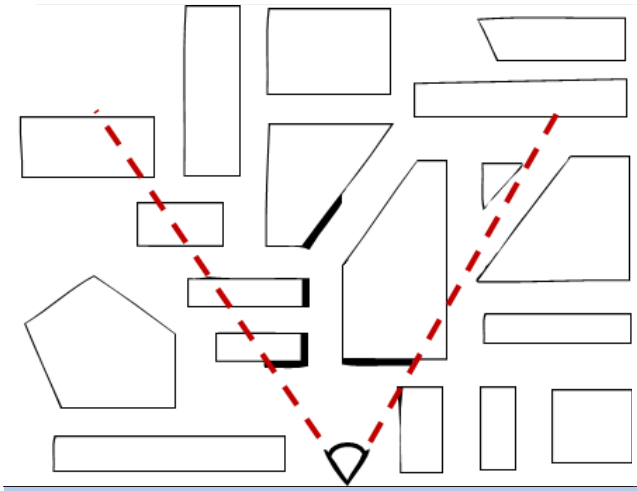


Γραφικά Υπολογιστών

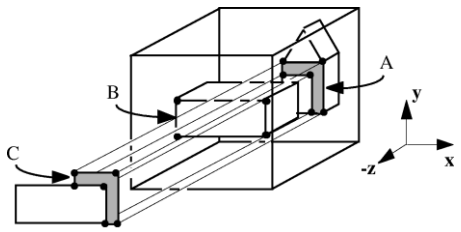
Διαγραφή Πίσω Επιφανειών και Απόκρυψη

Andreas Aristidou
andarist@ucy.ac.cy
http://www.andreasaristidou.com

1

Ανασκόπηση: Rendering Pipeline

- Σχεδόν ολοκληρώσαμε το rendering pipeline:
  - Μετασχηματισμοί αντικειμένων (Modeling transformations)
  - Μετασχηματισμοί κάμερας (Viewing transformations)
  - Προβολή (Projection transformations)
  - Αποκοπή (Clipping)
  - Σάρωσή (Scan conversion)
- Τώρα γνωρίζουμε τα πάντα για το πώς να σχεδιάσουμε ένα πολύγωνο στην οθόνη, εκτός από την απόκρυψη, ή αλλιώς ***visible surface detection***.



Polygon *A* is clipped by *B* which is in front of it. A new sub-polygon, *C*, is created.

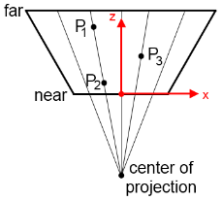
Διαγραφή Πίσω Επιφανειών και Απόκρυψη

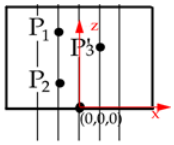
- Καθορίζει το μέρος κάθε αντικειμένου που είναι ορατό στην τελική εικόνα
- Γιατί χρειάζονται οι αλγόριθμοι απόκρυψης;
  - Αποφεύγουν τη δημιουργία λανθασμένων εικόνων
  - Επιταχύνουν τη δημιουργία εικόνων
- Στα Αγγλικά η απόκρυψη ονομάζεται "**visible surface determination**"
- Και η διαγραφή πίσω επιφανειών "**back-face elimination**" ή "**back face culling**"

To render or not to render, that is the question…

Περιεχόμενα

- Σήμερα θα ασχοληθούμε με τις ορατές τεχνικές ανίχνευσης επιφάνειας:
  - Γιατί ανίχνευση επιφάνειας;
  - Ανίχνευση του πίσω μέρους το αντικειμένου
  - Μέθοδος Depth-buffer
  - Μέθοδος A-buffer
  - Μέθοδος Scan-line





Αόρατες οντότητες

- *Πότε ένα πολύγωνο θα ήταν αόρατο;*
  - Τα πολύγωνα που είναι εκτός του πεδίου προβολής
  - Εάν από το πολύγωνο βλέπουμε το πίσω μέρος του (*backfacing*)
  - Εάν το πολύγωνο αποκρύβεται από κάποιο άλλο που είναι πιο κοντά στο viewpoint
- Όπως αντιλαμβάνεστε, για λόγους απόδοσης δεν θέλουμε να ασχολούμαστε με πολύγωνα τα οποία είναι *αόρατα* (outside field of view or backfacing)
- Για λόγους απόδοσης και σωστής απεικόνισης, πρέπει να γνωρίζουμε εάν ένα πολύγωνο αποκρύβεται από κάποιο άλλο αντικείμενο.

Visible Surface Detection

Υπάρχουν πολλοί αλγόριθμοί και τεχνικές που έχουν αναπτυχθεί κατά διαστήματα για να λύσουν το visible surface detection

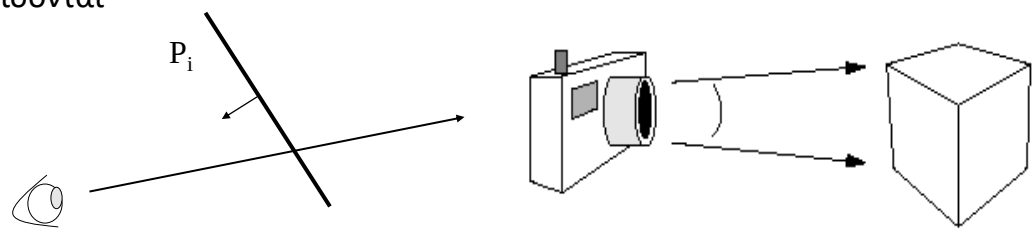
- Ορισμένες μέθοδοι απαιτούν περισσότερο χρόνο επεξεργασίας.
- Ορισμένες μέθοδοι απαιτούν μεγαλύτερη μνήμη.
- Κάποιες άλλες μπορούν να εφαρμοστούν μόνο σε συγκεκριμένους τύπους δεδομένων

Δύο κύριες προσεγγίσεις

- Οι Visible surface detection αλγόριθμοι ταξινομούνται κυρίως ως:
  - **Object Space Methods:** Συγκρίνει αντικείμενα και μέρη αντικειμένων μεταξύ τους μέσα στον ορισμό της σκηνής για να προσδιορίσει ποιες επιφάνειες είναι ορατές
  - **Image Space Methods:** Η ορατότητα αποφασίζεται ανά σημείο σε κάθε θέση pixel στο επίπεδο προβολής
- Οι Image space methods είναι πολύ πιο διαδεδομένες!

Διαγραφή πίσω επιφανειών (Back-Face Detection)

- Υποθέτουμε ότι τα αντικείμενα μας είναι στερεά
- Κάθε πολύγωνο έχει «μπρος» και «πίσω», ανάλογα με την σειρά των κορυφών του
- Το πιο απλό πράγμα που μπορούμε να κάνουμε είναι να βρούμε τα πρόσωπα στο πίσω μέρος του πολύεδρα και να τα απορρίψουμε
- Πολύγωνα των οποίων η κάθετος δεν κοιτάζει προς την κάμερα δεν αποδίδονται



Διαγραφή πίσω επιφανειών (Back-Face Detection)

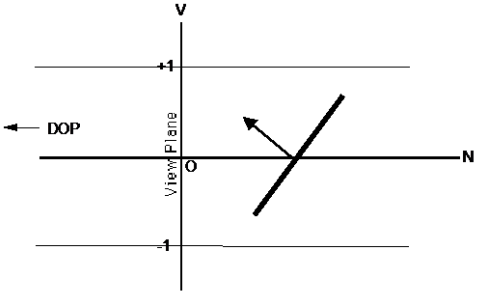
- Γνωρίζουμε από πριν ότι ένα σημείο (x, y, z) είναι πίσω από μια επιφάνεια πολυγώνου αν:

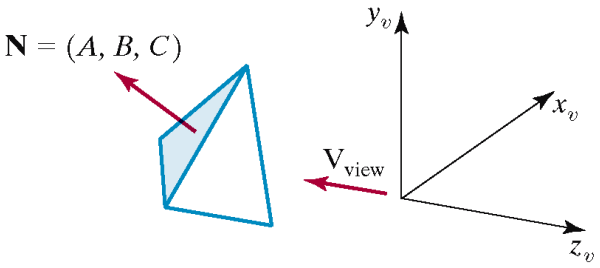
$$Ax + By + Cz + D < 0$$

- όπου A, B, C & D είναι οι παράμετροι της επιφάνειας.

Διαγραφή πίσω επιφανειών (Back-Face Detection)

- Εναλλακτικά, βεβαιωθείτε ότι έχουμε ένα δεξιόχειρο σύστημα με την κατεύθυνση κατά μήκος του αρνητικού Z-άξονα
- Μπορούμε να πούμε ότι αν το Z συστατικό της καθέτου στο πολύγωνο είναι μικρότερο από μηδέν, τότε η επιφάνεια δεν είναι ορατή





Διαγραφή πίσω επιφανειών (Back-Face Detection)

- Μπορούμε να απλοποιήσουμε αυτή τη μέθοδο αν υποθέσουμε πως το κάθετο διάνυσμα **N** στην επιφάνεια του πολυγώνου, έχει τις καρτεσιανές τιμές (A,B,C).

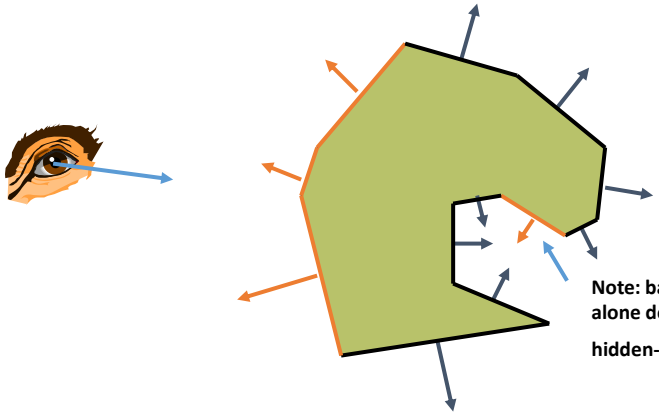
- Εάν **V** είναι ένα διάνυσμα στη κατεύθυνση προβολής, τότε το πολύγωνο είναι πίσω επιφάνεια και δεν φαίνεται αν **V●N > 0.**

Διαγραφή πίσω επιφανειών (Back-Face Detection)

- Σε γενικές γραμμές, η διαγραφή πίσω επιφανειών μπορεί να εξαλείψει περίπου το ήμισυ των επιφανειών ενός πολύγωνου σε μια σκηνή.
- Πιο περίπλοκες επιφάνειες ωστόσο μας δυσκολεύουν!
- Χρειαζόμαστε καλύτερες τεχνικές για να χειριστούμε τέτοιου είδους καταστάσεις.

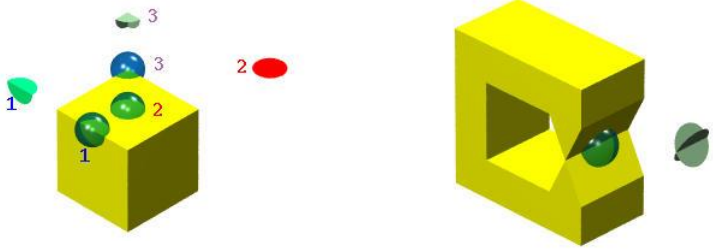
Διαγραφή πίσω επιφανειών (Back-Face Detection)

- Στην επιφάνεια των πολυγώνων των οποίων τα κάθετα διανύσματα δείχνουν μακριά από την κάμερα είναι πάντα μη ορατά:



Διαγραφή πίσω επιφανειών (Back-Face Detection)

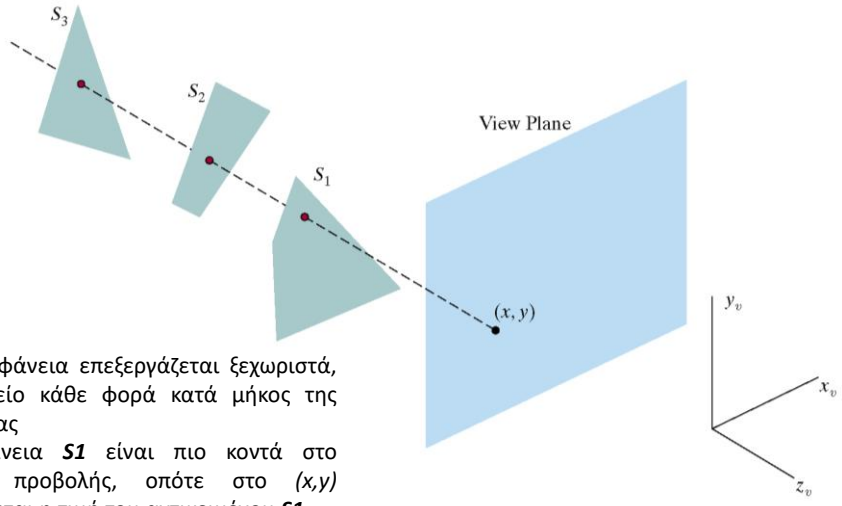
- Τα περισσότερα αντικείμενα στη σκηνή είναι συνήθως «στερεά» και μη διαφανές!
  - Τι συμβαίνει εάν δεν είναι;



Μέθοδοι Depth-Buffer

- Συγκρίνει τις τιμές του βάθους επιφάνειας σε μια σκηνή για κάθε μια θέση pixel στο επίπεδο προβολής
- Συνήθως εφαρμόζεται σε σκηνές που περιέχουν μόνο πολύγωνα
- Δεδομένου ότι οι τιμές βάθους μπορούν να υπολογιστούν εύκολα, αυτό τείνει να είναι πολύ γρήγορο
- Είναι επίσης γνωστές ως μέθοδοι z-buffer

Μέθοδοι Depth-Buffer



- Κάθε επιφάνεια επεξεργάζεται ξεχωριστά, ένα σημείο κάθε φορά κατά μήκος της επιφάνειας
- Η επιφάνεια ***S1*** είναι πιο κοντά στο επίπεδο προβολής, οπότε στο *(x,y)* προβάλλεται η τιμή του αντικειμένου ***S1***.

Ο αλγόριθμος του Depth-Buffer

1. Αρχικοποίησε το depth buffer και το frame buffer για το σημείο (x, y)

   depthBuff(x, y) = 1.0

   frameBuff(x, y) = bgColour

Ο αλγόριθμος του Depth-Buffer

2. Επεξεργαζόμαστε κάθε φορά ένα πολύγωνο. Για κάθε θέση σε μια επιφάνεια πολυγώνου, σύγκρινε τη θέση βάθους σε σχέση με την προηγούμενη αποθηκευμένη θέση στο depth buffer για να προσδιορίσουμε την ορατότητα του.
   - Για κάθε προβαλλόμενο θέση (x, y) pixel στο πολύγωνο, υπολόγισέ το βάθος του z (if not already known)
   - Εάν το z < depthBuff(x, y), υπολόγισε το χρώμα της επιφάνειας αυτής της θέσης και θέσε

     depthBuff(x, y) = z

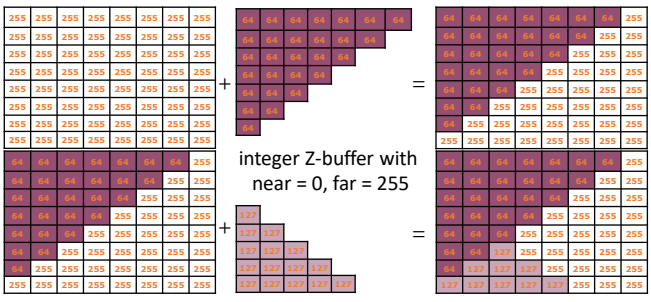
     frameBuff(x, y) = surfColour(x, y)

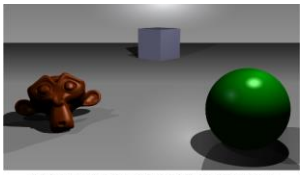
Αφού επεξεργαστούμε όλες τις επιφάνειες, τότε τα depthBuff και frameBuff θα έχουν τις σωστές τιμές

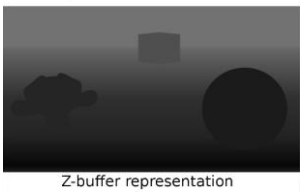
Ο αλγόριθμος του Depth-Buffer

```
void zBuffer() {
  int x, y;
  for (y = 0; y < YMAX; y++)
  for (x = 0; x < XMAX; x++) {
      WritePixel (x, y, BACKGROUND_VALUE);
      WriteZ (x, y, 1);
  }
  for each polygon {
    for each pixel in polygon's projection {
      //plane equation
      double pz = Z-value at pixel (x, y);
        if (pz <= ReadZ (x, y)) {
        // New point is closer to front of view
        WritePixel (x, y, color at pixel (x, y))
        WriteZ (x, y, pz);
      }
    }
  }
}
```


Ο αλγόριθμος του Depth-Buffer




A simple three-dimensional scene


Z-buffer representation

Επαναληπτικοί υπολογισμοί

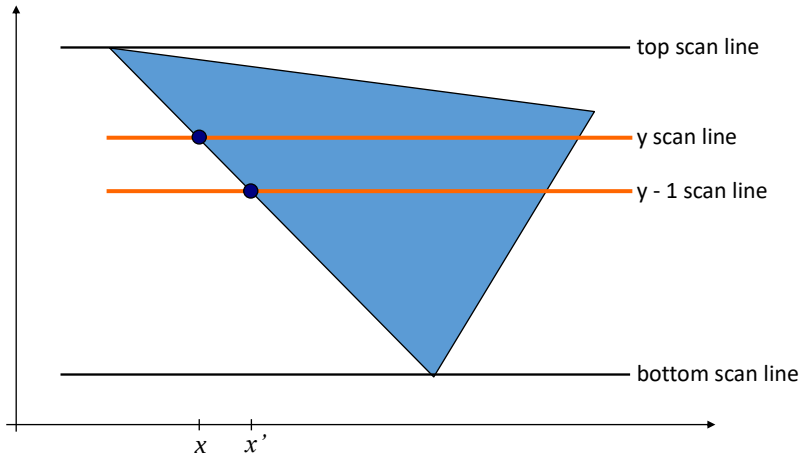
- Ο αλγόριθμος depth-buffer ξεκινά από την πάνω κορυφή του πολυγώνου
- Ακολούθως, επαναληπτικά υπολογίζει τις x-συντεταγμένες μέχρι την κάτω αριστερή κορυφή του πολύγωνου
- Η τιμή x για τη θέση έναρξης σε κάθε γραμμή σάρωσης μπορεί να υπολογιστεί από την προηγούμενη

$$x' = x - \frac{1}{m}$$ όπου m είναι η κλίση

- Η τιμές βάθος κατά μήκος των κορυφών υπολογίζονται ως εξής

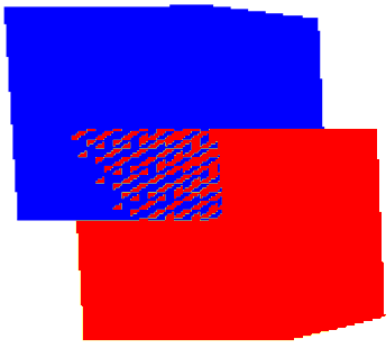
$$z' = z - \frac{A/m + B}{C}$$

Επαναληπτικοί υπολογισμοί



Z-Fighting

- Το Z-fighting συμβαίνει όταν δυο αντικείμενα έχουν ίδια τιμή στο the z-buffer



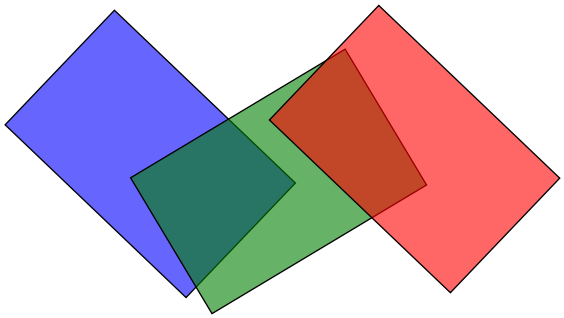
Two intersecting cubes

Μέθοδος A-Buffer

- Η μέθοδος A-buffer είναι μια επέκταση της μεθόδου depth-buffer
- Η μέθοδος A-buffer αναπτύχθηκε από τα Lucasfilm Studios για το σύστημα απόδοσης REYES (**R**enders **E**verything **Y**ou **E**ver **S**aw)

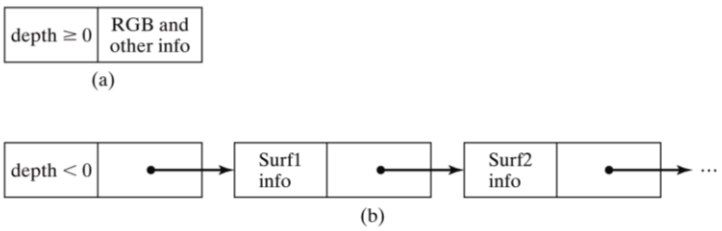
Μέθοδος A-Buffer

- Η μέθοδος A-buffer επεκτείνεται τη μέθοδο depth-buffer για να επιτρέπονται οι διαφανείς επιφάνειας
- Η βασική δομή στο A-buffer είναι το *accumulation buffer*



Μέθοδος A-Buffer

- Εάν το βάθος >= 0, τότε στην επιφάνεια αποθηκεύονται τα pixels όπως και προηγούμενος
- Εάν όμως το βάθος < 0 τότε αποθηκεύεται ένας δείκτης που δείχνει σε διαφορές επιφάνειες



Μέθοδος A-Buffer

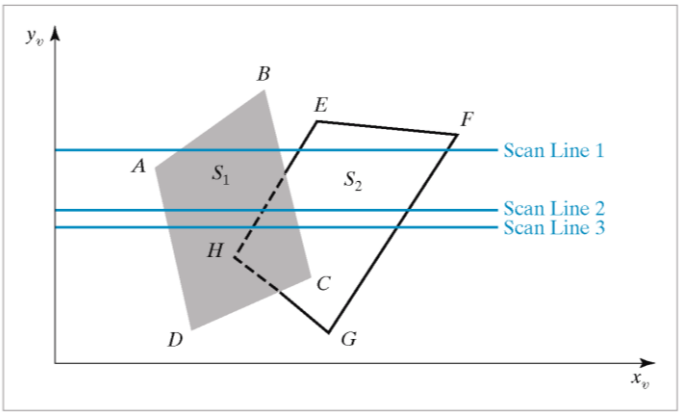
- Σε κάθε επιφάνεια έχουμε τις εξής πληροφορίες:
  - Τα RGB στοιχεία έντασης
  - Παράμετρος α-διαφάνειας (opacity)
  - Βάθος (Depth)
  - Ποσοστιαία περιοχή κάλυψης (Percent of area coverage)
  - Η ταυτότητα της επιφάνειας (Surface identifier)
  - Άλλες παράμετροι
- Ο αλγόριθμος προχωρά ακριβώς όπως τον αλγόριθμο depth buffer
- Το βάθος και η τιμή διαπεραστικότητας (opacity) χρησιμοποιούνται για τον προσδιορισμό του τελικού χρώματος του pixel

Painter's Algorithm

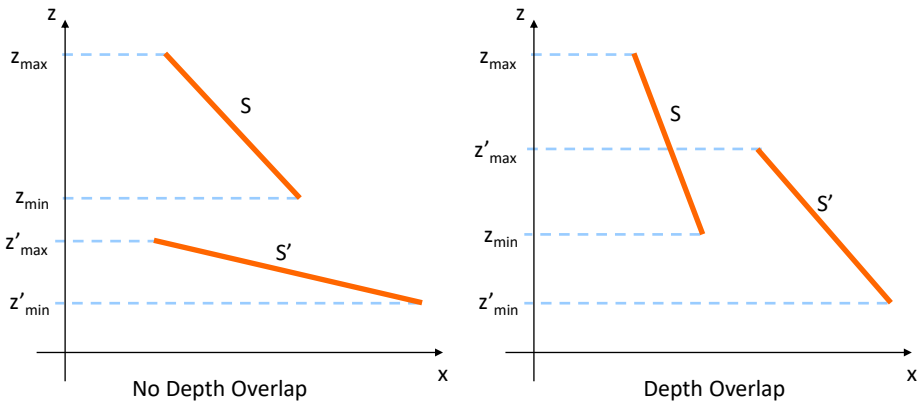
Deadpool © Twentieth Century Fox, Marvel Entertainment

Μέθοδος Scan-Line

- Μια διαφορετική μέθοδος για τον εντοπισμό των ορατών επιφανειών
- Υπολογίζει και συγκρίνει τις τιμές βάθους κατά μήκος των διαφόρων γραμμών σάρωσης για μια σκηνή

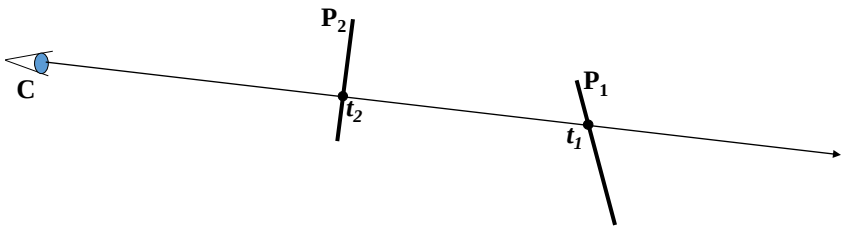


Depth-Sorting Method (Painter's Algorithm)



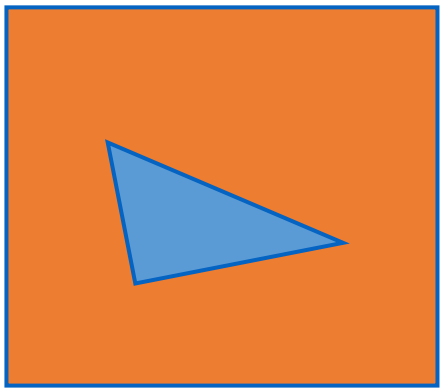
Visibility (Priority) Ordering

- ***Θα σχεδίαζες το P_1 μετά P_2 για να δεις το σωστό αποτέλεσμα (Painters Algorithm)***
- Δεδομένου ενός συνόλου από πολύγωνα *S* και ένα viewpoint *C*, βρέιτε μια σειρά $\{P_1 \ldots P_n\}$ στο *S* τέτοια ώστε οποιοδήποτε πολύγωνο P_i δεν συγκαλύπτει οποιοδήποτε από τα πολύγωνα $\{P_{i+1} \ldots P_n\}$.
- Ένας άλλος τρόπος να το σκέφτεστε: για κάθε 2 πολύγωνα που τέμνονται από μια ακτίνα μέσω του C, $P_{i\,;}$ έχει μεγαλύτερη προτεραιότητα από $P_{j,}$ (with i < j)

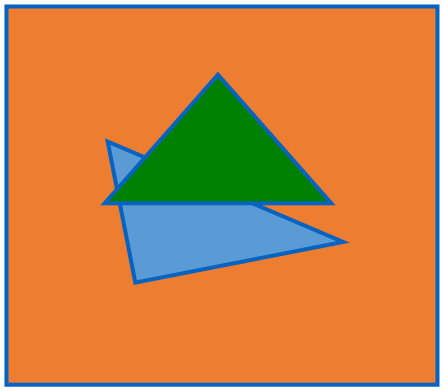


Depth-Sorting Algorithm (Painter's Algorithm)

- Απλή προσέγγιση: Απόδωσε τα πολύγωνα από πίσω προς τα εμπρός, "ζωγραφίζοντας πάνω" από τα προηγούμενα πολύγωνα:

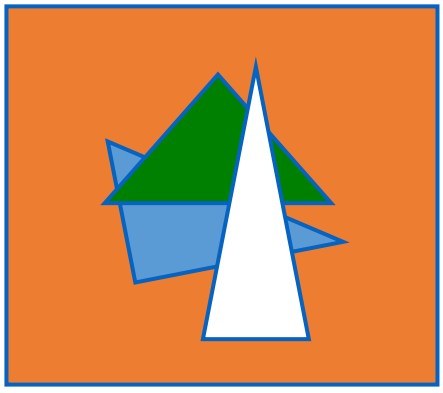


Depth-Sorting Algorithm (Painter's Algorithm)



35

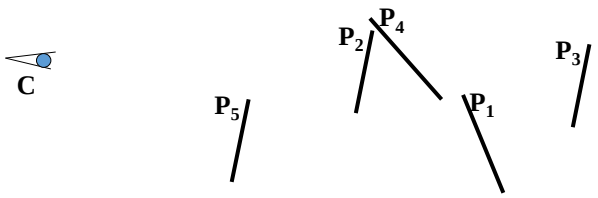
Depth-Sorting Algorithm (Painter's Algorithm)



36

Αρχή του αλγόριθμου του ζωγράφου (Painter's algorithms)

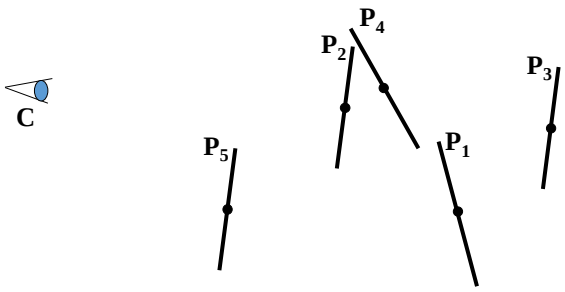
- Ταξινόμηση των αντικειμένων (ή πολυγώνων) από πίσω –προς-μπροστά
- Για να πάρουμε την τελική εικόνα: τα σχεδιάζουμε με αυτή τη σειρά η οποία επιτρέπει στα τελευταία (κοντινότερα) να ζωγραφιστούν πάνω (overwrite) τα προηγούμενα (μακρινότερα)



- Είναι μια υβριδική προσέγγιση καθώς το πρώτο βήμα, η ταξινόμηση, είναι στο χώρο αντικείμενου και το δεύτερο στο χώρο εικόνας
- **Το δυσκολότερο μέρος είναι η ταξινόμηση**

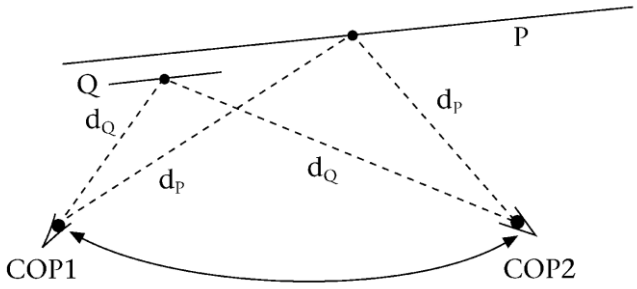
Z-sort στο Projection Space

- Απλή ταξινόμηση από πίσω προς τα μπρος όλων των πολυγώνων βασισμένη στο κέντρο τους



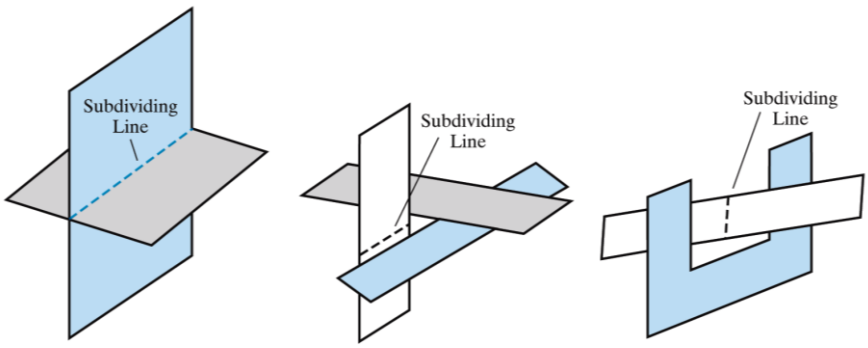
Z-sort στο Projection Space

- Δεν δουλεύει πάντα, π.χ.



Scan-Line Method Limitations

- Γενικά, οι scan-line μέθοδοι έχουν προβλήματα όταν οι επιφάνειες κόβονται μεταξύ τους ή με άλλο τρόπο επικαλύπτονται κυκλικά
- Οι επιφάνειες αυτές πρέπει να χωριστούν



Scan-Line Method Limitations

- Τα ***Intersecting polygons*** παρουσιάζουν επίσης πρόβλημα
- Ακόμη και τα μη τεμνόμενα πολύγωνα μπορούν να σχηματίσουν έναν κύκλο χωρίς έγκυρη σειρά ορατότητας:

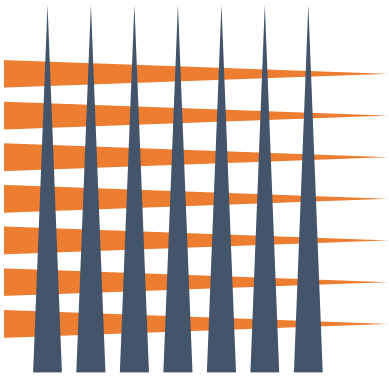
Analytic Visibility Algorithms

- Οι αλγόριθμοι έγκαιρης ορατότητας υπολογίζουν το σύνολο των ορατών τμημάτων του πολυγώνου άμεσα, και στη συνέχεια παρουσιάζουν τα επι μέρους τμήματα σε μια οθόνη:

Analytic Visibility Algorithms

- ***What is the minimum worst-case cost of computing the fragments for a scene composed of* n *polygons?***

- Answer:
  $O(n^2)$



Περίληψη

- Πρέπει να διασφαλίσουμε ότι θα προβάλουμε μόνο ορατές επιφάνειες κατά την απόδοση μιας σκηνής
- Υπάρχουν διάφορες τεχνικές για να γίνει αυτό, όπως
  - Back face detection
  - Depth-buffer method
  - A-buffer method
  - Scan-line method
- Τώρα θα εξετάσουμε περισσότερες τεχνικές και θα βρούμε ποιες τεχνικές είναι κατάλληλες και για ποιες περιπτώσεις