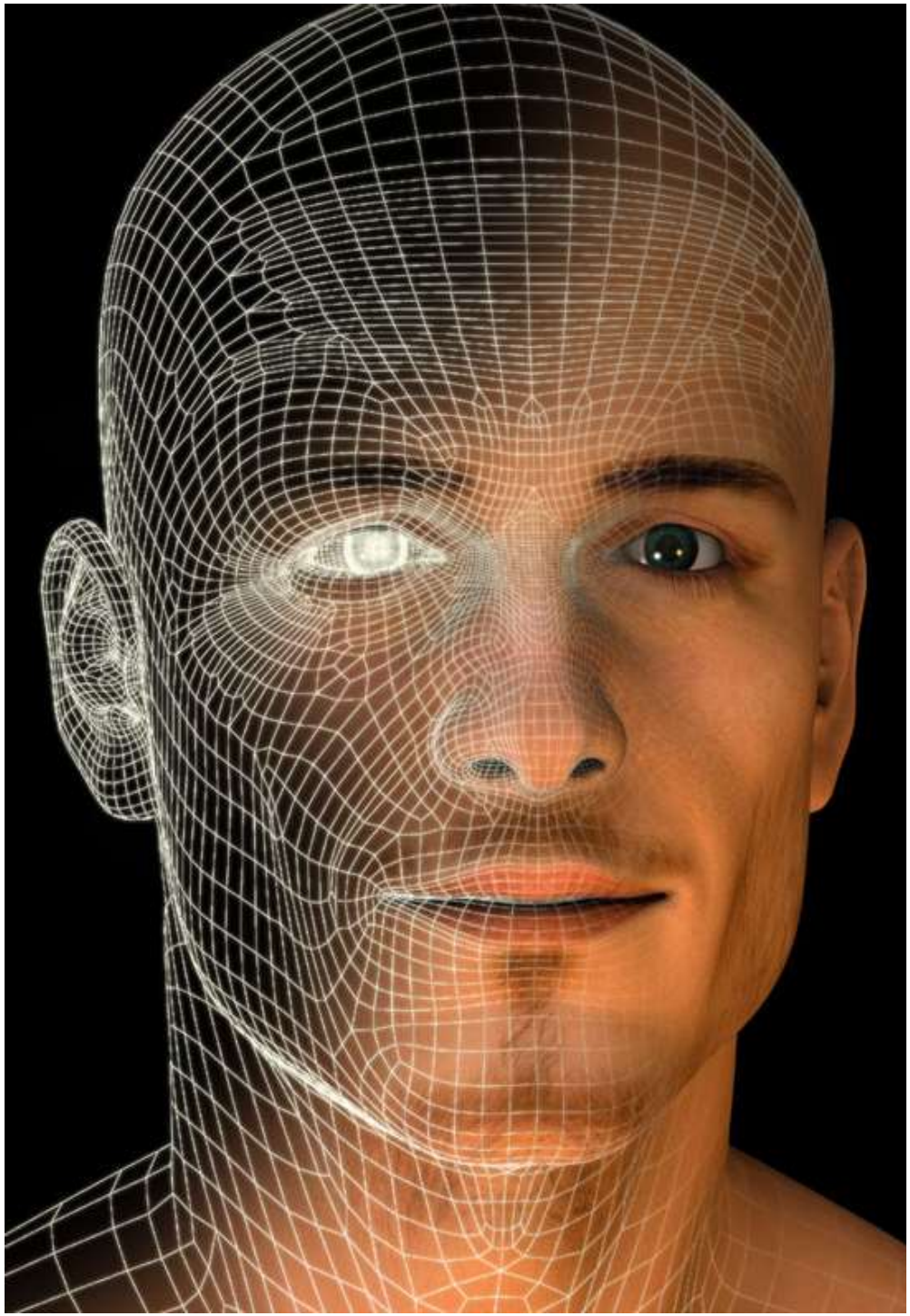


Τμήμα Μηχανικών Πληροφορικής

ΤΕΙ Ανατολικής Μακεδονίας και Θράκης

Γραφικά Υπολογιστών
ΣΤ’ Εξάμηνο

Δρ Κωνσταντίνος Δεμερτζής

ΤΕΙ Ανατολικής Μακεδονίας και Θράκης
Τμήμα Μηχανικών Πληροφορικής

Γραφικά Υπολογιστών

5η Ενότητα

Σχεδίαση Πολυγώνων και Καμπυλών

ΤΕΙ Ανατολικής Μακεδονίας και Θράκης
Τμήμα Μηχανικών Πληροφορικής

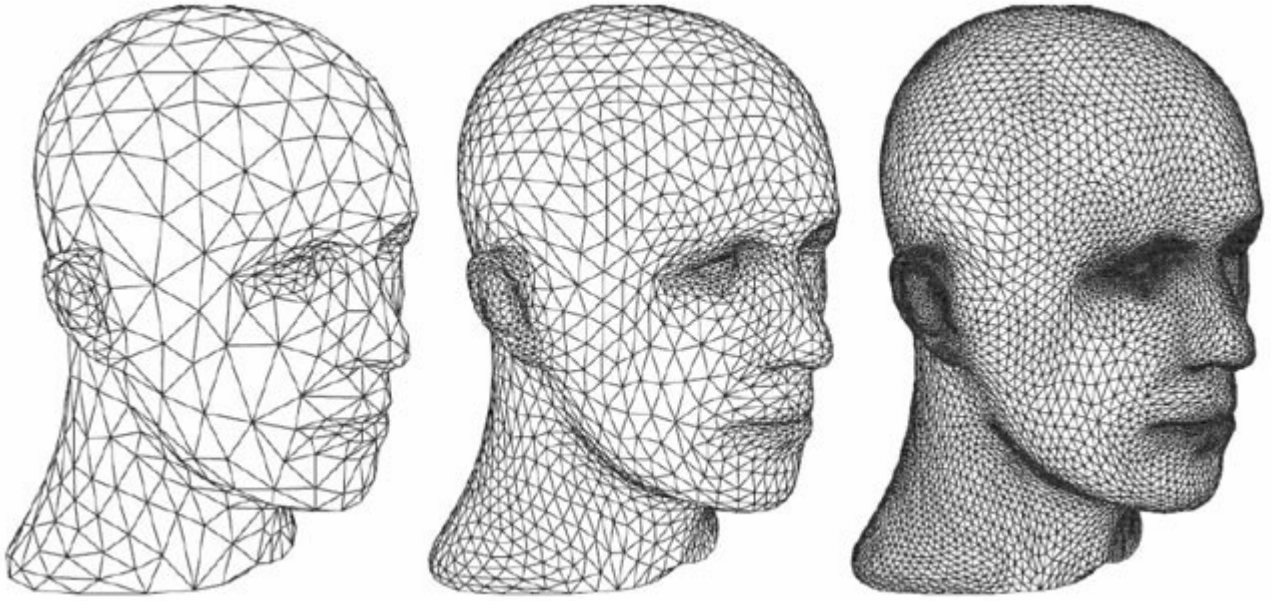
Γραφικά Υπολογιστών

kdemertz@fmenr.duth.gr

Γραφικά Υπολογιστών

Σχεδίαση Πολυγώνων

- ✓ *Τα ευθύγραμμα τμήματα δεν αποτελούν τα μόνα σχήματα που καλούνται να αποτυπωθούν πάνω σε πλεγματικές οθόνες.*
- ✓ *Τα περισσότερα από τα τελικά σχήματα που σχεδιάζονται, είναι σύνθετα και αποτελούνται μεταξύ άλλων και από πολύγωνα.*
- ✓ *Παρακάτω εξετάζονται τα πιο συνήθη πολύγωνα και κάποιοι ενδεικτικοί αλγόριθμοι σχεδίασης.*



Γραφικά Υπολογιστών

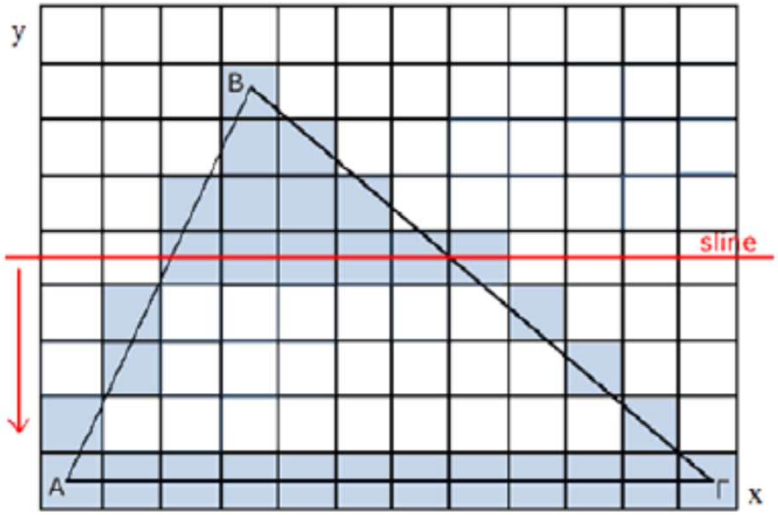
Αλγόριθμοι Σχεδίασης Τριγώνου

- ✓ *Πολύγραμμα είναι τα σχήματα που ορίζονται από μια σειρά διαδοχικών ευθύγραμμων τμημάτων. Τα πολύγραμμα σχεδιάζονται καλώντας επαναληπτικά τον αλγόριθμο σχεδίασης ευθύγραμμου τμήματος. Το κλειστό πολύγραμμο αποτελεί πολύγωνο.*
- ✓ *Το πιο απλό πολύγωνο είναι το τρίγωνο και αποτελείται από τρία ευθύγραμμα τμήματα που τέμνονται στις άκρες τους σχηματίζοντας ένα κλειστό σχήμα. Το τρίγωνο είναι πολύ σημαντικό σχήμα για το χώρο των γραφικών καθώς πολλές διαδικασίες εξετάζουν τις επιφάνειες ως τριγωνικά πλέγματα (τριγωνοποίηση).*
- ✓ *Ως επίπεδα σχήματα, τα πολύγωνα ορίζονται με βάση τις κορυφές τους οι οποίες όταν ενώνονται μεταξύ τους με ακμές ορίζουν:*
  - *α) ένα τριγωνικό πλέγμα (τα σημεία του σχήματος ανήκουν στις ακμές),*
  - *β) μια τριγωνική επιφάνεια (χρειάζεται ένας έλεγχος εσωτερικότητας των σημείων για να διαπιστωθεί αν αυτά ανήκουν ή όχι στην επιφάνεια που ορίζεται από τις ακμές).*

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου

- ✓ *Ο αλγόριθμος του Bresenham, ως ένας αποδοτικός αλγόριθμος σχεδίασης ευθύγραμμου τμήματος χρησιμοποιείται τρεις φορές, για να σχεδιαστούν οι ακμές του τριγώνου. Ο συγχρονισμός των τριών εκτελέσεων του αλγορίθμου επιτρέπει τη σχεδίαση ανά γραμμή σάρωσης. Για κάθε θέση της γραμμής σάρωσης υπολογίζεται εκ νέου το ευθύγραμμο τμήμα που περιέχει τα εσωτερικά σημεία του τριγώνου και στη συνέχεια χρωματίζονται αυτά τα σημεία αποκαλύπτοντας σταδιακά το επιθυμητό σχήμα.*



Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου

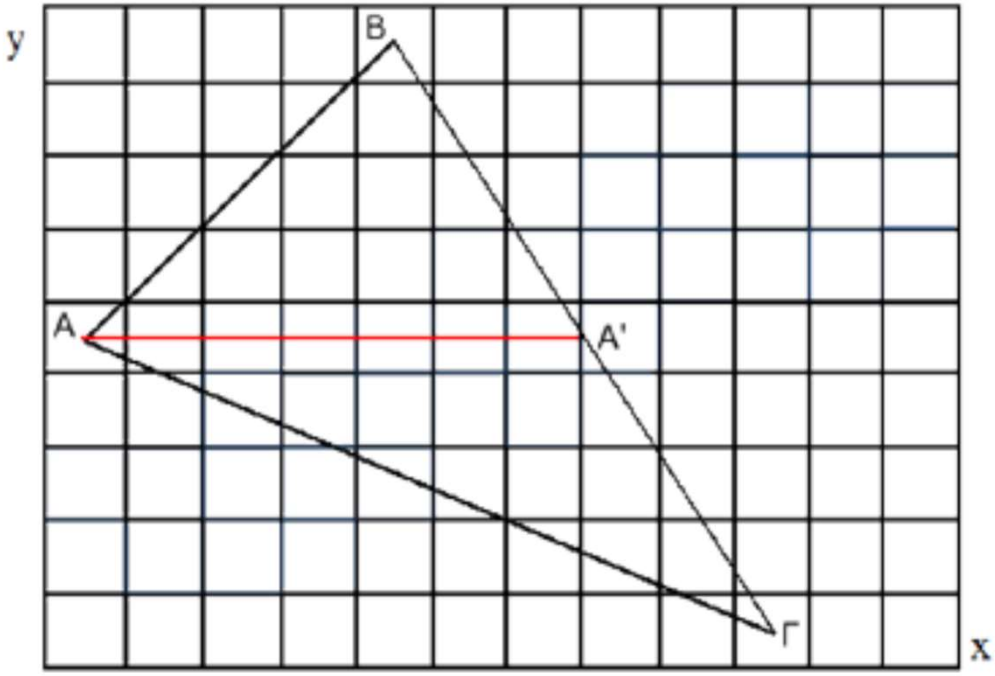
- ✓ *Στην αρχή τοποθετούνται οι δύο κορυφές πάνω στον οριζόντιο άξονα και υπολογίζονται οι κλίσεις των ευθύγραμμων τμημάτων από την άνω κορυφή προς τη βάση του τριγώνου. Σε κάθε νέα θέση η γραμμή σάρωσης υπολογίζει με βάση τις κλίσεις slope1 και slope2, τις νέες θέσεις των σημείων που ορίζουν την εσωτερική γραμμή. Η drawline (π.χ. Bresenham) είναι η συνάρτηση που καλείται, για να χρωματίσει τα σημεία της γραμμής σάρωσης από το x1 έως το x2.*
- ✓ *Ο αλγόριθμος που υλοποιεί αυτήν την προσέγγιση παρατίθεται παρακάτω:*

```
1. vertex v1, v2, v3;
2. float slope1 = (v2.x - v1.x) / (v2.y - v1.y);
3. float slope2 = (v3.x - v1.x) / (v3.y - v1.y);
4. float x1 = v1.x;
5. float x2 = v1.x;
6. for (int slineY = v1.y; sline <= v2.y; sline++){
7. drawline((int)x1, sline, (int)x2, sline);
8. x1+ = slope1;
9. x2+ = slope2;
10. }
11.}
```

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου

✓ *Το παραπάνω παράδειγμα παρουσίαζε τη βολική περίπτωση οι δύο κορυφές του τριγώνου να είναι παράλληλα τοποθετημένες πάνω σε μια οριζόντια ευθεία. Στη γενική περίπτωση, ένα τρίγωνο είναι τοποθετημένο στο 2D επίπεδο, όπως φαίνεται στην Εικόνα παρακάτω:*



Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου

- ✓ *Η λύση είναι να χωριστεί το τρίγωνο σε δύο μικρότερα τρίγωνα με βάση την οριζόντια γραμμή που διέρχεται από την ενδιάμεση ως προς το ύψος (άξονας y) κορυφή. Το πάνω τρίγωνο σχεδιάζεται με τη βοήθεια του αλγορίθμου που παρουσιάστηκε προηγουμένως, ενώ το κάτω τρίγωνο σχεδιάζεται με μια παραλλαγή του αλγορίθμου που θεωρεί ότι οι πάνω ακμές βρίσκονται στο ίδιο ύψος. Το ποιος αλγόριθμος θα εκτελεστεί κάθε φορά καθορίζεται από τον έλεγχο του τεμαχισμού όπως φαίνεται παρακάτω:*

```
1. sortVerticesByY();// Ταξινόμηση των κορυφών ως προς y
2. if (v2.y == v3.y){
3. fillUpRightTriangle(v1, v2, v3); }
4. else if (vt1.y == vt2.y){
5. fillReverseTriangle(v1, v2, v3);
6. } else {
7. Vertice v4 = new Vertice((int)(vt1.x +
8. ((float)(vt2.y - vt1.y) / (float)(vt3.y - vt1.y))
9.* (vt3.x - vt1.x)), vt2.y);
10.fillUpRightTriangle(g, vt1, vt2, v4);
11.fillReverseTriangle(g, vt2, v4, vt3);
12. } }
```

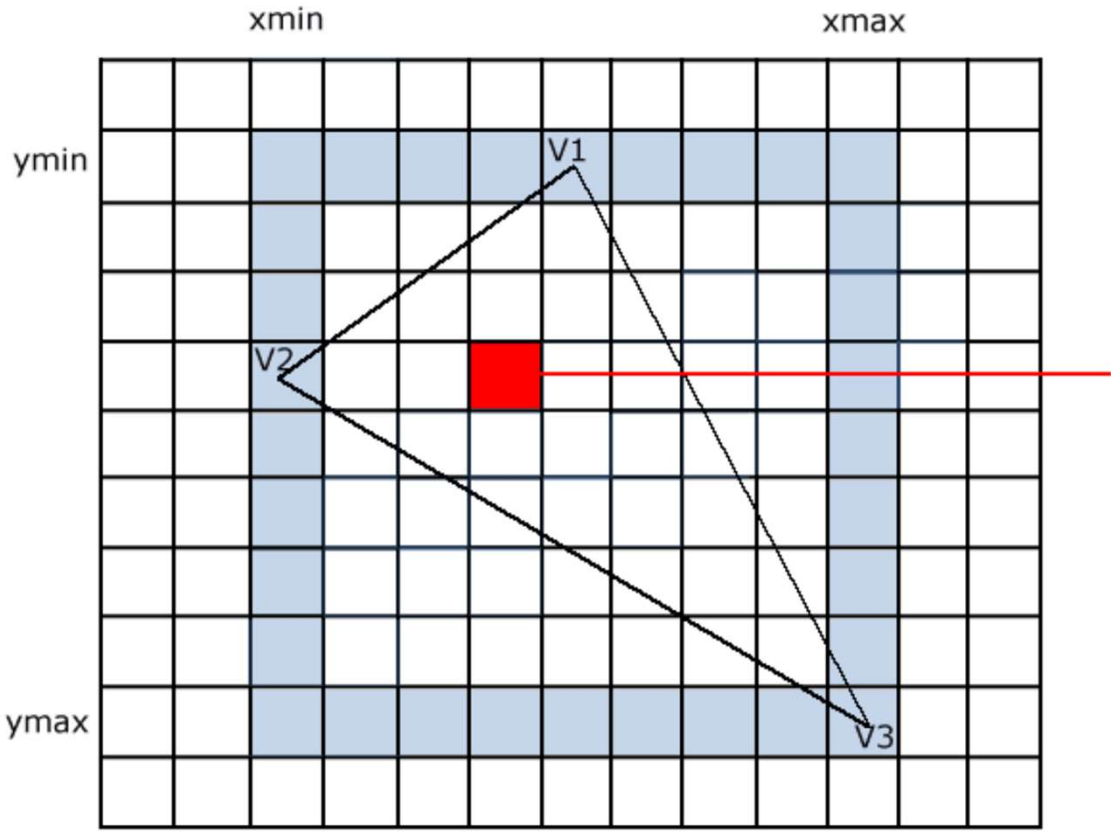
Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου

- ✓ *Στην αρχή θεωρείται μια λίστα με τις κορυφές του τριγώνου ταξινομημένες ως προς τον άξονα Υ. Στη συνέχεια γίνονται μια σειρά έλεγχοι για να διαπιστωθεί αν το τρίγωνο ανήκει ή όχι σε μια από τις βολικές περιπτώσεις (όρθιο ή ανάστροφο τρίγωνο). Αν όχι, τότε τεμαχίζει το τρίγωνο και καλεί τους δύο κατάλληλους αλγορίθμους.*
- ✓ *Σύμφωνα με μια άλλη προσέγγιση, κάθε τρίγωνο περικλείεται από ένα ορθογώνιο παραλληλόγραμμο του οποίου τα εικονοστοιχεία ελέγχονται αν ανήκουν ή όχι στο τρίγωνο. Με αυτήν την απλή σχετικά τεχνική σχεδιάζεται ένα τρίγωνο χωρίς ταξινόμηση των κορφών του και χωρίς κατάτμηση. Ο έλεγχος εσωτερικότητας των εικονοστοιχείων που περικλείονται από το ορθογώνιο παραλληλόγραμμο καθορίζει ποια εικονοστοιχεία θα χρωματιστούν και ποια όχι.*

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου



Ανάκτηση πλαισίου οριοθέτησης του τριγώνου και έλεγχος εσωτερικότητας των εικονοστοιχείων

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Τριγώνου

```
1. int maxX = max(v1.x, v2.x, v3.x);
2. int minX = min(v1.x, v2.x, v3.x);
3. int maxY = max(v1.y, v2.y, v3.y);
4. int minY = min(v1.y, v2.y, v3.y);
5. for (int x = minX; x <= maxX; x++){
6. for (int y = minY; y <= maxY; y++){
7. if (checkInternal(x,y)){
8. drawPixel(x, y);
9. }
10.}
11.}
```

Ο κώδικας σχεδίασης τριγώνου με έλεγχο εσωτερικότητας των εικονοστοιχείων

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Ο Αλγόριθμος Σάρωσης Πολυγώνου (Scanline polygon fill algorithm) χρησιμοποιείται για να γεμίσει το εσωτερικό ενός πολυγώνου.*
- ✓ *Η προβληματική αναφέρεται στο ποια εικονοστοιχεία (με βάση τις συντεταγμένες του κέντρου τους) πρέπει να χρωματιστούν για να προκύψει η επιφάνεια που ορίζεται από το πολύγωνο, ή η γραμμή της περιμέτρου στην περίπτωση που δε μας ενδιαφέρουν τα εσωτερικά σημεία.*
- ✓ *Η σάρωση πολυγώνου βασίζεται στον έλεγχο των σημείων τομής μεταξύ των ακμών του πολυγώνου και της γραμμής σάρωσης. Θεωρούμε γραμμή σάρωσης εκείνη που σαρώνει το πολύγωνο από τη χαμηλότερη θέση (κατά y) έως την υψηλότερη.*

Γραφικά Υπολογιστών

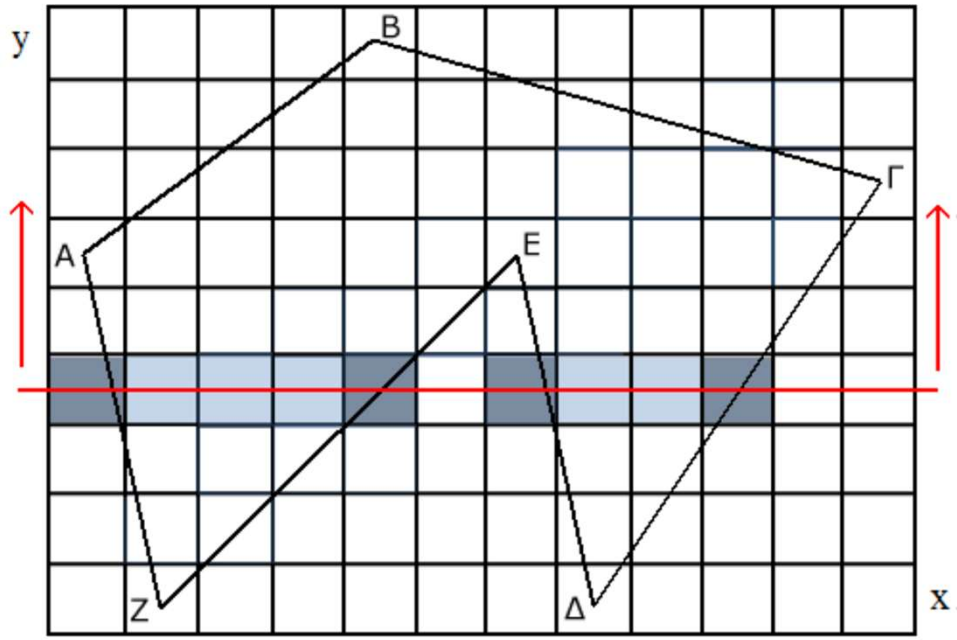
Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Ο αλγόριθμος σάρωσης πολυγώνου ακολουθεί τα παρακάτω βήματα:*
  ***Βήμα 1**: Για κάθε θέση της γραμμής σάρωσης βρες τα σημεία τομής της γραμμής σάρωσης με τις ακμές του πολυγώνου.*
  ***Βήμα 2**: Ταξινόμησε τα σημεία τομής ως προς x.*
  ***Βήμα 3**: Χρωμάτισε τα εικονοστοιχεία που βρίσκονται ανάμεσα σε ζεύγη σημείων τομής [(x1,y1), (x2,y2)] δηλαδή για όσα ισχύει y = y1 = ysl και x1 <= x<= x2, όπου ysl είναι η τρέχουσα θέση της γραμμής σάρωσης.*
- ✓ *Τα ζεύγη σημείων τομής που προκύπτουν από το δεύτερο βήμα είναι ταξινομημένα ως προς χ, προκειμένου να εφαρμοστεί ο κανόνας των μονών-ζυγών σημείων (έλεγχος ισοτιμίας – Parity check).*
- ✓ *Σύμφωνα με αυτόν τον κανόνα, ξεκινώντας από το αριστερότερο άκρο έχουμε αρχικά μηδέν σημεία τομής.*

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Ο αλγόριθμος χρωματίζει τα σημεία που βρίσκονται στις περιοχές μονού αριθμού σημείων τομής και δεξιότερα.*

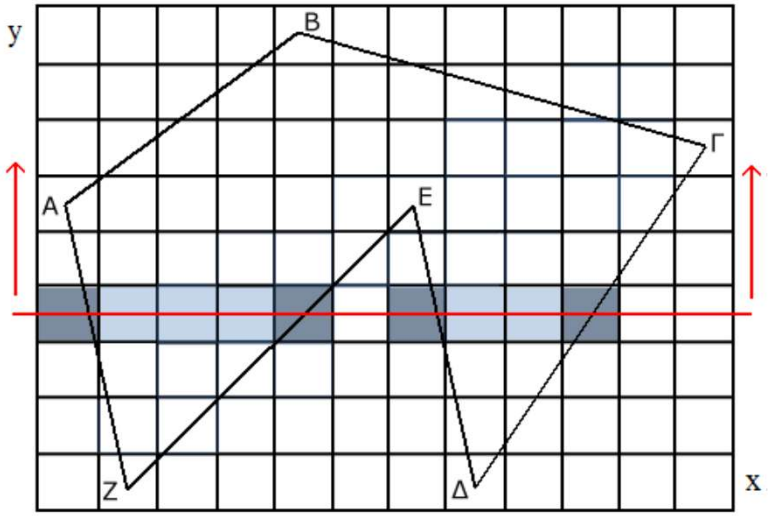


Αποτέλεσμα της σάρωσης πολυγώνου για τη γραμμή σάρωσης y=3

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Υπάρχουν περιπτώσεις που ο παραπάνω αλγόριθμος αποτυγχάνει να αποδώσει με επιτυχία. Τέτοιες περιπτώσεις συμβαίνουν, όταν έχουμε π.χ. μια κορυφή πολυγώνου που βρίσκεται πάνω στη γραμμή σάρωσης. Για το πόσες τομές πρέπει να θεωρήσει ο αλγόριθμος για να λειτουργήσει σωστά, δεν υπάρχει κάποια προφανής λύση που να ισχύει σε όλες τις περιπτώσεις, καθώς άλλοτε μια κορυφή πρέπει να μετρήσει για ένα, δύο ή κανένα σημείο τομής. Στο προηγούμενο παράδειγμα, η κορυφή Ε πρέπει να μετρήσει ως καμία ή δύο, αλλιώς αν μετρήσει ως μία, τότε το τμήμα από το σημείο Ε έως το δεξί άκρο δε θα χρωματιστεί.*



Γραφικά Υπολογιστών

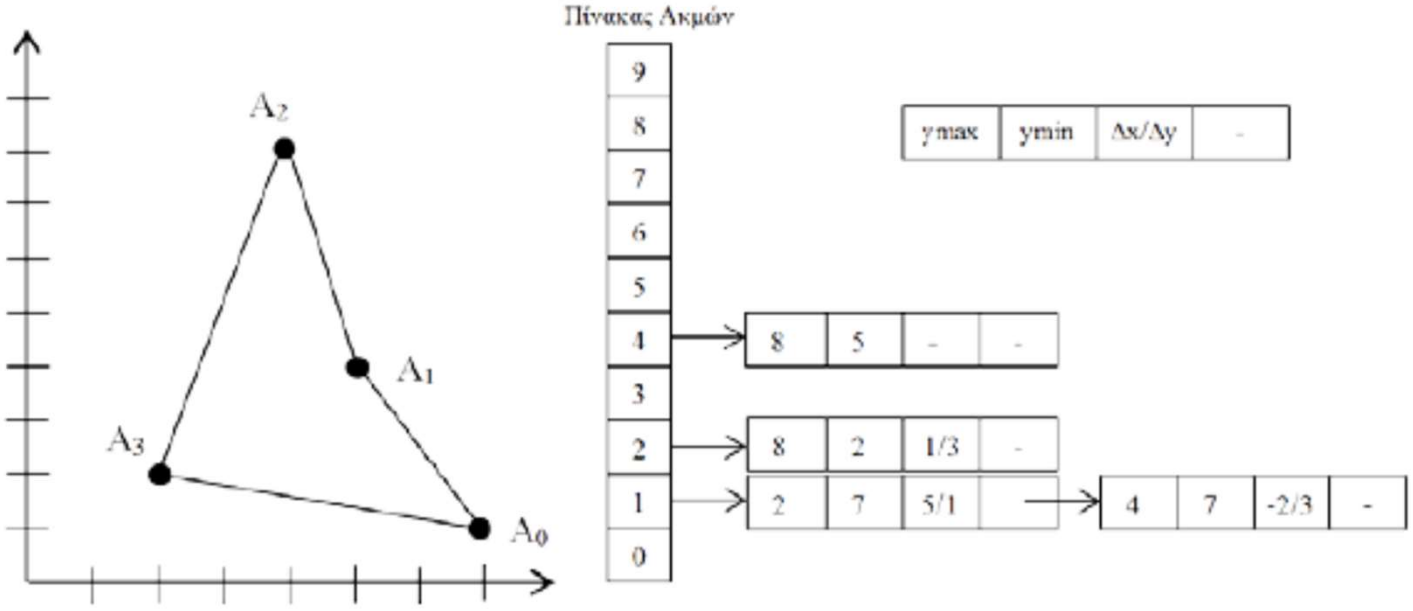
Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Για να αντιμετωπιστούν αυτές οι ιδιάζουσες καταστάσεις, γίνεται μια σειρά από παραδοχές. Δεδομένου ότι κάθε ακμή έχει δύο κορυφές, μόνο η χαμηλότερη κορυφή (κορυφή με το μικρότερο y) θεωρείται έγκυρη. Με αυτόν τον τρόπο θεωρείται ότι υπάρχει μια μικρή μετατόπιση της κορυφής ως προς τον άξονα y, τόσο αμελητέα, ώστε να μην αλλοιώνει την ποιότητα του τελικού αποτελέσματος.*
- ✓ *Μια δεύτερη παραδοχή θέλει τις οριζόντιες γραμμές να μη μετράνε. Έτσι τα σημεία Α και Γ μετράνε για ένα σημείο τομής το καθένα, τα σημεία Β και Ε μετράνε για μηδέν σημεία τομής και τέλος, τα σημεία Ζ και Δ μετράνε για δύο σημεία.*
- ✓ *Το μειονέκτημα που εισάγει το παραπάνω σύνολο παραδοχών είναι ότι τα εικονοστοιχεία που βρίσκονται πάνω πάνω σε ένα πολύγωνο δε χρωματίζονται. Το λάθος αυτό θεωρείται ανεκτό, ενώ στην περίπτωση δύο πολυγώνων που μοιράζονται την ίδια ακμή, η αποφυγή χρωματισμού της κοινής ακμής για το κάτω πολύγωνο δεν αποτελεί ουσιαστικό πρόβλημα. Στην περίπτωση που δύο πολύγωνα μοιράζονται την ίδια κάθετη ακμή, τότε η ακμή χρωματίζεται δύο φορές με υπερισχύον το χρώμα του δεύτερου πολυγώνου.*

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

✓ *Παρατηρούμε ότι στον αλγόριθμο σάρωσης πολυγώνου τα σημεία τομής των ακμών με τη γραμμή σάρωσης i δε διαφέρουν πολύ από την αμέσως υψηλότερη γραμμή i+1. Αυτό μπορούμε να το εκμεταλλευτούμε για να κάνουμε τον αλγόριθμο πιο γρήγορο και αποδοτικό. Ως παράδειγμα δίνεται το πολύγωνο που φαίνεται στην Εικόνα.*



Ο πίνακας ακμών του πολυγώνου και οι γραμμές σάρωσης

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Δύο διαδοχικά στιγμιότυπα της γραμμής σάρωσης, τα yn και yn+1 διαφέρουν κατά ένα εικονοστοιχείο ως προς τον κάθετο άξονα. Ζητούμενο είναι το πόσο διαφέρουν κατά τον οριζόντιο άξονα.*
- ✓ *Με βάση την κλίση της ευθείας που δίνεται από τον τύπο: m=(yn+1-yn)/(xn+1-xn) μπορούμε να υπολογίσουμε την οριζόντια μετατόπιση κατά xn+1=χn+1/m.*
- ✓ *Η παραπάνω συλλογιστική στην πλεγματική οθόνη μπορεί να γραφτεί ως:*

  xn+1=round(xn+1/m), όπου m=ΔΥ/ΔΧ
- ✓ *Έτσι, λοιπόν, μπορούμε να διατηρούμε τις τιμές της κάθε γραμμής σάρωσης με τις ακμές του πολυγώνου στη Λίστα Ενεργών Ακμών (ΛΕΑ) και να ελέγχουμε κατά τη μετάβαση από τη μία γραμμή σάρωσης στην επόμενη για το ποιες αλλαγές έχουν επέλθει λόγω της κλίσης της ακμής.*
- ✓ *Όλες οι ακμές του πολυγώνου διατηρούνται στον Πίνακα Ακμών (ΠΑ).*

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

- ✓ *Μετά τη δημιουργία του Πίνακα Ακμών, ο αλγόριθμος ακολουθεί τα παρακάτω βήματα:*
  - *1) Θέσε το Y στο χαμηλότερο Y των εγγραφών του πίνακα ΠΑ*
  - *2) Αρχικοποίησε την ΛΕΑ (κενή)*
  - *3) Επανάλαβε μέχρι η ΛΕΑ και ο ΠΑ να είναι κενοί (από το Ymin έως το Ymax):*
  - *3.1) Μετακίνησε από τον ΠΑ προς την ΛΕΑ τις εγγραφές που έχουν Ymin=Y*
  - *3.2) Ταξινόμησε την ΛΕΑ κατά X*
  - *3.3) Χρωμάτισε τα εικονοστοιχεία της γραμμής σάρωσης χρησιμοποιώντας ζεύγη από συντεταγμένες X από την ΛΕΑ.*
  - *3.4) Αύξησε την τιμή της γραμμής σάρωσης κατά 1.*
  - *3.5) Αφαίρεσε από την ΛΕΑ τις εγγραφές που έχουν Y=Ymax*
  - *3.6) Για κάθε μη-κάθετη ακμή στη ΛΕΑ ανανέωσε τα X για τη νέα τιμή του Y).*
- ✓ *Ο αλγόριθμος σάρωσης πολυγώνου (Scan-fill Polygon) μπορεί να γενικευτεί για να χειρίζεται σύνολα πολυγώνων, όταν αυτά είναι ταξινομημένα κατά προτεραιότητα (κατά βάθος zindex).*

Γραφικά Υπολογιστών

Αλγόριθμοι Σάρωσης Πολυγώνου

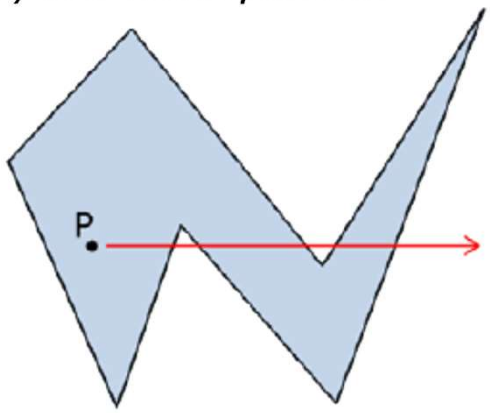
✓ *Μια ιδέα για υλοποίηση του συγκεκριμένου αλγορίθμου παρουσιάζεται παρακάτω:*

```
1. Array ET = readData(edges[]);
2. Ymin=min(ET[]);
3. AET=null;
4. for (y=Ymin; y<=Ymax; y++){
5. MergeSortbyX(ET[y], AET);
6. FillPairsofX(AET);
7. for (int j=0; j=AET.length; j++){
8. if (edge[j].Ymax=Y){
9. remove(edge[j], AET);
10. }
11. Else{
12.Edge[j].X=edge[j].X+dx/dy;
13. }
14. sortbyX(AET);
15.}
```

Γραφικά Υπολογιστών

Έλεγχος Εσωτερικότητας Σημείων

- ✓ *Οι αλγόριθμοι σχεδίασης βασίζονται στον έλεγχο του αν ένα σημείο (εικονοστοιχείο) είναι εσωτερικό ή όχι ενός πολυγώνου.*
- ✓ *Μια μέθοδος για τον έλεγχο αν το σημείο P ανήκει ή όχι στην επιφάνεια του πολυγώνου Π, το οποίο ορίζεται από την ακολουθία n κορυφών (n1, n2, … n-1 με n ακμές), πραγματοποιείται με την αρίθμηση των τομών της ημιευθείας από το P προς το άπειρο.*
- ✓ *Σύμφωνα με αυτήν την τεχνική ελέγχου ισοτιμίας (parity test), με αφετηρία το σημείο P, θεωρείται μια ημιευθεία προς οποιαδήποτε κατεύθυνση και μετράται ο αριθμός των τομών με τις ακμές του πολυγώνου:*

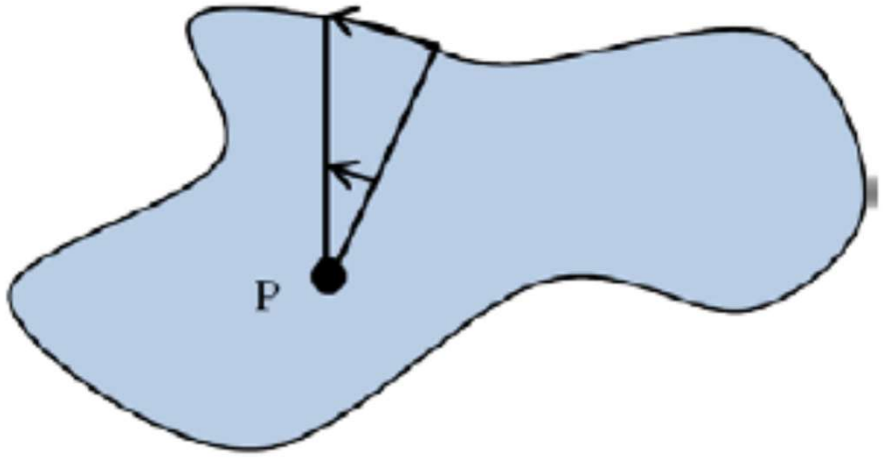


Έλεγχος εσωτερικότητας σημείου με έλεγχο ισοτιμίας (parity test)

Γραφικά Υπολογιστών

Έλεγχος Εσωτερικότητας Σημείων

- ✓ *Αν αυτός ο αριθμός είναι περιττός, τότε το σημείο P βρίσκεται εντός του πολυγώνου Π. Σε αντίθετη περίπτωση, ένας ζυγός αριθμός τομών σημαίνει ότι το σημείο P ανήκει στην εξωτερική περιοχή του Π.*
- ✓ *Η δεύτερη μέθοδος βασίζεται στην καταμέτρηση του αριθμού των περιελίξεων μιας ακτίνας που ενώνει το σημείο P με την περίμετρο μιας κλειστής καμπύλης K για μια πλήρη περιδιάβαση:*



Έλεγχος εσωτερικότητας σημείου με καταμέτρηση του αριθμού των περιελίξεων

Γραφικά Υπολογιστών

Έλεγχος Εσωτερικότητας Σημείων

- ✓ *Οι περιελίξεις ορίζονται ως:*

$$\Omega(K, P) = \frac{1}{2\pi}\int d\varphi$$

- ✓ *Για κάθε περιστροφή προς τη θετική φορά (δηλαδή αντίθετα με τους δείκτες του ρολογιού) το Ω(K,P) αυξάνεται κατά μία μονάδα.*
- ✓ *Αντίθετα, η περιστροφή προς την αρνητική φορά (σύμφωνα με τους δείκτες του ρολογιού) αφαιρεί από το Ω (K,P) μία μονάδα.*
- ✓ *Αν στο τέλος ο αριθμός των περιελίξεων είναι περιττός αριθμός, τότε το σημείο P βρίσκεται εντός της κλειστής καμπύλης K, διαφορετικά είναι εκτός.*

Γραφικά Υπολογιστών

Χρωματισμός Εσωτερικού Πολυγώνων

- ✓ *Το πρόβλημα του χρωματισμού του εσωτερικού ενός πολυγώνου, δηλαδή σχεδιασμού της επιφάνειας που προσδιορίζεται με βάση μια κλειστή πολυγραμμή μπορεί να διατυπωθεί ως το πρόβλημα επιλογής των εικονοστοιχείων της περιμέτρου και του συνόλου των εσωτερικών εικονοστοιχείων.*
- ✓ *Για να γεμίσουμε το εσωτερικό ενός πολυγώνου μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο σάρωσης πολυγώνου για να προσδιορίσουμε τα σημεία τομής με τη γραμμή σάρωσης και σε κάθε βήμα σχεδιάζουμε το ευθύγραμμο τμήμα που ενώνει δύο σημεία τομής (PolygonScan-Conversion).*
- ✓ *Εναλλακτικά μπορούμε να ξεκινήσουμε με ένα εικονοστοιχείο που γνωρίζουμε ότι είναι εσωτερικό του πολυγώνου κα γεμίζουμε τη γειτονική περιοχή μέχρι να συναντήσουμε τα όρια του πολυγώνου.*
- ✓ *Ο αλγόριθμος εσωτερικού γεμίσματος θεωρεί ότι έχει σχεδιαστεί πρώτα η περίμετρος του πολυγώνου και έχει βρεθεί ένα εσωτερικό σημείο που θα θεωρηθεί ως το σημείο εκκίνησης.*

Γραφικά Υπολογιστών

Χρωματισμός Εσωτερικού Πολυγώνων

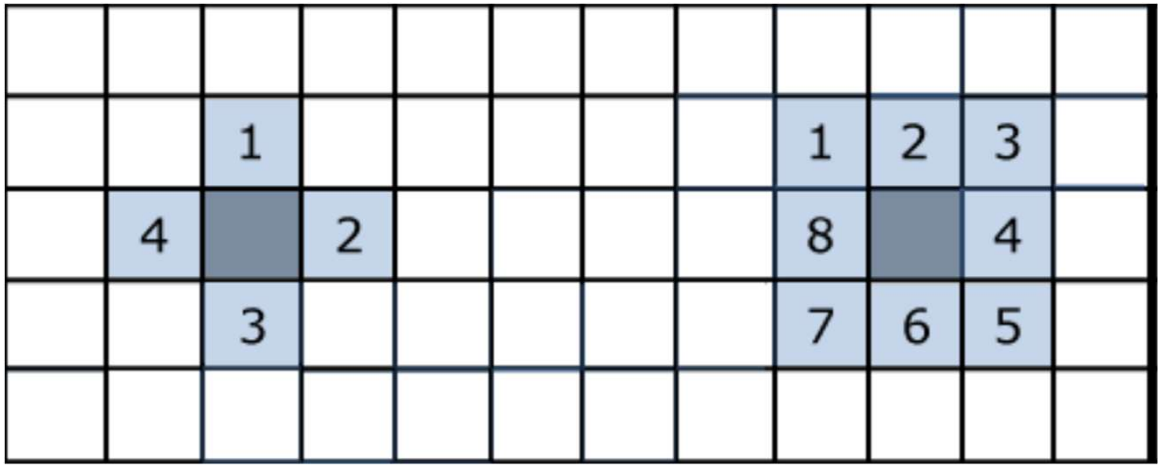
✓ *Ο ψευδοκώδικας του αλγορίθμου φαίνεται παρακάτω:*

```
1. fillPolygon(x,y){
2. Color c;
3. C=getColor(x,y);
4. if (c<>fill_color){
5. setPixel(x,y,color);
6. fillPolygon(x-1,y-1);
7. fillPolygon(x,y+1);
8. fillPolygon(x+1,y+1);
9. fillPolygon(x+1,y);
10.fillPolygon(x+1,y+1);
11.fillPolygon(x,y+1);
12.fillPolygon(x-1,y+1);
13.fillPolygon(x-1,y);
14. }
15.}
```

Γραφικά Υπολογιστών

Χρωματισμός Εσωτερικού Πολυγώνων

- ✓ *Η επαναληπτική κλήση της συνάρτησης fillPolygon γίνεται για τα γειτονικά σημεία. Εδώ μπορεί να οριστεί μια γειτονιά είτε πέντε (5) είτε εννέα (9) σημείων όπως φαίνεται στην Εικόνα:*



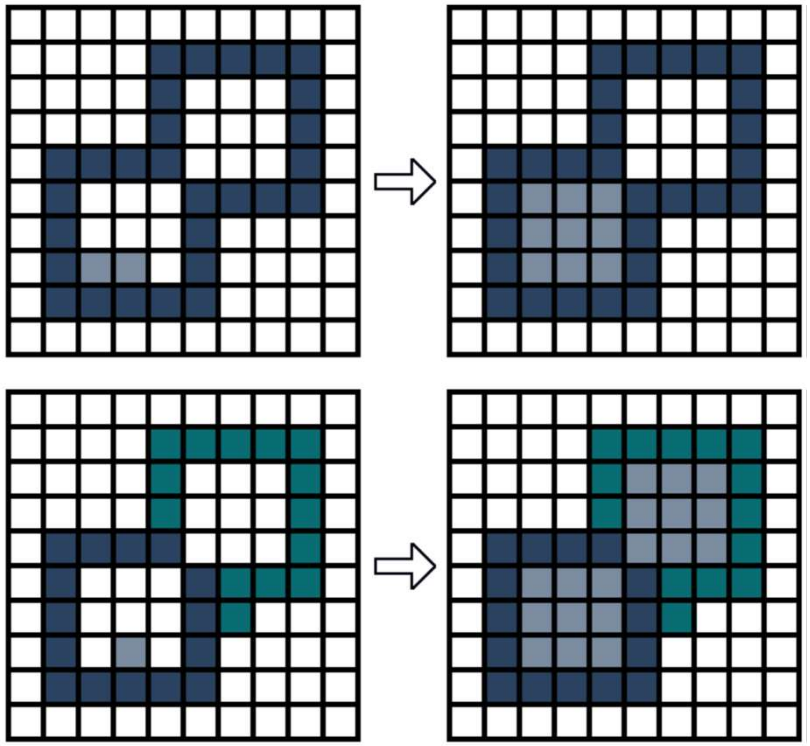
Περιοχές εικονοστοιχείων: α) 4 σημείων (αριστερά) και β) 8 σημείων (δεξιά)

- ✓ *Για περιοχές με τετραπλή σύνδεση, οι τέσσερις αναδρομικές κλήσεις της fill-Polygon κρίνονται επαρκείς. Σε διαφορετική περίπτωση οι περιοχές με οκταπλή σύνδεση χρειάζονται και τις οκτώ αναδρομικές κλήσεις για να καλύψουν τη γειτονιά των 9 σημείων.*

Γραφικά Υπολογιστών

Χρωματισμός Εσωτερικού Πολυγώνων

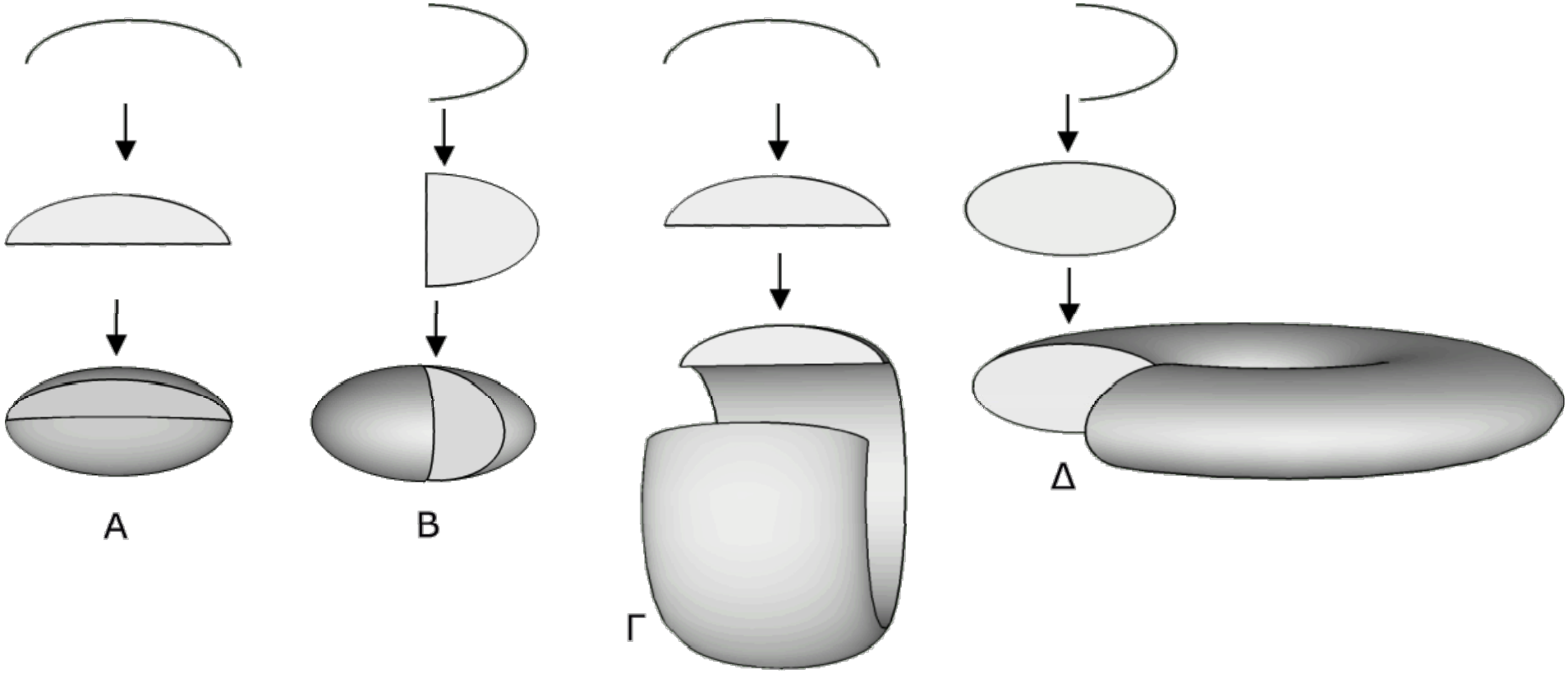
- ✓ *Υπάρχουν, όμως, και μειονεκτήματα των δύο προσεγγίσεων.*
- ✓ *Η τετραπλή σύνδεση ενδέχεται σε ορισμένες περιπτώσεις να σταματά πρόωρα, όπως η περίπτωση στην Εικόνα (πάνω) όπου το πολύγωνο συνεχίζεται προς την πάνω δεξιά κατεύθυνση, αλλά το στένωμα στη μέση κάνει τον αλγόριθμο που βασίζεται στην τετραπλή σύνδεση να σταματήσει, ενώ η οκταπλή σύνδεση του εικονοστοιχείου θα μπορούσε να ολοκληρώσει το γέμισμα.*
- ✓ *Αντίστοιχα, ούτε η οκταπλή σύνδεση δεν είναι απαλλαγμένη από προβλήματα καθώς όπως φαίνεται στην περίπτωση της Εικόνας (κάτω) υπάρχει διαρροή προς ένα γειτονικό σχήμα:*



Γραφικά Υπολογιστών

Σχεδίαση Καμπυλών

- ✓ *Οι καμπύλες μπορεί να υλοποιηθούν από εξίσωση έλλειψης (κύκλου), υπερβολής, παραβολής ή ακόμη και ζεύγους τεμνόμενων ή παράλληλων ευθειών.*



Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Κύκλου

- ✓ *Μια άλλη μεγάλη κατηγορία βασικών σχημάτων είναι οι κωνικές τομές. Ξεκινώντας με τον κύκλο που είναι το πιο απλό παράδειγμα κωνικής τομής, τότε από τη γνωστή εξίσωση του κύκλου αν λύσουμε ως προς y:*

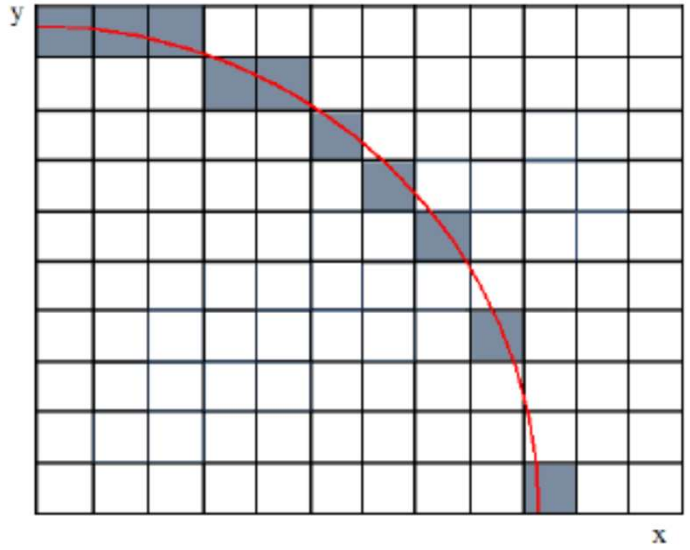
$$y = \pm\sqrt{r^2 - x^2}$$

- ✓ *Με αυτόν τον τρόπο υπολογίζονται ζεύγη σημείων (x, y) που χρωματίζονται για να σχεδιαστεί η περίμετρος του κύκλου. Εξετάζοντας τώρα το πρόβλημα της σχεδίασης π.χ. στο δεύτερο οκταμόριο, ο άξονας που μεταβάλλεται γρηγορότερα είναι ο χ. Δηλαδή, αν ένα σημείο διατρέχει τον κύκλο σύμφωνα με τους δείκτες του ρολογιού, τότε η κάθε επόμενη θέση του θα διαφέρει κατά ένα εικονοστοιχείο ως προς τον οριζόντιο άξονα, ενώ ο κάθετος άξονας θα μειώνεται κατά ένα σε κάποιες περιπτώσεις. Η εφαρμογή μιας λύσης που βασίζεται στην εξίσωση του κύκλου αφήνει κενά στη σχεδίαση, πράγμα μη επιτρεπτό. Μάλιστα, όσο αυξάνεται η κλίση, τόσο αυξάνονται και τα κενά που προκύπτουν ενδιάμεσα.*

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Κύκλου

✓ *Ο παρακάτω Πίνακας, παρουσιάζει έναν κύκλο ακτίνας r=10 στον οποίο προκύπτουν μη συνεχόμενες για το y τιμές.*

Σχεδίαση με βάση την καρτεσιανή εξίσωση της περιφέρειας του κύκλου	Βήματα	X	Y	Επιλεγμένο Εικονοστοιχείο
	1	1	9,95	(1, 10)
	2	2	9,80	(2, 10)
	3	3	9,54	(3, 10)
	4	4	9,17	(4, 9)
	5	5	8,66	(5, 9)
	6	6	8,00	(6, 8)
	7	7	7,14	(7, 7)
	8	8	6,00	(8, 6)
	9	9	4,36	(9, 4)
	10	10	0,00	10, 0)

Βήματα εκτέλεσης του αλγορίθμου σχεδίασης με βάση την εξίσωση του κύκλου από όπου φαίνεται ότι η σχεδίαση δε δίνει ικανοποιητικό αποτέλεσμα διότι δημιουργούνται κενά.

Γραφικά Υπολογιστών

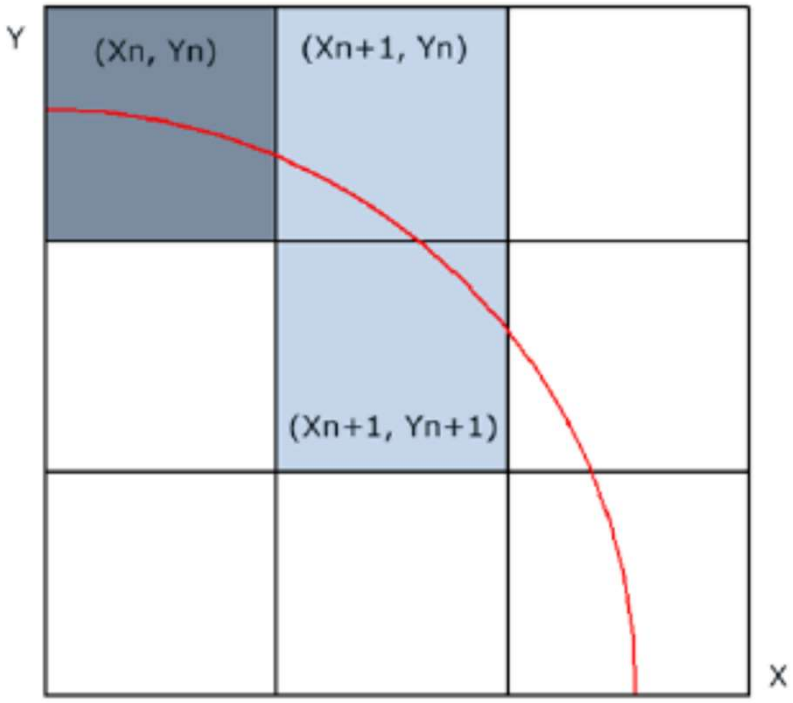
Αλγόριθμοι Σχεδίασης Κύκλου

- ✓ *Στη συνέχεια θα παρουσιαστούν τρόποι σχεδίασης της περιμέτρου με βάση την οκταπλή συμμετρία που παρουσιάζει το σχήμα του κύκλου.*
- ✓ *Αυτό σημαίνει ότι αν σχεδιαστεί ένα από τα οκταμόριά του, τα υπόλοιπα τμήματα του κύκλου μπορούν να προκύψουν με εναλλαγή προσήμων στις τιμές των x και y.*
- ✓ *Αν έχουμε έναν κύκλο ακτίνας r=1 και κέντρο την αρχή των αξόνων, τότε με δεδομένο ένα σημείο (x,y) που ανήκει στον κύκλο μπορούμε να θεωρήσουμε με βεβαιότητα ότι και τα σημεία: [(y, x), (y, -x), (x, -y), (-x, -y), (-y, -x), (-y, x), (-x, y)] ανήκουν επίσης στον ίδιο κύκλο.*
- ✓ *Αυτό είναι απόρροια της οκταπλής συμμετρίας του κύκλου.*

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Κύκλου

- ✓ *Παρατηρούμε ότι σε κάθε μετατόπιση προς τα δεξιά, κατεύθυνση αύξησης του X κατά μία μονάδα σε κάθε βήμα, το επόμενο σημείο προς επιλογή είναι είτε το διπλανό του (Xn+1,Yn), είτε το διαγωνίως πιο κάτω (Xn+1,Yn+1), ανάλογα με το πιο από τα δύο βρίσκεται κοντύτερα στον κύκλο, όπως φαίνεται στην Εικόνα:*



Σχεδίαση με βάση την εξίσωση του κύκλου

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Κύκλου

- ✓ *Αν τώρα θέσουμε έναν παράγοντα μετατόπισης κατά μισό εικονοστοιχείο προς τα πάνω, η συνάρτηση κύκλου γράφεται:*

$$f_{c\left(x,y\frac{1}{2}\right)} = x^2 + \left(y-\frac{1}{2}\right)^2 - r^2 = 0$$

- ✓ *Αυτή είναι η μεταβλητή σφάλματος που δίνει την απόφαση για την επιλογή του καταλληλότερου εικονοστοιχείου. Αν η παραπάνω ποσότητα προκύψει μικρότερη του μηδενός, τότε το σημείο ανήκει στο εσωτερικό του κύκλου. Αν είναι μεγαλύτερη του μηδενός, τότε δεν ανήκει στον κύκλο και είναι εξωτερικό σημείο. Τέλος, αν ισούται με μηδέν, τότε είναι σημείο της περιφέρειας του κύκλου.*

$$f_c = \begin{cases} < 0, \text{σημείο εντός του κύκλου (δίσκου)} \\ = 0, \text{σημείο στην περίμετρο του κύκλου} \\ > 0, \text{σημείο έξω από την περιφέρεια} \end{cases}$$

- ✓ *Αν και η μεταβλητή σφάλματος υπολογίζεται ότι πρέπει να αρχικοποιηθεί στην τιμή:*

$$f_{c(0,r)} = \left(r-\frac{1}{2}\right)^2 - r^2 = \frac{1}{4} - r$$

Γραφικά Υπολογιστών

Αλγόριθμοι Σχεδίασης Κύκλου

✓ *Συνολικά η λύση αυτή εκφρασμένη σε ψευδοκώδικα φαίνεται παρακάτω:*

```
1. int x=0;
2. int y=r;
3. int e=-r;
4. while (x <= y) {
5. setPixel(y, x);
6. setPixel (y, -x);
7. setPixel (x, -y);
8. setPixel (-x, -y);
9. setPixel (-y, -x);
10.setPixel (-y, x);
11.setPixel (-x, y);
12. e=e+2*x+1;
13.x=x+1;
14.if (e >= 0) {
15. e=e-2y+2;
16. y=y-1;
17. } }
```

Γραφικά Υπολογιστών

Σχεδίαση Καμπυλών Bezier

- ✓ *Οι καμπύλες ελεύθερης μορφής χρησιμοποιούνται κατά κόρον στα γραφικά για να αποδώσουν ποικίλα σχήματα και αντικείμενα σε περιβάλλοντα διανυσματικής σχεδίασης, αλλά και για περιγραφή της κίνησης σε περιβάλλοντα animation.*
- ✓ *Το πρόβλημα της σχεδίασης καμπυλών μπορεί να λυθεί εφόσον μιλάμε για μοντέλα ομαλών καμπυλών που ορίζονται με μαθηματικό τρόπο.*
- ✓ *Οι εύκαμπτες καμπύλες (splines) ορίζονται με βάση ένα σύνολο από σημεία ελέγχου όπως είναι οι παραμετρικές καμπύλες Bezier.*
- ✓ *Ένας αλγόριθμος σχεδίασης, προκειμένου να σχεδιάσει μια καμπύλη Bezier, χρειάζεται να δέχεται ως είσοδό του τη λίστα των σημείων ελέγχου τα οποία θα ορίζουν μοναδικά το σχήμα της καμπύλης (ή της επιφάνειας όταν πρόκειται για γενίκευση σε περισσότερες της μίας διάστασης).*
- ✓ *Στις καμπύλες Bezier, η μορφή της καμπύλης εξαρτάται μόνο από τα σημεία ελέγχου που ορίζουν την καμπύλη και άρα υπάρχει η επιθυμητή σταθερότητα.*
- ✓ *Συνήθως χρειάζεται να λυθεί το πρόβλημα της σχεδίασης κυβικής καμπύλης (4 σημεία ελέγχου) καθώς μεγαλύτερης πολυπλοκότητας καμπύλες προκύπτουν από συνδυασμούς διασυνδεμένων καμπυλών (συνέχεια θέσης, κλίσης, καμπυλότητας).*

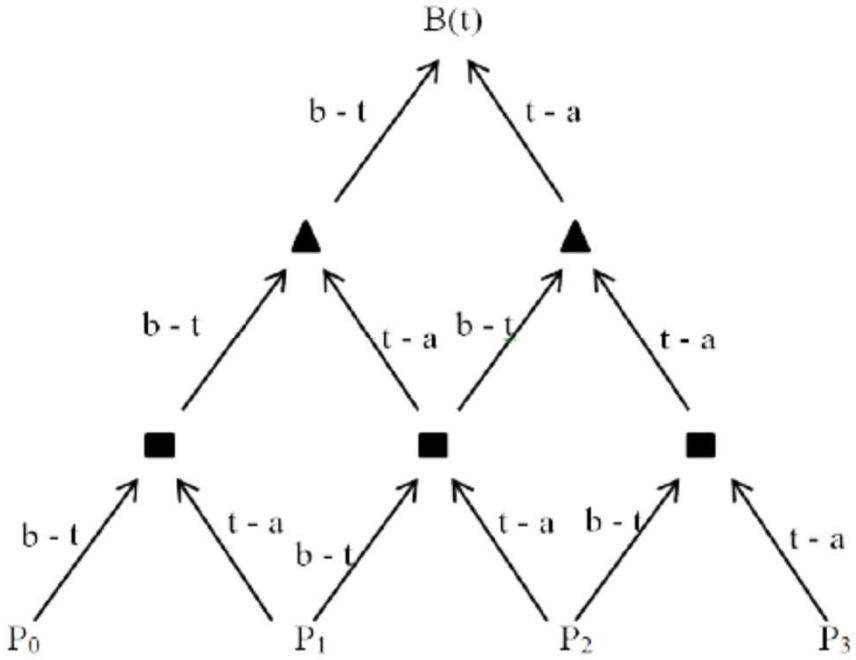
Γραφικά Υπολογιστών

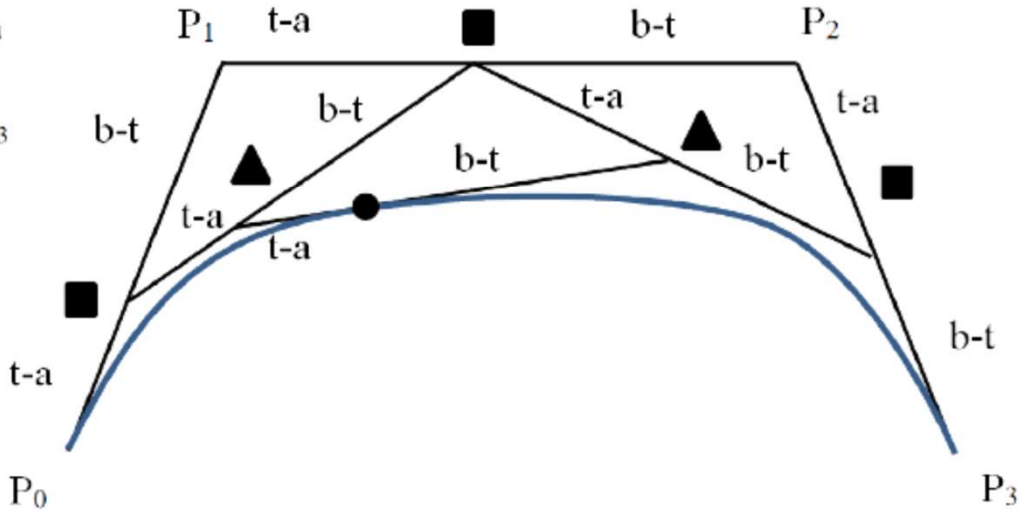
Σχεδίαση Καμπυλών Bezier

- ✓ *Η καμπύλη Bezier βρίσκεται εντός του κυρτού κελύφους ή περιβλήματος (convex hull) που ορίζεται από τα σημεία ελέγχου της, ενώ παρεμβάλλεται στο πρώτο και στο τελευταίο σημείο διότι P(0)=p0 και P(n)=pn.*
- ✓ *Αυτό για το πρόβλημα της σχεδίασης σημαίνει ότι το πρώτο και το τελευταίο σημείο της καμπύλης είναι γνωστά, ενώ αναζητούνται τα ενδιάμεσα.*
- ✓ *Σε περίπτωση συγγραμμικών σημείων ελέγχου προκύπτει ευθύγραμμο τμήμα.*
- ✓ *Η πολυπλοκότητα της καμπύλης και επομένως και της σχεδίασης είναι ανάλογες με τον αριθμό των σημείων ελέγχου. Έτσι, με n σημεία ελέγχου προκύπτουν n-1 διαδοχικά επίπεδα γραμμικής παρεμβολής για να σχεδιαστεί μια καμπύλη Bezier βαθμού n. Αν B(t) είναι μια καμπύλη Bezier που ορίζεται με βάση τα P0, P1, P2, P3 σημεία ελέγχου και το t παίρνει τιμές στο διάστημα [a, b], τότε για $0 \leq t \leq 1$ τα διαστήματα t-a ισούνται με t (το πρώτο τμήμα κάθε ευθύγραμμου τμήματος που ενώνει δύο διαδοχικά σημεία ελέγχου) και το b-t γίνεται 1-t που αντιστοιχεί στο δεύτερο τμήμα.*

Γραφικά Υπολογιστών

Σχεδίαση Καμπυλών Bezier





Γραφικά Υπολογιστών

Σχεδίαση Καμπυλών Bezier

✓ *Ο ψευδοκώδικας του αλγορίθμου deCasteljau ο οποίος παρατίθεται παρακάτω, προϋποθέτει ότι για να σχεδιάσουμε ολόκληρη την καμπύλη, υπολογίζουμε ένα-ένα τα σημεία και τα συνδέουμε με ευθύγραμμα τμήματα, όπως φαίνεται και στις 2 προηγούμενες εικόνες.*

```
1. point tmpPoint;
2. double step=0.001;
3. array Points[]=getControlPoints;
4. for (double t=0; t<=1; t+=step){
5. tmpPoint=getCasteljauPoint(length(Points)-1, 0, t);
6. setPixel(tmpPoint.X, tmpPoint.Y); }
8. point getCasteljauPoint (int r, int I, double t){
9. if (r==0) return Points[i];
10.point p1 = getCasteljauPoint(r-1, i, t);
11.point p2 = getCasteljauPoint(r-1, i+1, t);
12.point resPoint;
13.resPoint.X = int((1-t)*p1.X+t*p2.X);
14.resPoint.Y = int((1-t)*p1.Y+t*p2.Y));
15.return resPoint; }
```

Γραφικά Υπολογιστών

Βιβλιογραφία

- ✓ *Σ. Καλαφατούδη, "Γραφικά με Υπολογιστή," Εκδόσεις Νέων Τεχνολογιών, 1991.*
- ✓ *Α. Στυλιάδη, "Γραφικά με Η/Υ," Εκδόσεις Ζήτη, 1999.*
- ✓ *Θ. Θεοχάρης, Α. Μπέμ, "Γραφικά: Αρχές και Αλγόριθμοι," Εκδόσεις Συμμετρία, 1999.*
- ✓ *Γ. Παρασχάκη, Μ. Παπαδοπούλου, Π. Πατιάς, "Σχεδίαση με Η/Υ," Εκδόσεις Ζήτη, 1999.*
- ✓ *J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, R. L. Phillips, "Introduction to Computer Graphics, "Addison Wesley, 1994.*
- ✓ *Κ. Μουστάκας Ι. Παλιόκας Α. Τσακίρης Δ. Τζοβάρας, (2015), "Γραφικά και Εικονική Πραγματικότητα", ISBN: 978-960-603-255-4, www.kallipos.gr*
- ✓ Λαζαρίνης, Φ, (2015), "Πολυμέσα", ISBN: 978-960-603-141-0, *www.kallipos.gr*
- ✓ *Γεώργιος Λέπουρας, Αγγελική Αντωνίου, Νίκος Πλαιής, Δημήτρης Χαρίχος, (2015), "Ανάπτυξη συστημάτων εικονικής πραγματικότητας", ISBN: 978-960-603-382-7, www.kallipos.gr*