ΘΕΜΑΤΑ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΟΡΑΣΗΣ ΚΑΙ ΓΡΑΦΙΚΩΝ

ΟΡΑΤΟΤΗΤΑ

2013-2014

Ε. Θεοδωρίδης, Α. Τσακαλίδης

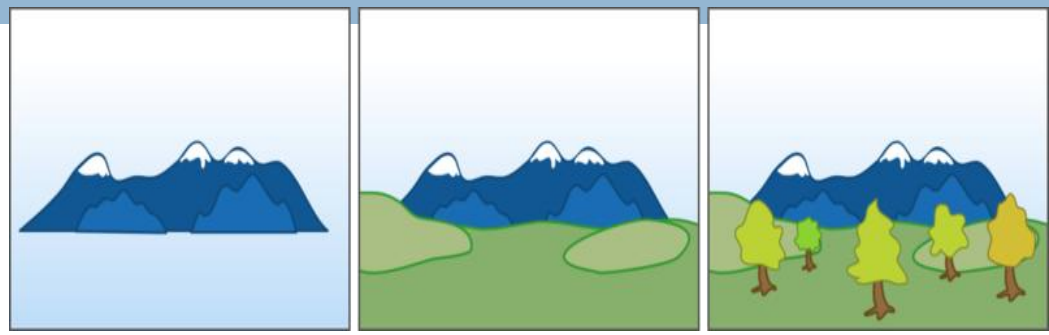
Ορατότητα - Visibility

- Δεδομένου ενός συνόλου πολυγώνων (επιφανειών), ποιο είναι ορατό σε κάθε pixel
- Απομάκρυνση μη ορατών επιφανειών - hidden surface removal
- Μεγάλος αριθμός αλγορίθμων οι οποίοι διαχωρίζονται σε δύο ομάδες:
  - Χώρος Αντικειμένων: υπολογισμοί σε γεωμετρικά primitives
  - Χώρος Εικόνας: υπολογισμός σε επίπεδο pixel
- Όλα τα στάδια του viewing pipeline διατηρούν το βάθος (z), συνεπώς είναι δυνατός ο υπολογισμός
  - World, View και Screen spaces μπορούν να χρησιμοποιηθούν
  - Το βάθος μπορεί να ενημερώνεται ανά pixel καθώς σαρώνονται τα πολύγωνα ή οι γραμμές

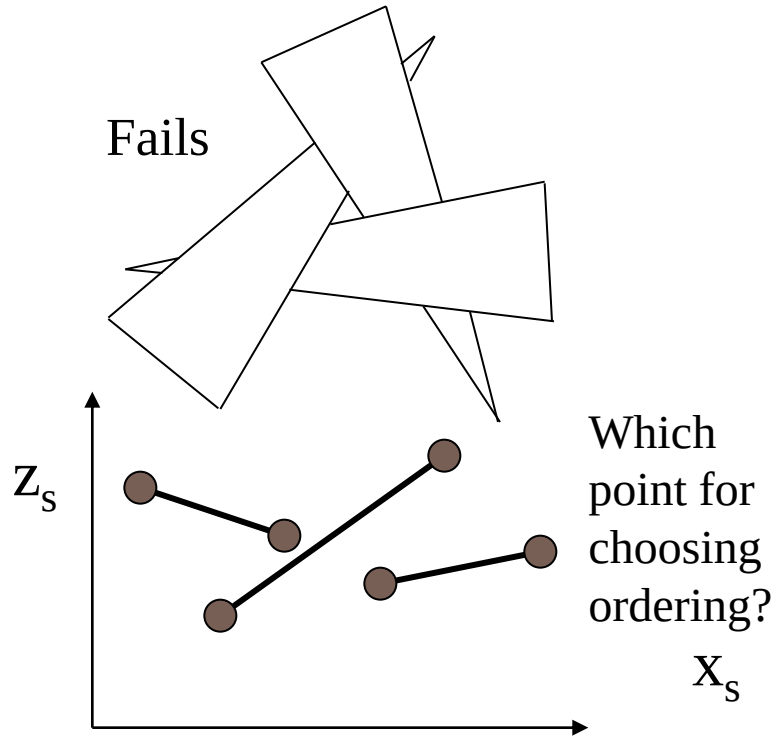
Θέματα Ορατότητας

- Efficiency - είναι αργό να επικαλύπτονται και να επανα-ζωγραφίζονται pixel που τελικά δεν θα φανούν
- Accuracy - το τελικό αποτέλεσμα πρέπει να είναι ορθό και να συμπεριφέρεται ομαλά όταν το σημείο παρατήρησης μεταβάλλεται
- Τεχνικές διαχείρισης μεγάλες συλλογές σύνθετων αντικειμένων
- Σε σύνθετες σκηνές λίγα αντικείμενα παρουσιάζονται σε κάθε στιγμή
- Complexity - η αύξηση της ακρίβειας στην ορατότητα μπορεί να οδηγήσει σε μεγάλο αριθμό μικρών τμημάτων πολυγώνων

Painters Algorithm (Image Precision)

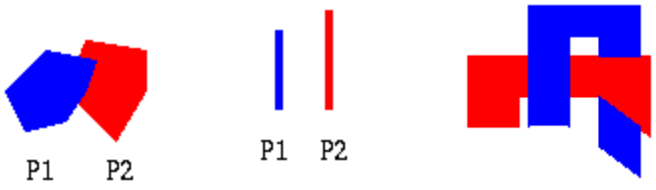


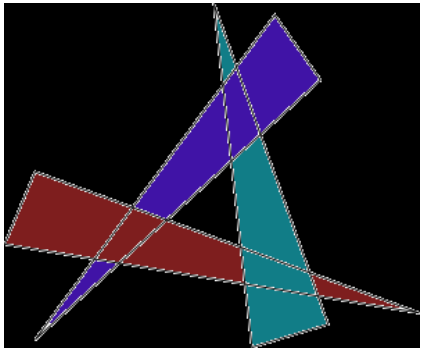
- Αλγόριθμος:
  - Επέλεξε μία διάταξη των πολυγώνων με βάση ένα χαρακτηριστικό (πχ. Βάθος ενός σημείου των πολυγώνων)
  - Απεικόνισε τα πολύγωνα με βάση την διάταξη, το βαθύτερο πρώτο
- Απεικονίζουμε τα κοντινότερα αντικείμενα πάνω από βαθύτερα
- Θέματα:
  - Λειτουργεί για μερικές γεωμετρίες (2.5D - πχ VLSI)
  - Δεν λειτουργεί για τις περισσότερες γεωμετρίες - αδυναμία δημιουργίας διάταξης



The Z-buffer (Image Precision)

- Για κάθε pixel της οθόνης, υπάρχουν τουλάχιστον 2 buffers
  - Color buffer αποθηκεύει το τρέχον χρώμα του κάθε pixel
  - Z-Buffer αποθηκεύει σε κάθε pixel το βάθος (z) του κοντινότερου αντικειμένου που έχουμε δει
- Αρχικοποίηση του buffer σε μία τιμή που αντιστοιχεί στο πιο απομακρυσμένο σημείο (z=1.0)
- Καθώς ένα πολύγωνο προστίθεται, υπολογίζεται το βάθος του κάθε pixel που πρέπει να σημειωθεί αυτό το πολύγωνο
  - if depth < z-buffer depth, γέμισε το pixel color και το νέο depth
  - else αγνόησε





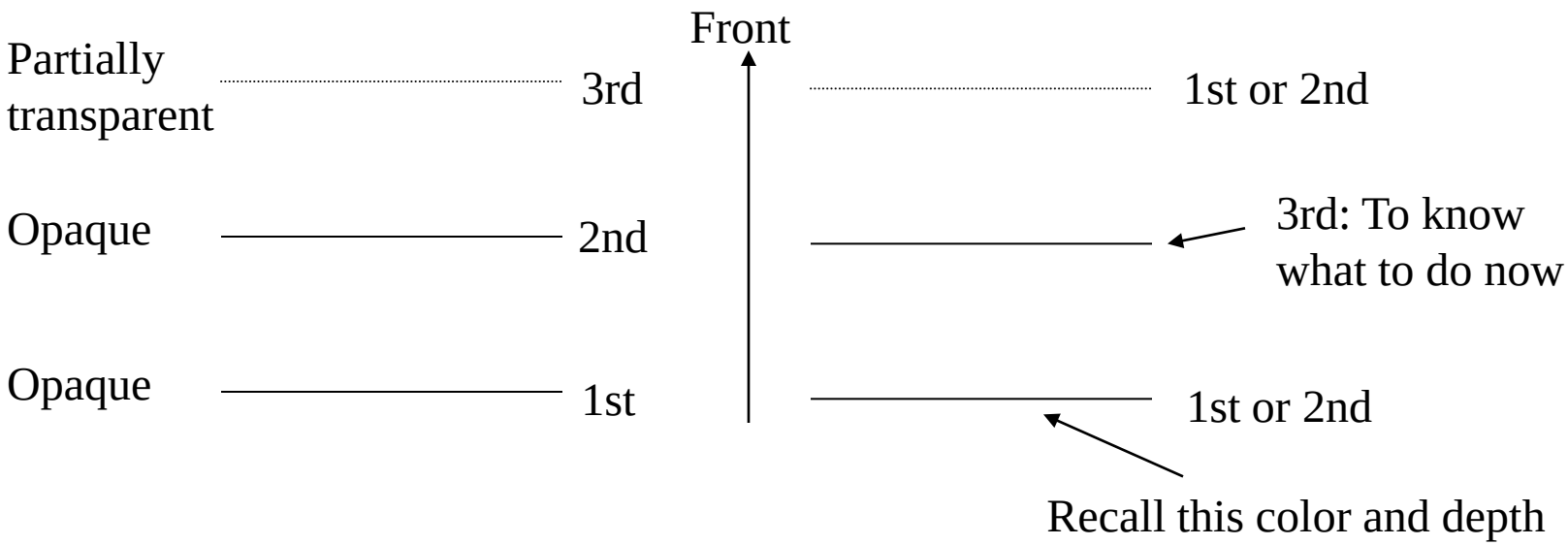
The Z-buffer

- Πλεονεκτήματα
  - Απλός και καθολικής χρήσης στο hardware
  - Υπολογισμός των τιμών βάθους είναι απλός

- Μειονεκτήματα
  - Απεικονίζει άχρηστα στοιχεία
  - Depth quantization errors
  - Δεν είναι εύκολο να υλοποιηθεί η διαφάνεια (transparency) ή filtering για anti-aliasing (που χρησιμοποιεί πληροφορία για μερικώς επικαλυπτόμενα πολύγωνα)

Z-Buffer και Transparency

- Η απεικόνιση γίνεται από πίσω προς τα εμπρός
- Αλλιώς πρέπει να αποθηκεύεται το πρώτο opaque polygon πίσω από το διάφανο



OpenGL Depth Buffer

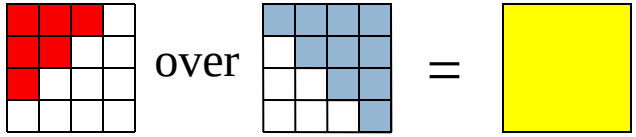
- OpenGL χρησιμοποιεί έναν depth buffer για την ορατότητα
- `glEnable(GL_DEPTH_TEST)`
- `glClear(GL_DEPTH_BUFFER_BIT)`
  - color και depth: `glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)`
- Ο αριθμός των bit που χρησιμοποιούνται για τις τιμές του βάθους είναι δυνατό να προσδιοριστεί (windowing system dependent, και hardware μπορεί να το περιορίσει ανάλογα με την διαθέσιμη μνήμη)
- Επίσης η συνάρτηση σύγκρισης βάθους μπορεί να προσδιορισθεί `glDepthFunc(…)`

A-buffer (Image Precision) (1)

- Χειρίζεται τις διάφανες επιφάνειες και χειρίζεται το anti-aliasing
- Για κάθε pixel, διατηρεί έναν δείκτη στη λίστα των πολυγώνων που είναι διατεταγμένα με βάση το βάθος, και ένα sub-pixel overage mask για κάθε πολύγωνο
- Πίνακας από bits υποδεικνύει ποια μέρη του pixel καλύπτονται
- Αλγόριθμος: Όταν ζωγραφίζεται ένα pixel:
  - if polygon είναι opaque ΚΑΙ επικαλύπτει το pixel, βάλτο στην λίστα απομακρύνοντας τα βαθύτερα πολύγονα
  - if polygon είναι transparent Ή μερικώς επικαλύπτει το pixel, βάλτο στην λίστα , μην απομακρύνεις τα βαθύτερα pixel

The A-buffer (2)

- Algorithm: Rendering pass
  - Για κάθε pixel, διάτρεξε τον buffer χρησιμοποιώντας τα χρώματα (polygon colors) και τις μάσκες επικάλυψης (coverage masks) για την σύνθεση:
- Πλεονεκτήματα



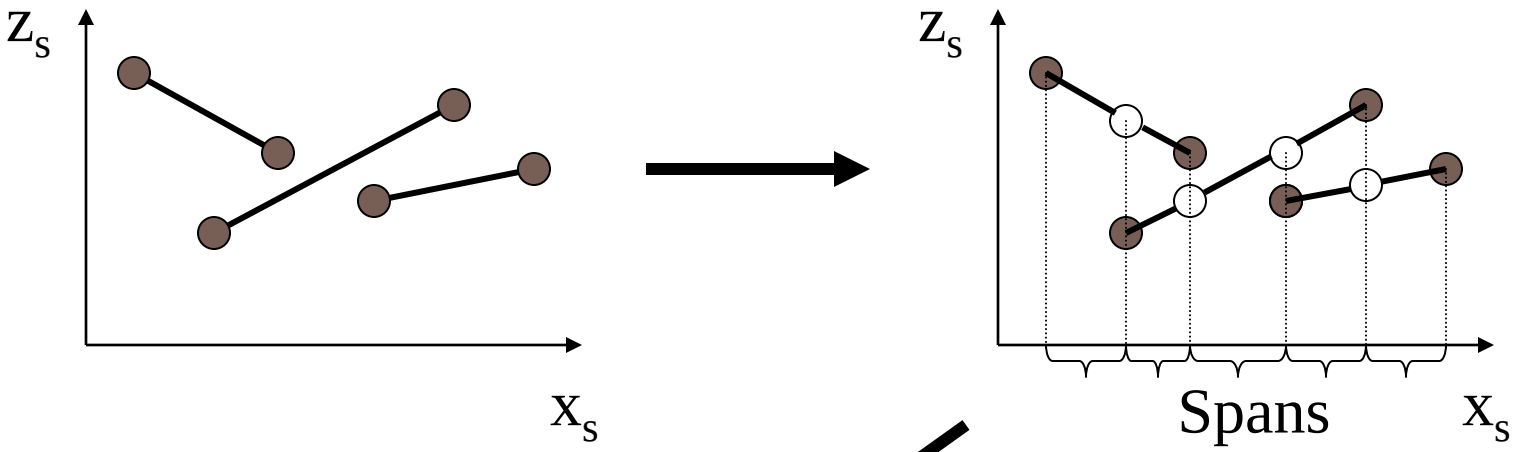
  - Υπερσύνολο του Z-buffer
  - Coverage mask μπορεί να χρησιμοποιηθεί και σε άλλους αλγόριθμους
- Μειονεκτήματα
  - Όχι στο hardware, και αργό στο software
  - z-buffer: Over-rendering και depth quantization προβλήματα

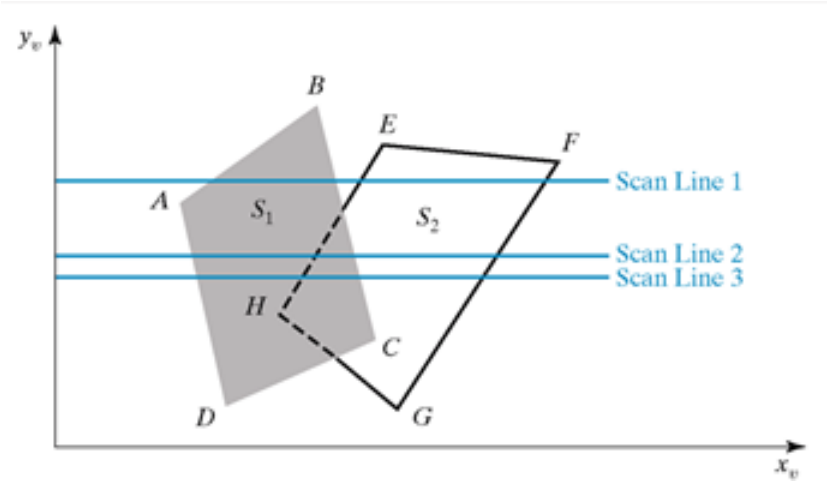
➔ χρησιμοποιείται μόνο σε high quality rendering σχεδιαστικά εργαλεία

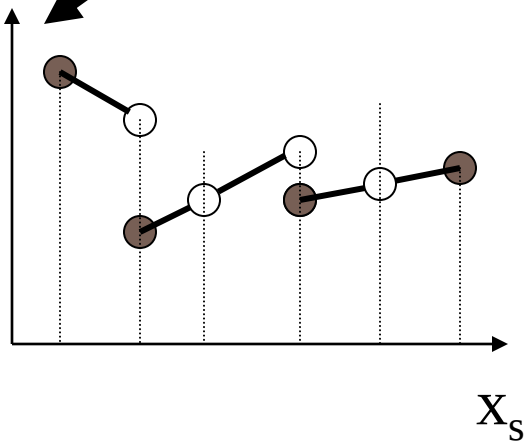
Scan Line Algorithm (Image Precision) (1)

- Υπολογίζει και συγκρίνει βάθη με χρήση scan lines
- Υποθέτουμε ότι τα πολύγωνα δεν τέμνονται
  - Ίσος μόνο στις πλευρές ή στις κορυφές
- Παρατήρηση: κατά μήκος κάποιας scan line, το ορατό πολύγωνο μπορεί να αλλάζει μόνο σε μία ακμή
- Διατηρούνται
  - Πίνακας ακμών (άκρα, κλίση, δείκτες στις επιφάνειες)
  - Πίνακας επιφανειών (επιφάνεια, ιδιότητες, δείκτες στις ακμές)
- Αλγόριθμος
  - Γέμισμα όλων των πολυγώνων ταυτόχρονα
  - Σε κάθε scan line, βάλε όλες τις ακμές που τέμνουν την scan line
  - Διατήρησε το τρέχον βάθος στο τρέχον pixel - χρησιμοποιείται για την απόφαση ποιο είναι μπροστά από τμήμα γεμίσματος

Scan Line Algorithm (2)



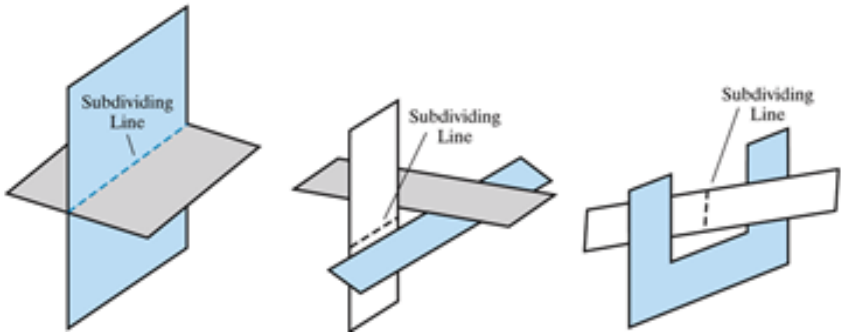




Where polygons overlap, draw front polygon

Scan Line Algorithm (3)

- Πλεονεκτήματα
  - Απλότητα
  - Λιγότερα σφάλματα quantization errors
  - Δεν over-render (each pixel only drawn once)
  - Μπορεί να εκτελέσει Filter anti-aliasing (έχει την πληροφορία ποιο πολύγωνο καλύπτει ποιο pixel)
- Μειονέκτημα
  - Η τομή έχει δυσκολίες

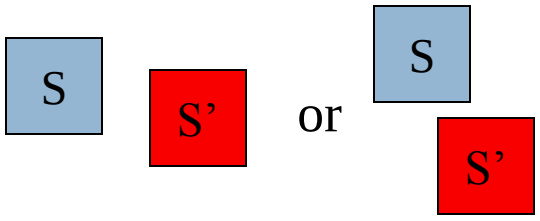


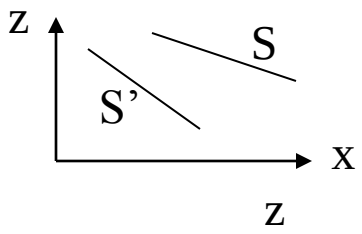
Depth Sorting (Object Precision, view space) (1)

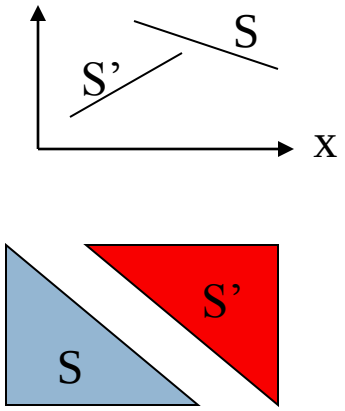
- Ομοιότητα με τον *list-priority* algorithm
- Διάταξε τα πολύγωνα ως προς το βάθος για ένα σημείο
- Απεικόνισε από τα πίσω προς τα εμπρός Render from back to front (αλλαγή διάταξη δυναμικά)
- Μέθοδος: Για την επιφάνεια S με μεγαλύτερο βάθος
  - Εάν δεν υπάρχει επικάλυψη με άλλα πολύγωνα στο βάθος, scan convert
  - Αλλιώς, δοκιμή για επικάλυψη στο image plane
    - Εάν όχι, scan convert και επεξεργασία του επόμενου πολυγώνου
  - Εάν S, S' επικαλύπτονται και στο βάθος και στο image plane, άλλαξε την σειρά και προσπάθησε ξανά
  - Εάν S, S' έχουν εναλλαχθεί ξανά διέσπασε και ξανα-εισήγαγε

Depth Sorting (2)

- Έλεγχος επικαλύψεων: ξεκίνα απεικόνιση όταν ισχύει η πρώτη συνθήκη:
  - x-extents or y-extents do not overlap
  - S is behind the plane of S'
  - S' is in front of the plane of S
  - S and S' do not intersect in the image plane

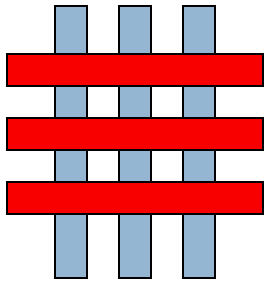






Depth sorting

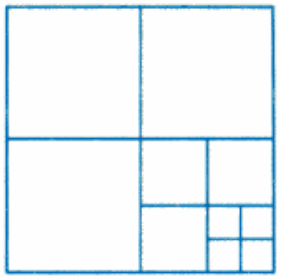
- Πλεονεκτήματα
  - Filter anti-aliasing λειτουργεί
  - Δεν υπάρχει depth quantization error
    - Συγκρίσεις βάθους γίνονται στο high-precision view space
- Μειονεκτήματα
  - Over-rendering
  - Μπορεί να υπάρξει μεγάλος αριθμός διασπάσεων $\Omega(n^2)$ τμήματα από n πολύγωνα



Area Subdivision

- Προσπαθεί να εκμεταλλευτεί τη χωρική συνοχή → μικρές περιοχές μίας εικόνας είναι πιο πιθανό να καλύπτονται από ένα πολύγωνο

- Τρεις εύκολες περιπτώσεις
  1. Ένα πολύγωνο είναι εντελώς μέσα στη περιοχή και μπροστά από κάθε άλλο μέσα στην περιοχή
  2. Τίποτα δεν προβάλλεται στην περιοχή
  3. Μόνο ένα πολύγωνο υπάρχει μέσα, τέμνει ή περιβάλλει την περιοχή



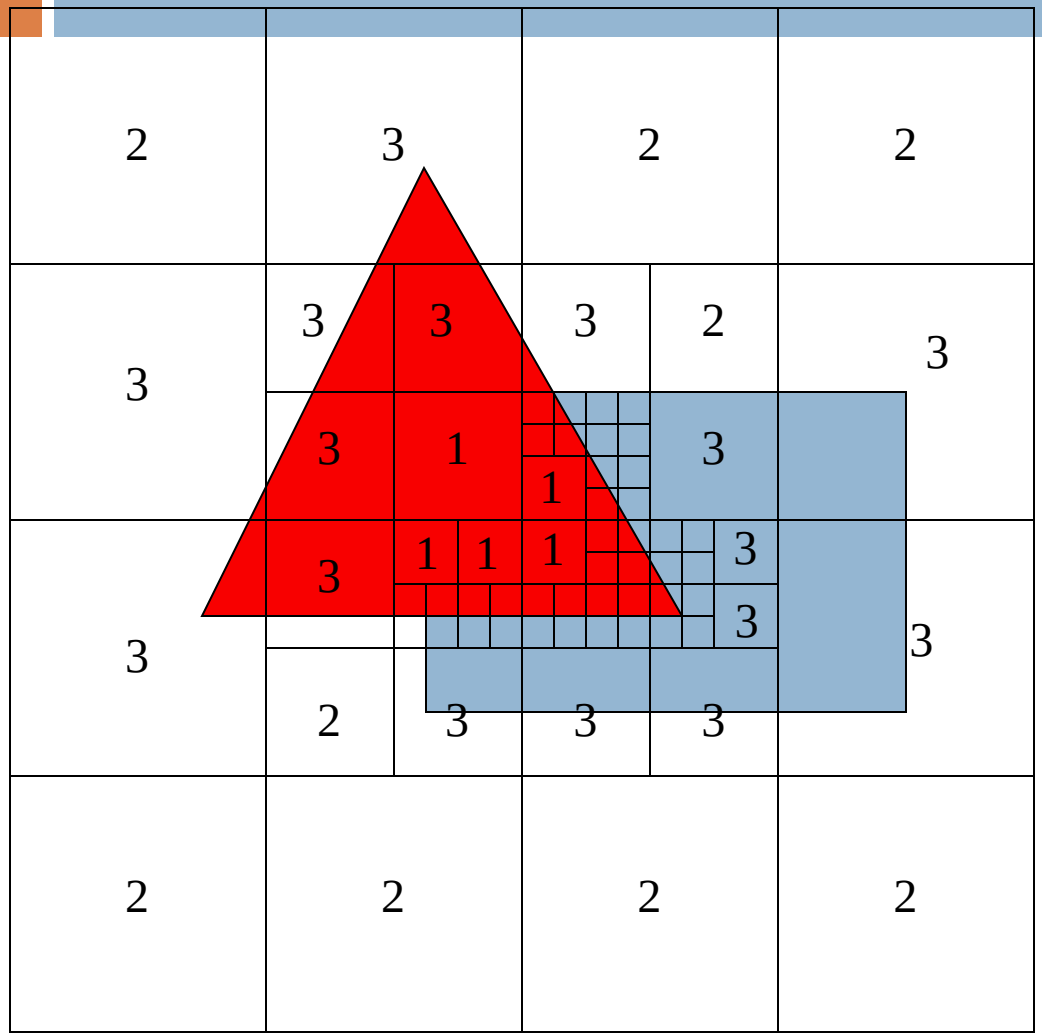
Warnock's Area Subdivision

(Image Precision)

- Έναρξη με όλη την εικόνα
- Αν ισχύσει μία από τις εύκολες περιπτώσεις ζωγράφισε ότι υπάρχει μπροστά
- Αλλιώς, χώρισε την περιοχή και επανέλαβε αναδρομικά
- Εάν η περιοχή γίνει pixel, επέλεξε το πολύγωνο με το μικρότερο βάθος

- Πλεονεκτήματα
  - Όχι over-rendering
  - Καλό Anti-aliasing - πηγαίνεις στην αναδρομή βαθύτερα για sub-pixel information
- Μειονεκτήματα
  - Οι έλεγχοι είναι πολύπλοκοι και αργοί

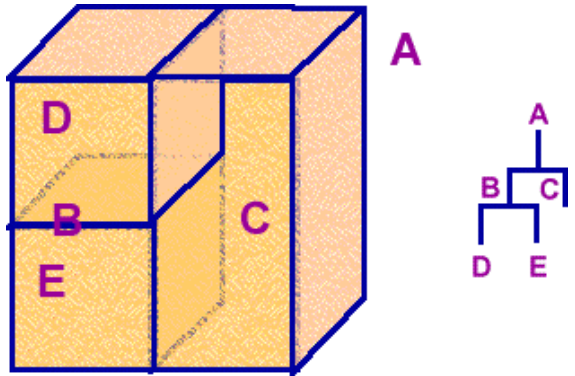
Warnock's Algorithm



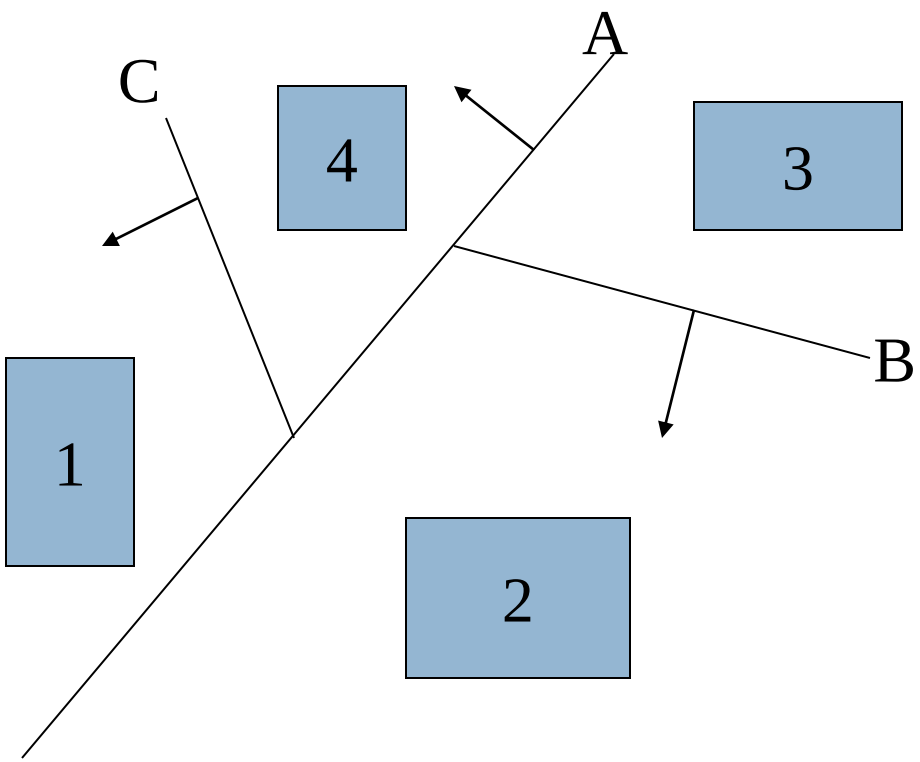
- Οι περιοχές χαρακτηρίζονται ανάλογα με την κατηγορία
  1) One polygon in front
  2) Empty
  3) One polygon inside, surrounding or intersecting
- Μικρές περιοχές δεν παίρνουν ετικέτα
- Rendering algorithm + HSR algorithm
  - Assuming you can draw squares

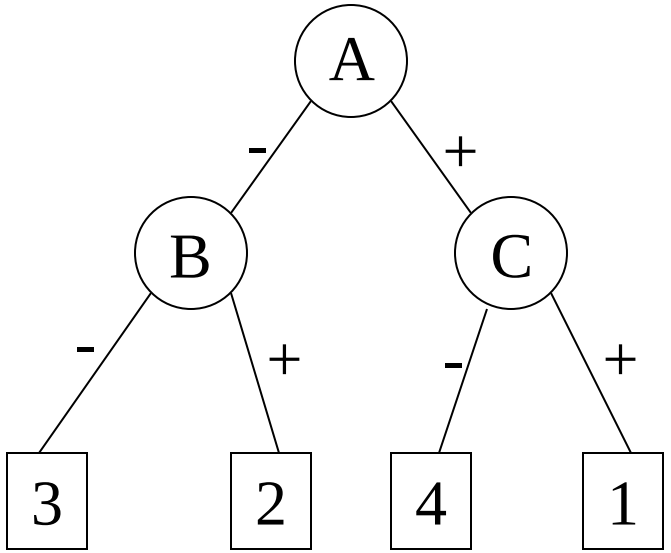
BSP-Trees (Object Precision)

- Κατασκεύασε ένα *binary space partition* tree
  - Η δομή δίνει μία «σειρά απεικόνισης»
  - Ένας list-priority algorithm
- Η δομή διαχωρίσει τον χώρο με 3D επιφάνειες
  - Ο χώρος κατακερματίζεται σε κλειστά κελιά
  - Κάθε κελί είναι μία τομή όλες τα ημί-χωρία στο μονοπάτι του
- Χρησιμοποιείται για να μοντελοποιήσει το σχήμα αντικειμένων και σε άλλους αλγορίθμους ορατότητας
  - BSP visibility είναι διαφορετική για τα παιχνίδια



BSP-Tree Παράδειγμα



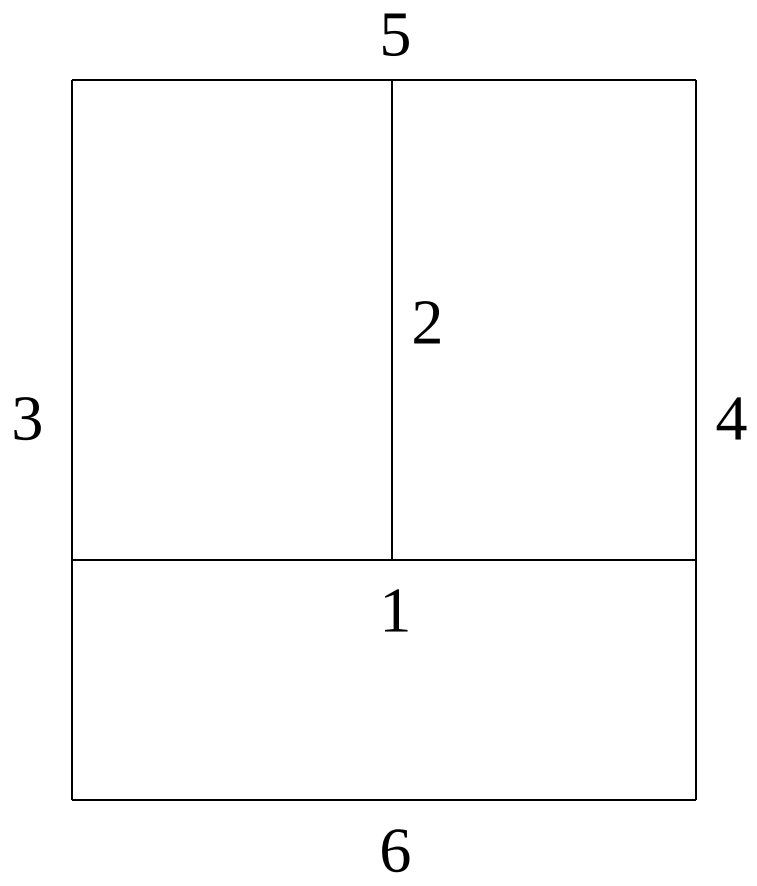


Κατασκευή BSP-Trees

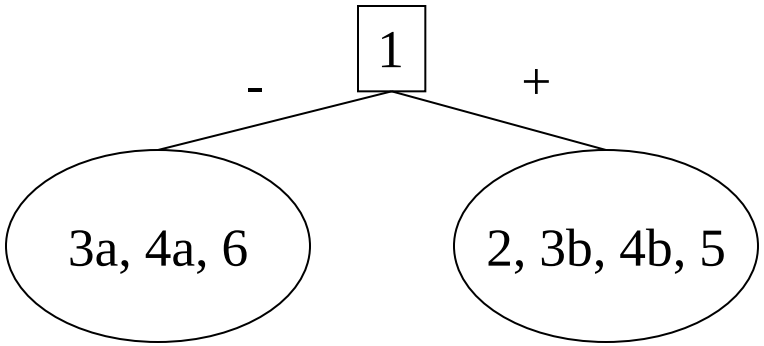
- Επιλογή ενός τυχαίου πολύγωνου
- Διάσπαση του κελιού με χρήση μίας επιφάνειας
  - Ίσως χρειαστεί η διάσπαση και του πολυγώνου
- Συνέχισε έως ότου κάθε κελί έχει μόνο ένα τμήμα πολυγώνου
- Επιφάνειες διαχωρισμού μπορούν να επιλεγούν με διάφορους τρόπους (ποιο είναι βέλτιστο?)
  - Optimal means minimum number of polygon fragments in a balanced tree

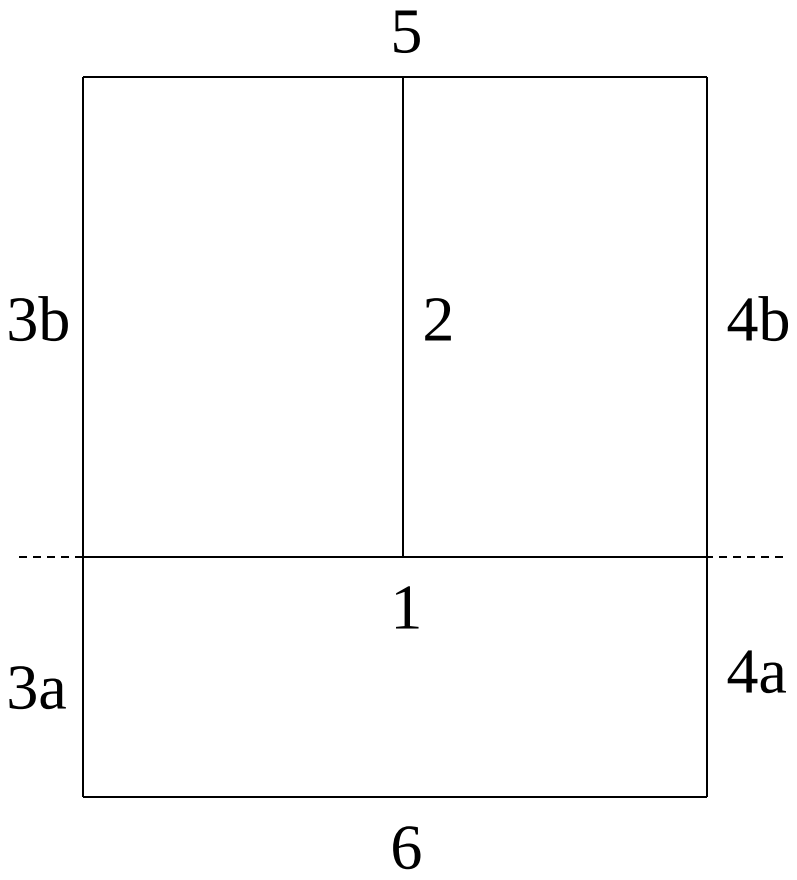
BSP Παράδειγμα (1)

- BSP tree, 2D, ένα κτήριο 3 δωματίων 3
  - Χωρίς πόρτες
- Αριθμοί - η σειρά διαχωρισμού του χώρου

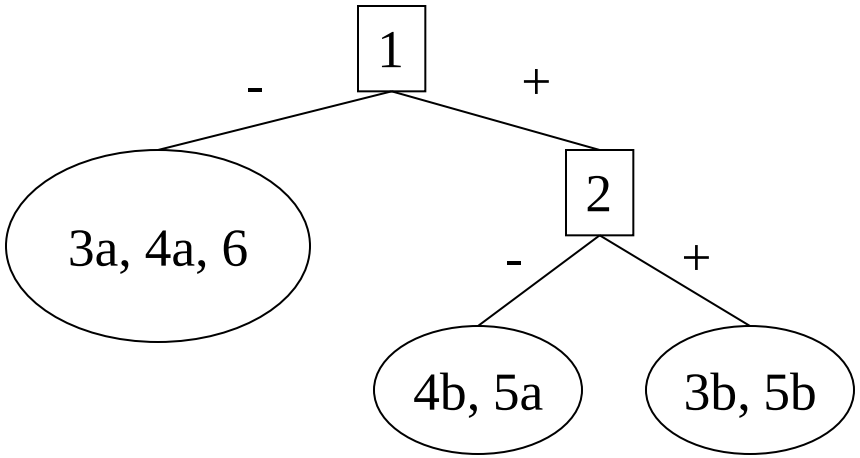


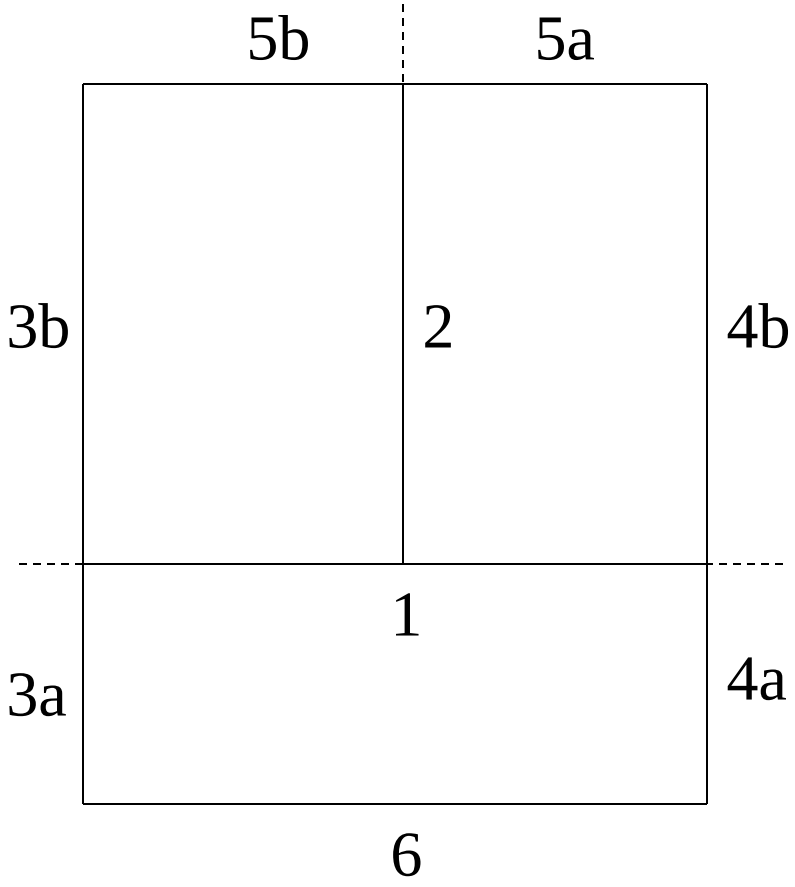
BSP Παράδειγμα (2)



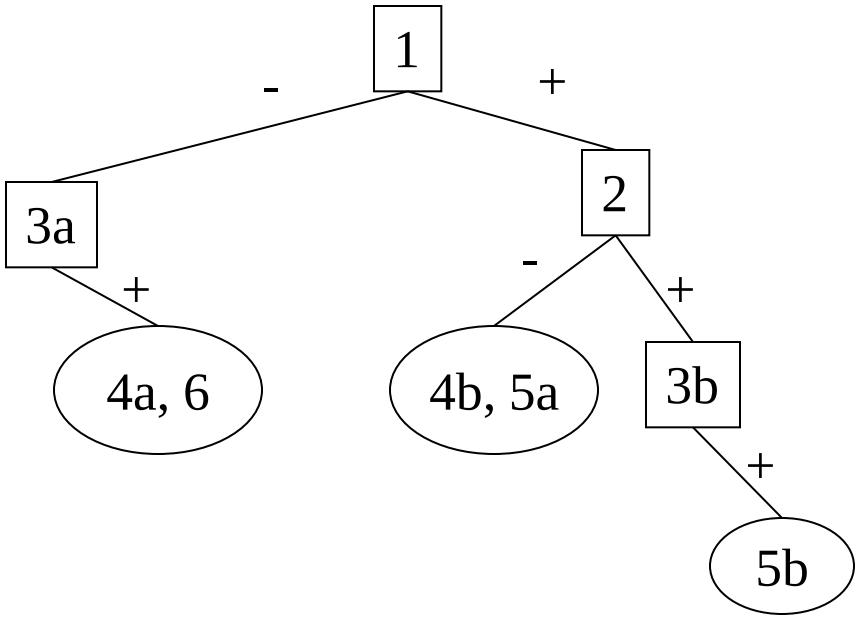


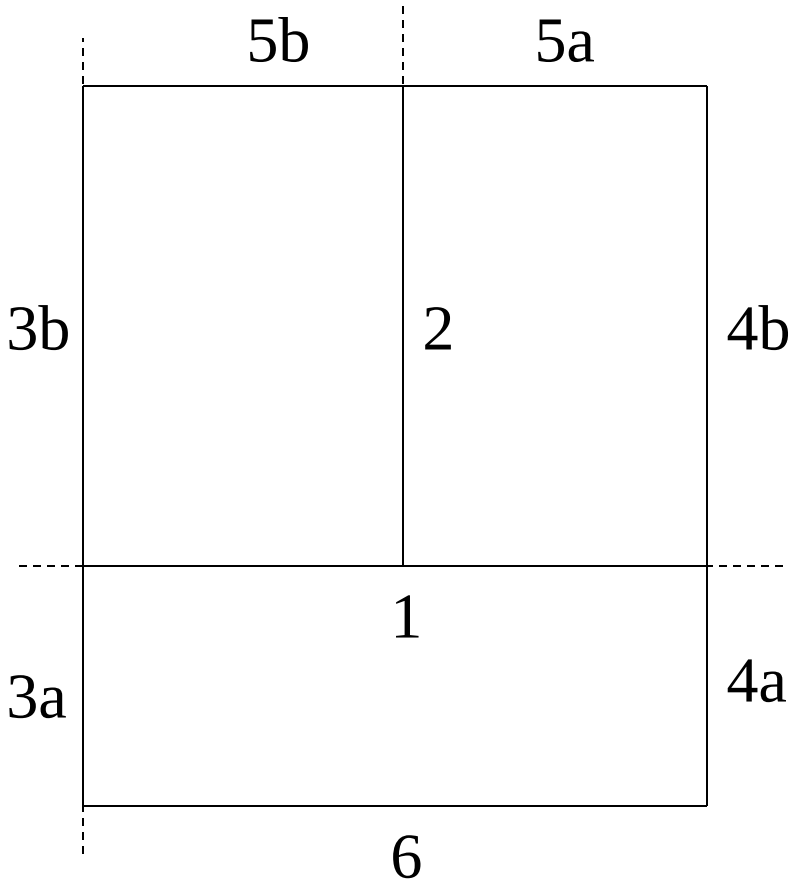
BSP Παράδειγμα (3)



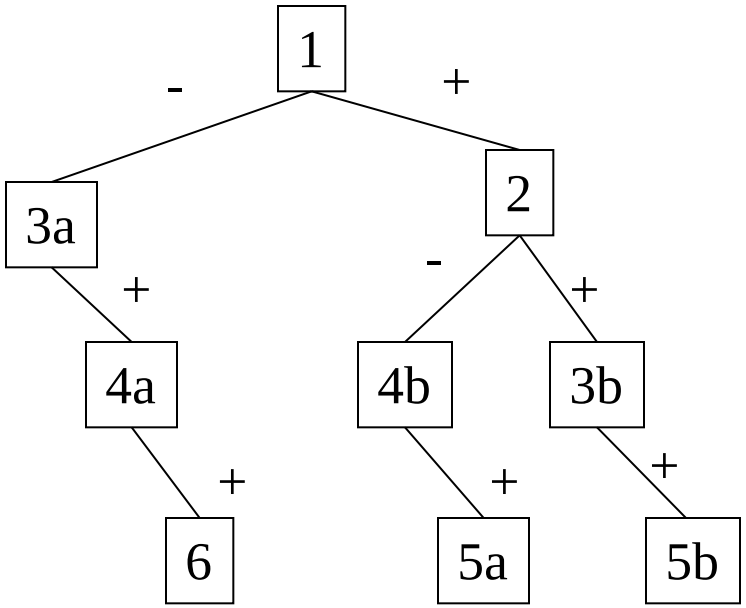


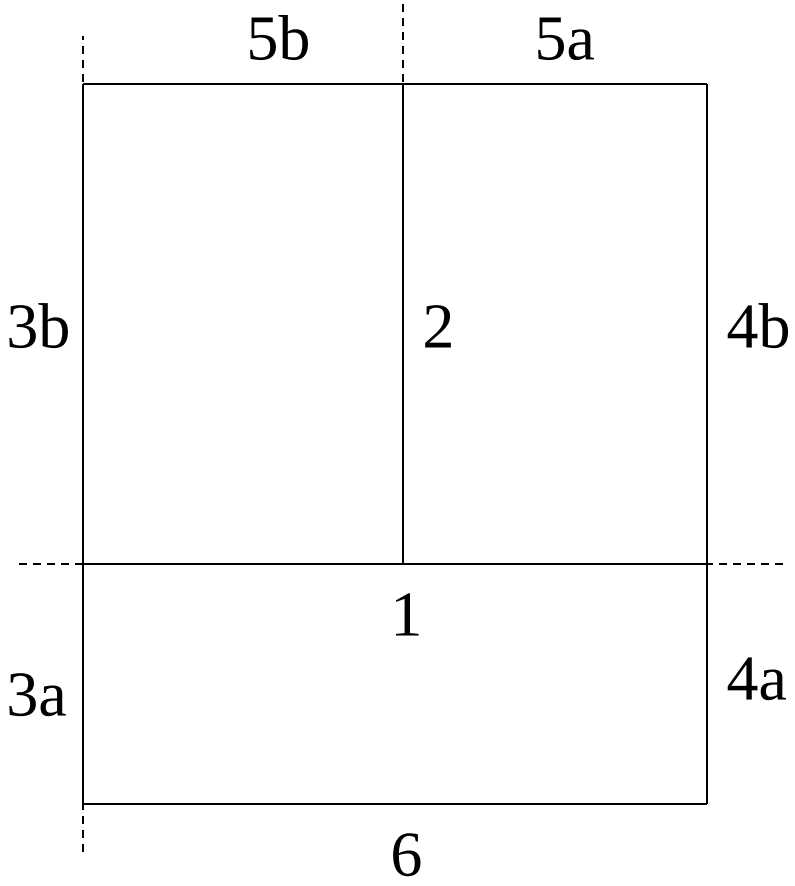
BSP Παράδειγμα (4)





BSP Παράδειγμα (5)

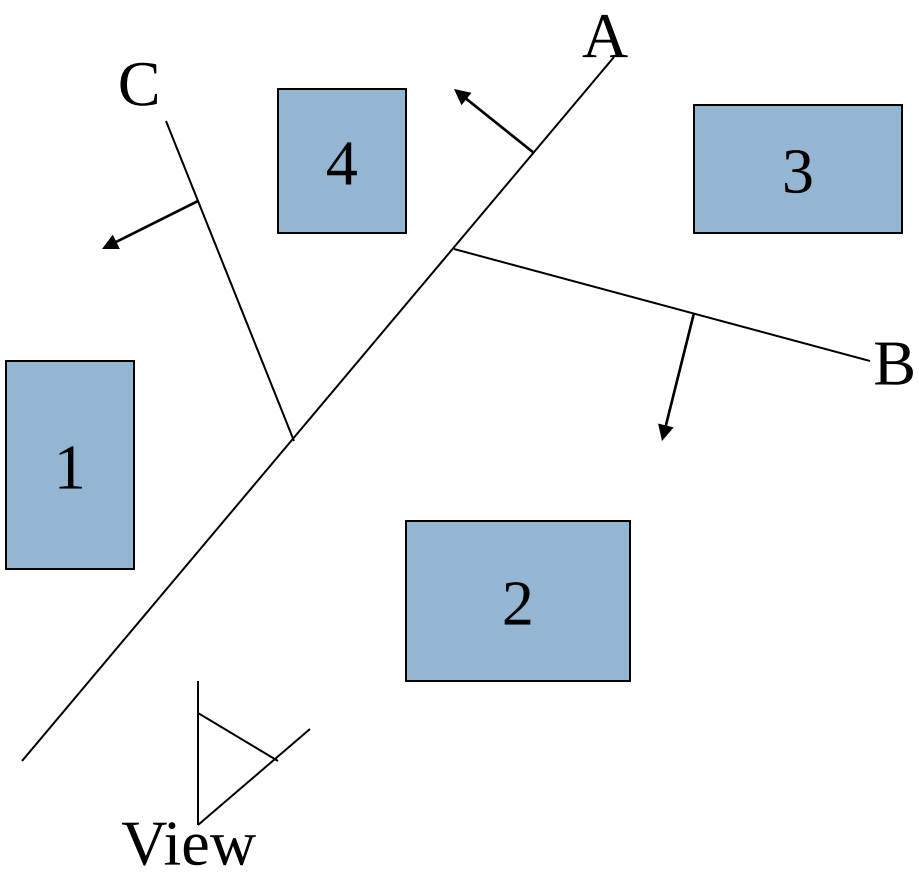


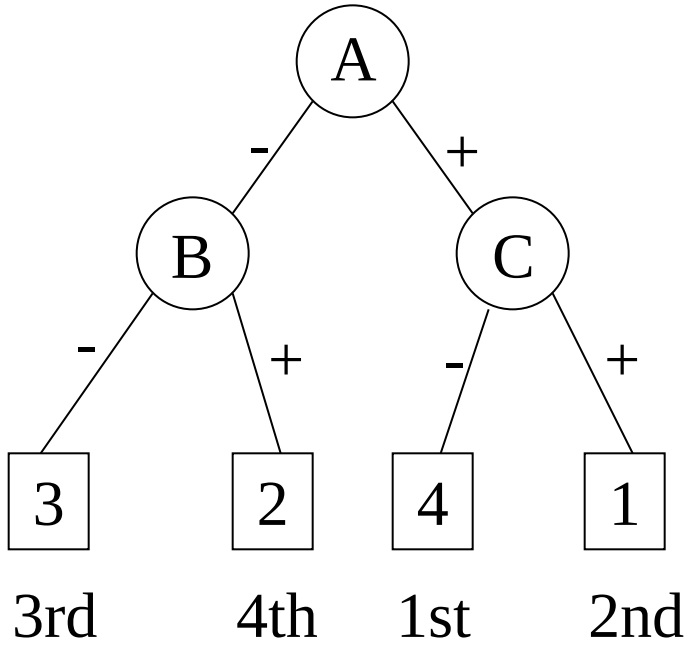


BSP-Tree Rendering

- Παρατήρηση: Αντικείμενα στην απέναντι πλευρά από μία επιφάνεια διαχωρισμού από το σημείο παρατήρησης δεν κρύβουν αντικείμενα από την ίδια πλευρά…
- Ο αλγόριθμός αναδρομικά κατέρχεται την δομή
- Σε κάθε κόμβο (for back to front rendering):
  - Κατέβα κάτω στην πλευρά του υπο-δένδρου που δεν περιέχει το σημείο παρατήρησης
    - Test viewpoint against the split plane to decide which tree
  - Ζωγράφισε το πολύγωνο στην επιφάνεια διαχωρισμού
    - Paint over whatever has already been drawn
  - Κατέβα κάτω στο άλλο υπο-δένδρο

BSP-Tree Rendering Παράδειγμα





BSP-Tree Rendering

- Πλεονεκτήματα
  - Κάθε δένδρο δουλεύει για οποιοιδήποτε σημείο παρατήρησης
  - Filter anti-aliasing και transparency δουλεύουν
    - Πίσω προς τα εμπρός σύνθεση …
  - Είναι δυνατό εμπρός προς τα πίσω σύνθεση χωρίς να ζωγραφίζονται τα πίσω πολύγωνα
- Μειονεκτήματα
  - Μικρά τμήματα πολυγώνου
  - Over-rendering

Exact Visibility

- Ένας *exact visibility* αλγόριθμος σου επιστρέφει μόνο ότι είναι ορατό και μόνο αυτό
  - Όχι over-rendering
  - Πχ . Warnock's algorithm
- Δύσκολα να γίνει αυτό αποδοτικά στην πράξη
  - Μικρά λεπτομερή αντικείμενα το κάνουν δύσκολο …
- Αλλά , σε όγκους ή απλοϊκά αντικείμενα, exact visibility γίνεται πολύ αποδοτικά …

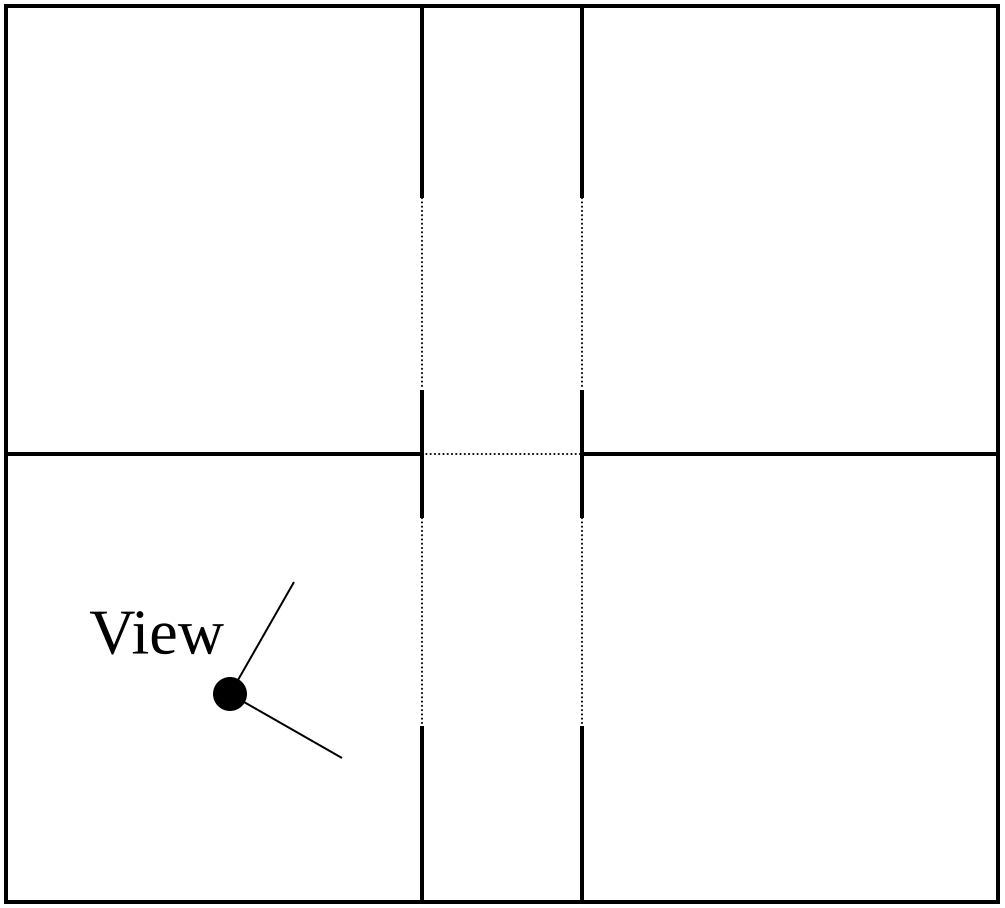
Cells και Portals

- Ας υποθέσουμε ότι ο κόσμος μπορεί να διαχωριστεί σε *cells*
  - Απλά σχήμα
  - Δωμάτια κλπ
- Ορίζονται *portals ως τα διάφανα όρια μεταξύ των* cells
  - Πόρτες, Παράθυρα κλπ
- Σε έναν κόσμο σαν και αυτό είναι δυνατό να προσδιοριστεί ποια μέρη του δωματίου είναι ορατά ακριβώς

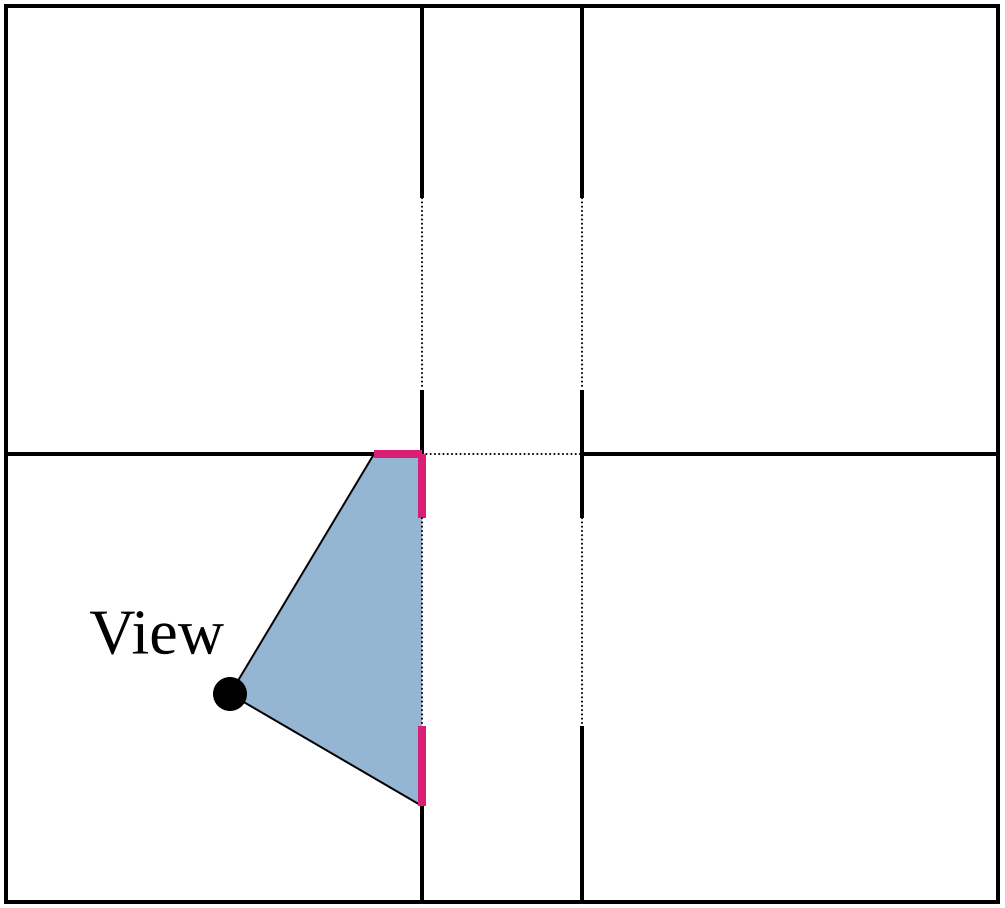
Cell and Portal Visibility

- Start in the cell containing the viewer, with the full viewing frustum
- Render the walls of that room and its contents
- Recursively clip the viewing frustum to each portal out of the cell, and call the algorithm on the cell beyond the portal

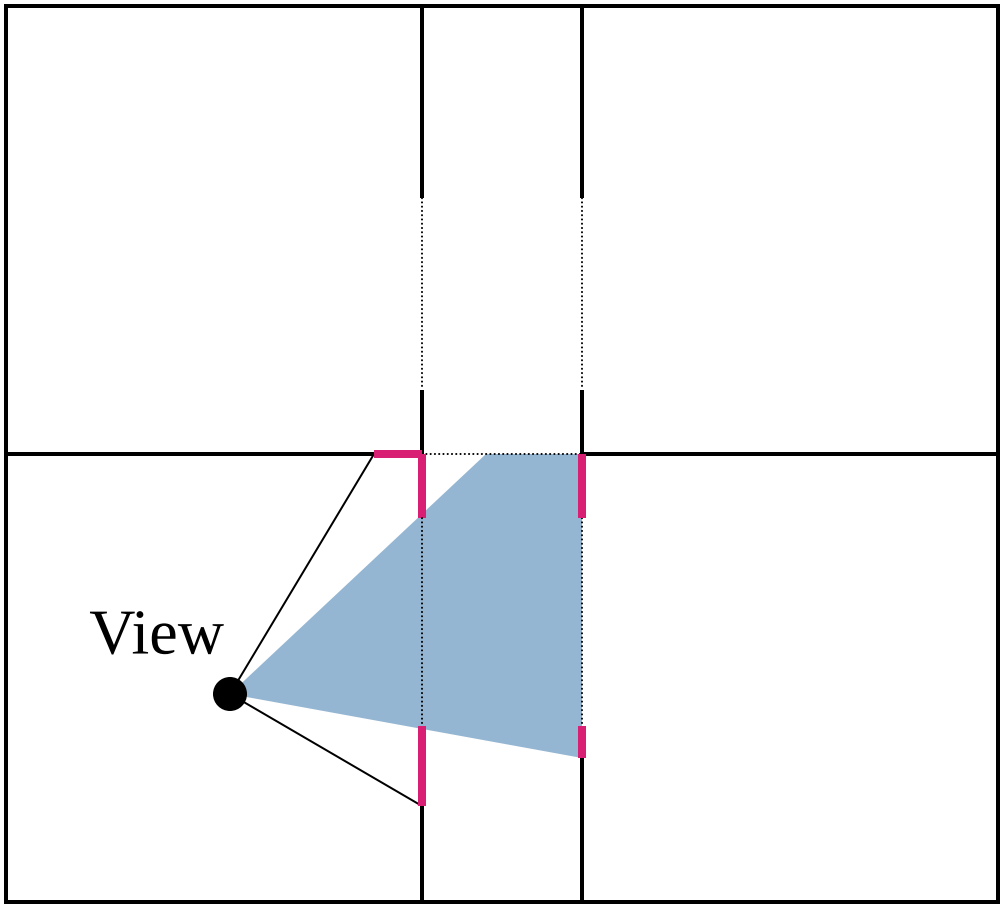
Cell-Portal Παράδειγμα (1)



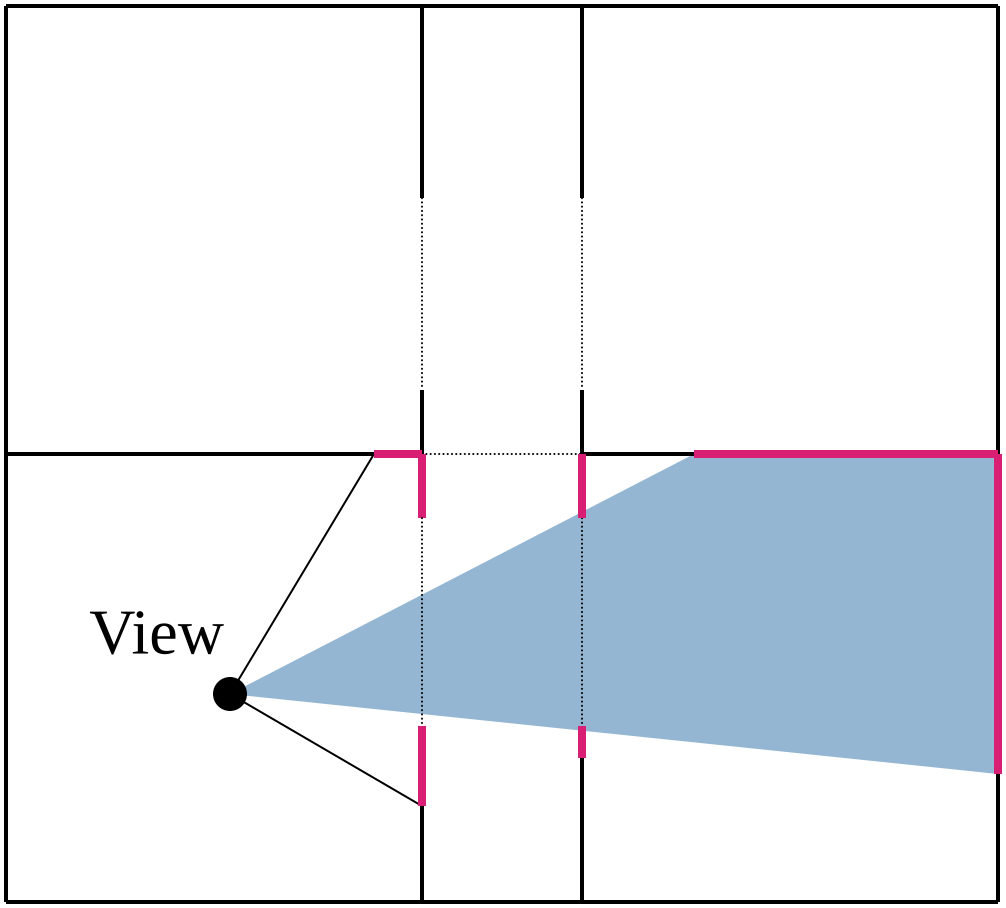
Cell-Portal Παράδειγμα (2)



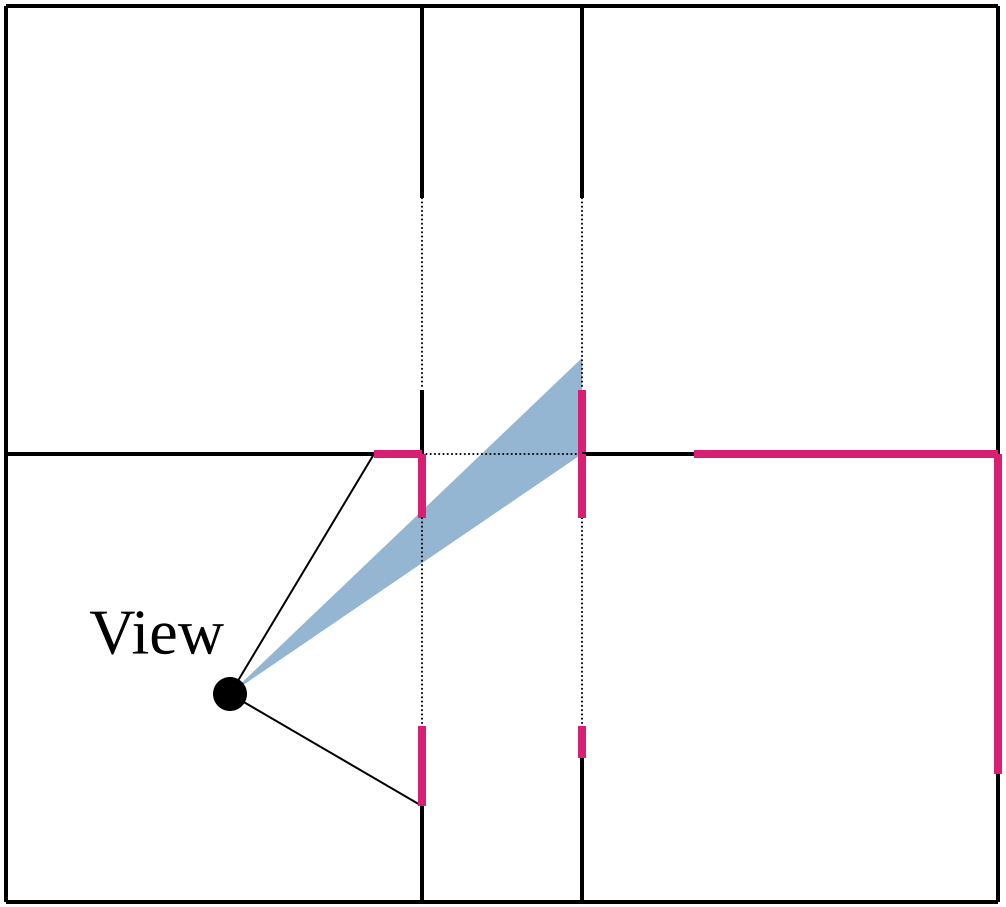
Cell-Portal Παράδειγμα (3)



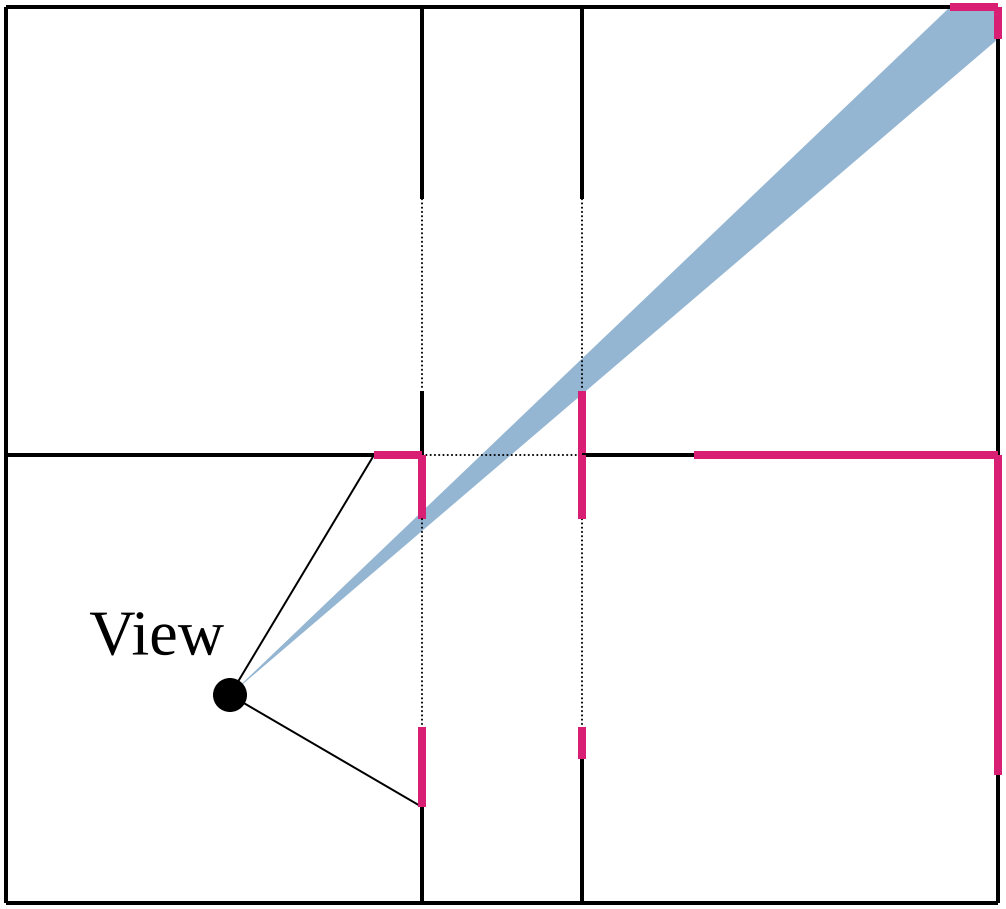
Cell-Portal Παράδειγμα (4)



Cell-Portal Παράδειγμα (5)



Cell-Portal Παράδειγμα (6)



Cell-Portal Πράξεις

- Πρέπει να αποκοπούν πολύγωνα (clipping) στο τρέχον view frustum (και όχι στο αρχικό)
  - Είναι δυνατό με hardware clipping planes, σε μερικές αρχιτεκτονικές
- Αποκόπτεται το αρχικό view frustum σε σχέση με το portal
  - Είναι πιο εύκολο να αποκοπεί το portal στο frustum, και στη συνέχεια αντιστοίχιση του frustum στο clipped portal

Cell-Portal Ιδιότητες

- Πλεονεκτήματα
  - Πολύ αποδοτικό - επεξεργασία μόνο cells τα οποία είναι visible: visibility culling
  - Εύκολη παραλλαγή για approximate visibility - απεικόνισε όλα τα μερικώς ορατά cells, και το depth buffer τα διαχωρίζει
  - Χειρίζεται καθρέπτες - ανάκλαση (flip world) κόσμου γύρο από τον καθρέπτη και έστω ότι είναι αυτός portal

- Μειονεκτήματα
  - Περιορίζεται σε κόσμους με αυστηρή cell/portal δομή