Đồ họa máy tính

Xác định mặt hiện
(Visible surface determination)

Sự hữu hình của các đối tượng cơ bản

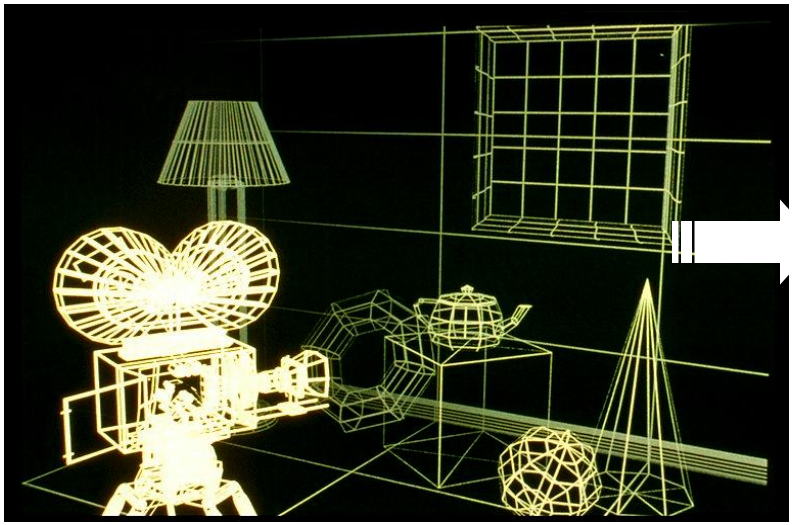
- Chúng ta không muốn phí thời gian để hiển thị những đối tượng không đóng góp vào bức ảnh cuối cùng.
- Một đối tượng có thể không hữu hình vì 3 lý do:
  - Nằm ngoài vùng hiển thị
  - Quay vào trong (*back-facing*)
  - Bị che bởi các đối tượng khác gần người quan sát hơn
- Làm thế nào để loại bỏ chúng một cách hiệu quả?
- Làm thế nào để xác định chúng một cách hiệu quả?

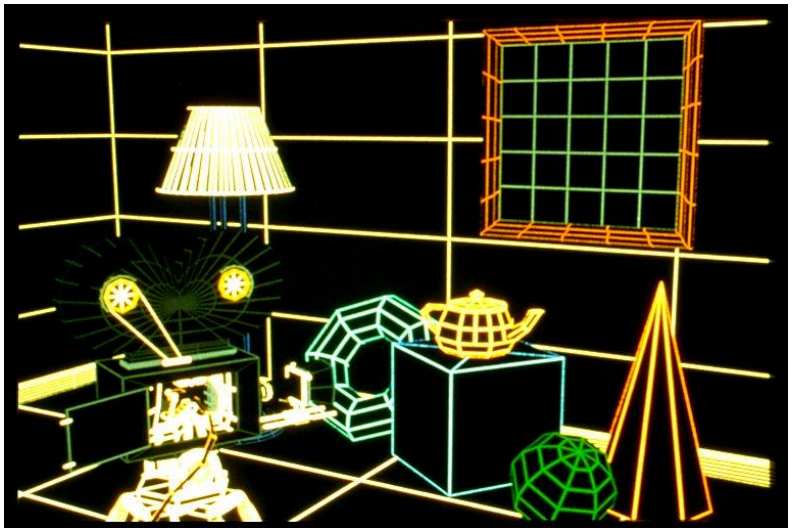
Vấn đề hữu hình

Hai vấn đề còn lại:

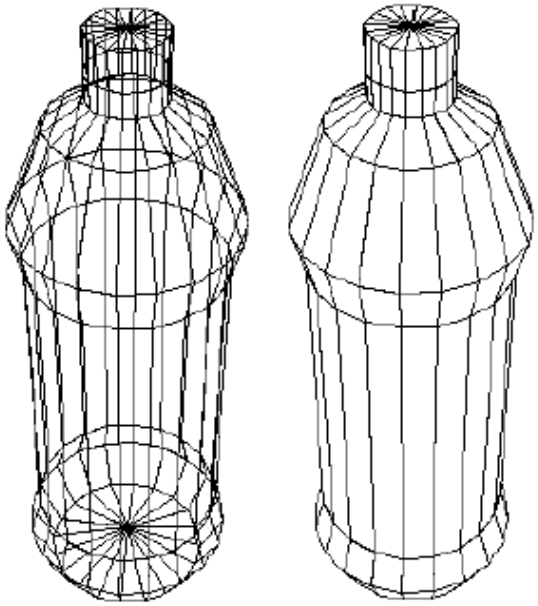
(Chúng ta đã làm quen với clipping)

- Loại bỏ các bề mặt hướng ra phía khác so với người quan sát.
- Loại bỏ các bề mặt che bởi các đối tượng gần hơn.





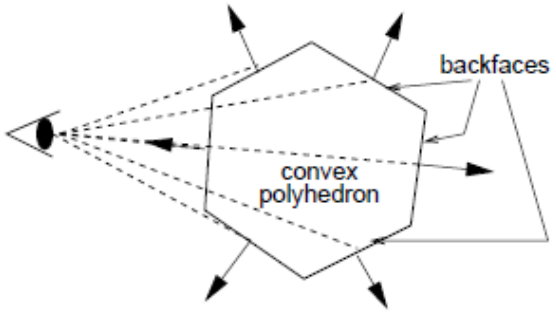
Xác định mặt hiện vs. Loại bỏ mặt khuất

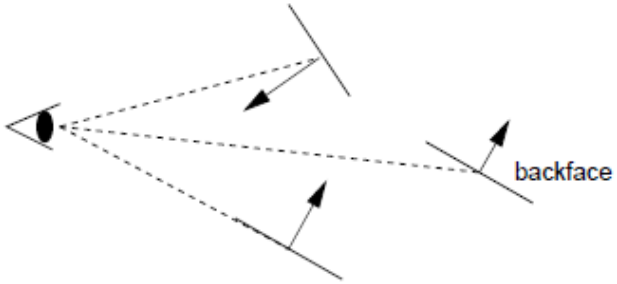


Các thuật toán mặt hiện

3 dạng của các thuật toán xác định mặt hiện

- Chính xác theo đối tượng (object precision)
- Chính xác theo ảnh (image precision)
- Ưu tiên theo danh sách (list priority)





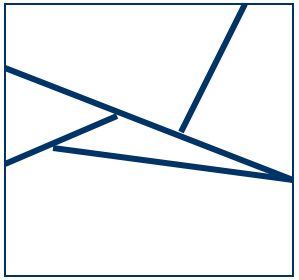
Các thuật toán mặt hiện

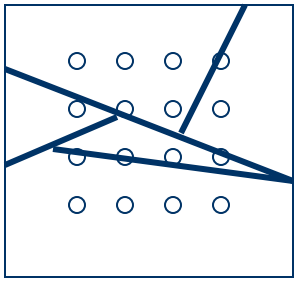
Loại bỏ/Xác định Mặt/đoạn Ẩn/hiện

- Yêu cầu
  - Có thể xử lý các tập đối tượng khác nhau
  - Có thể xử lý một lượng lớn các đại lượng hình học

Phân loại: Sutherland, Sproull, Schumacher (1974):

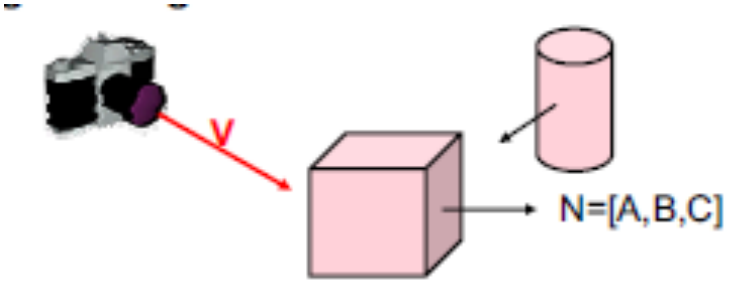
- Không gian vật thể
  - Tính toán hình học liên quan đến đa giác
  - Độ chính xác số thực
  - Thường xử lý cảnh vật theo thứ tự các vật thể
- Không gian ảnh
  - Visibility at pixel samples
  - Độ chính xác số nguyên
  - Thường xử lý cảnh vật theo thứ tự ảnh

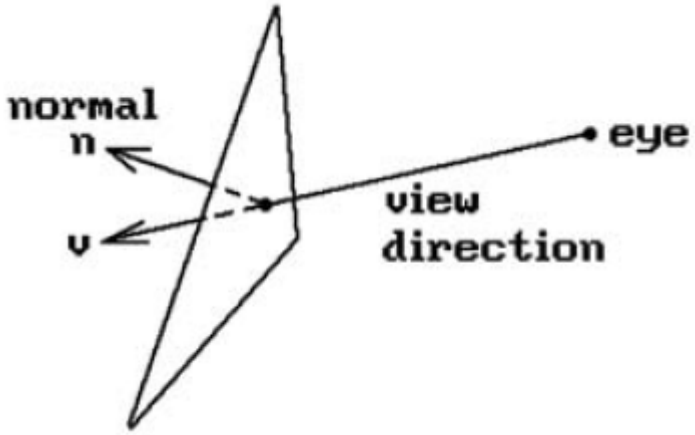




Loại bỏ mặt quay vào trong

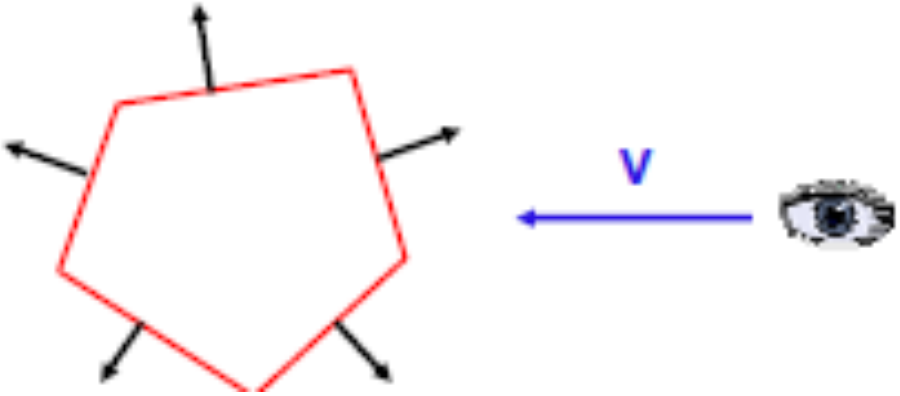
- Với sự phát triển của các thiết bị hiển thị dẫn đến nhu cầu thể hiện các vật thể một cách thực tế hơn, đòi hỏi các mô hình có rất nhiều đa giác.
- Từ đó dẫn đến nhu cầu phát triển các thuật toán để loại bỏ mặt ẩn (*hidden surface removal*).





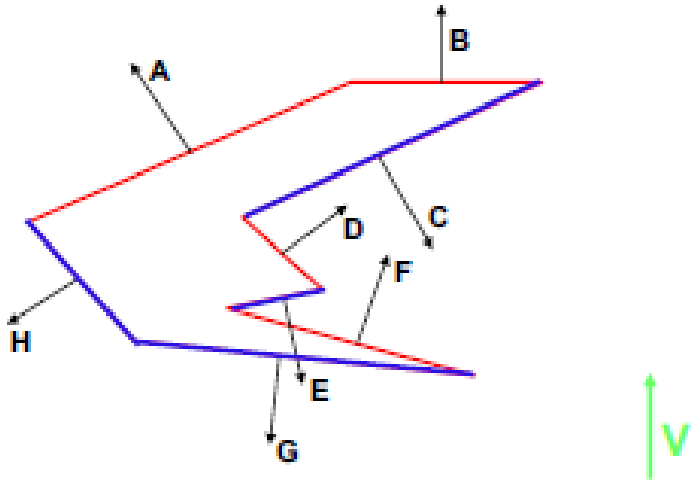
Loại bỏ mặt quay vào trong

- 3 khả năng
- *V.N>0: Mặt sau*
- *V.N<0: Mặt trước*
- *V.N=0: Song song với hướng nhìn*



Loại bỏ mặt quay vào trong

- Ví dụ

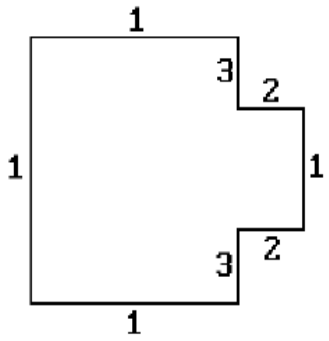


Mặt sau: A, B, D, F

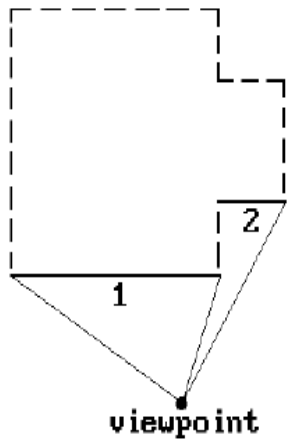
Mặt trước: C, E, G, H

Thuật toán ưu tiên theo danh sách Schumacker

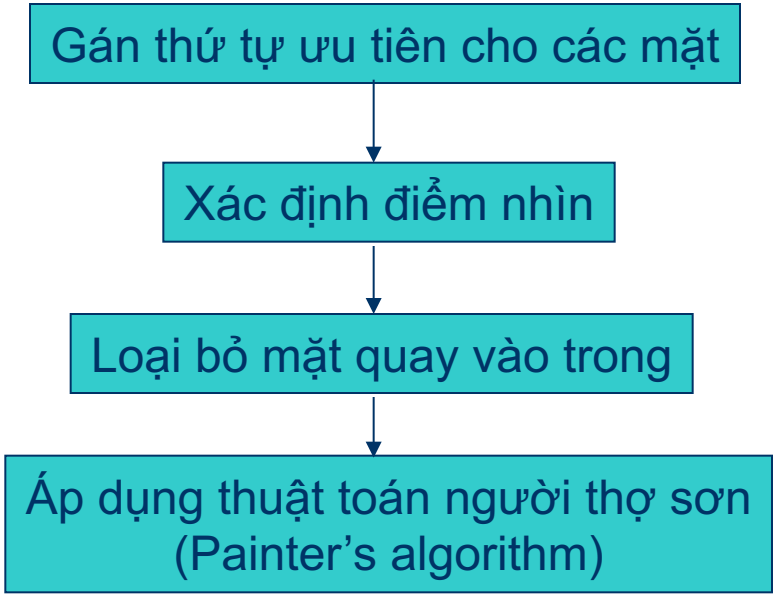
- Ý tưởng: gán thứ tự ưu tiên cho các mặt



(a)

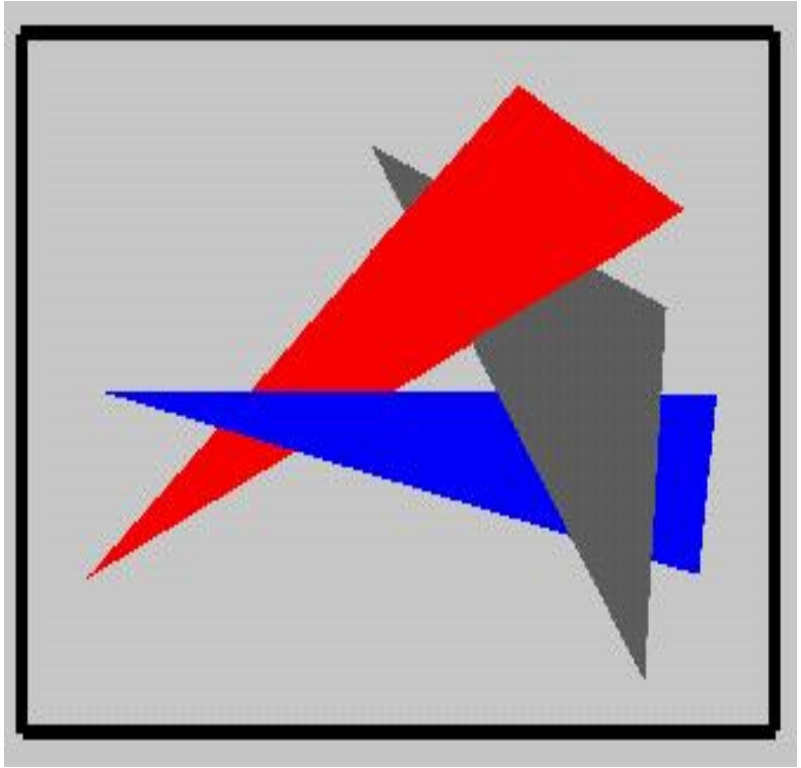


(b)

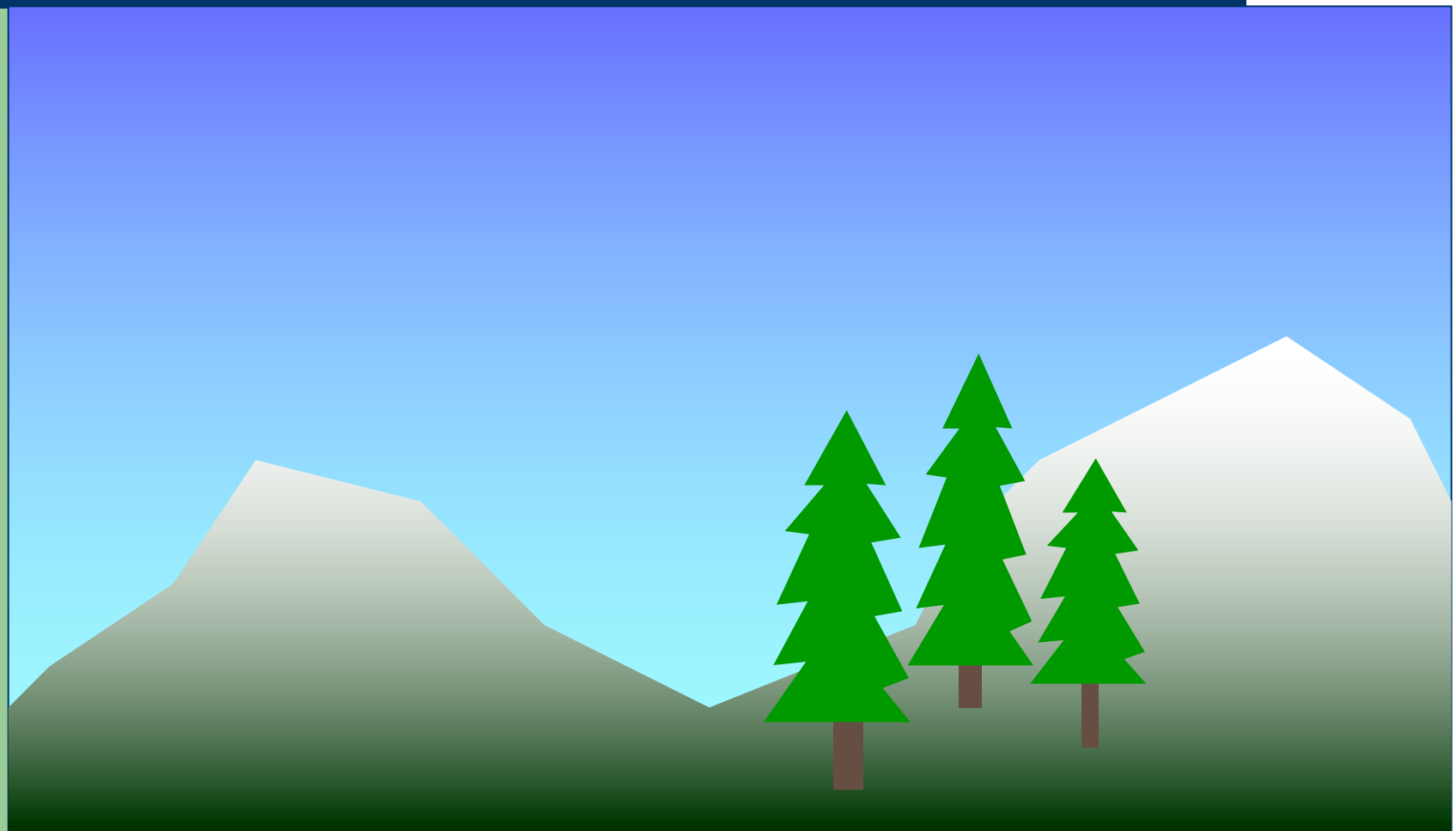


Thuật toán người thợ sơn

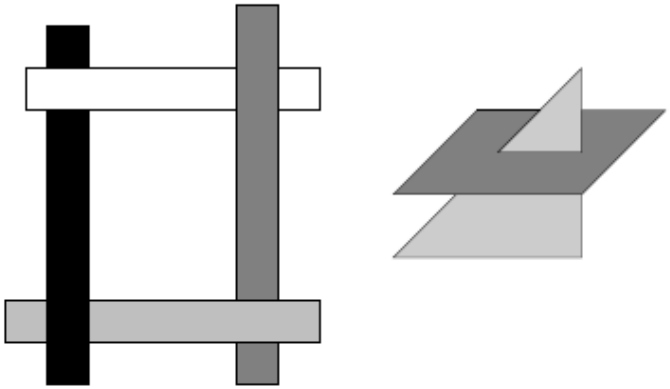
- Vẽ các bề mặt theo thứ tự từ sau đến trước – các đa giác gần hơn sẽ được vẽ đề lên đa giác xa hơn.
- Hỗ trợ tính trong suốt.
- Vấn đề mấu chốt là xác định thứ tự.
- Không phai lúc nào cũng thực hiện được.



Thuật toán người thợ sơn



Gán thứ tự ưu tiên?

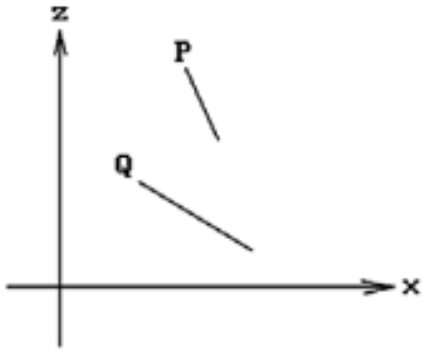


- Sắp xếp các đối tượng theo chiều sâu

→ Thuật toán Newell-Newell-Sancha

Sắp xếp theo chiều sâu Newell-Newell-Sancha

- Sắp xếp các đối tượng theo chiều sâu dựa trên giá trị z
- Xét P – đa giác xa nhất so với điểm nhìn và đa giác tiếp theo Q
- P&Q tách biệt nhau về độ sâu
  - Đúng: P không bao giờ che khuất mặt nào → vẽ P
  - Sai: Xét các tập đa giác {QS} giao P theo chiều sâu



Sắp xếp theo chiều sâu Newell-Newell-Sancha

- {QS} giao P? → Các phép thử:

1. Có thể phân tách P và {QS} theo x được không?
2. Có thể phân tách P và {QS} theo y được không?

Sắp xếp theo chiều sâu Newell-Newell-Sancha

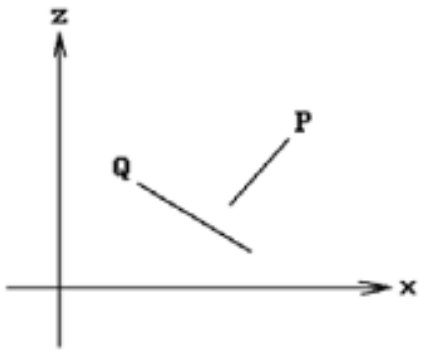
- {QS} giao P? → Các phép thử:

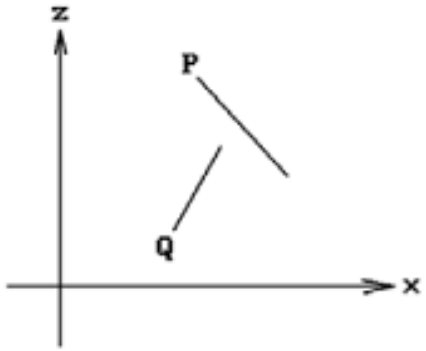
3. P có nằm ở phần xa của {QS} không?

(all vertices of P lie deeper than the plane of Q)

4. {QS} có nằm ở phần gần của P không?

(all vertices of Q lie closer to the viewpoint than the plane of P)





Sắp xếp theo chiều sâu Newell-Newell-Sancha

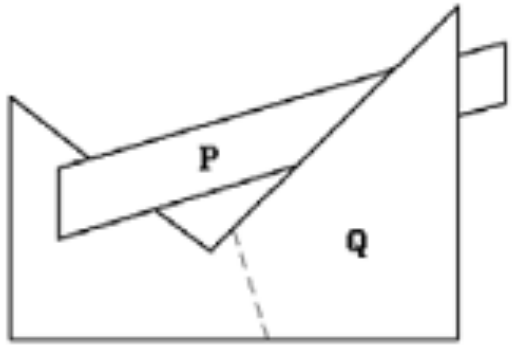
- {QS} giao P? → Các phép thử:

5. Hình chiếu của P và {QS} có rời rạc không?
   nếu tất cả các câu trả lời là không

→ Hoán đổi P với một mặt trong {QS}: lặp lại các phép thử

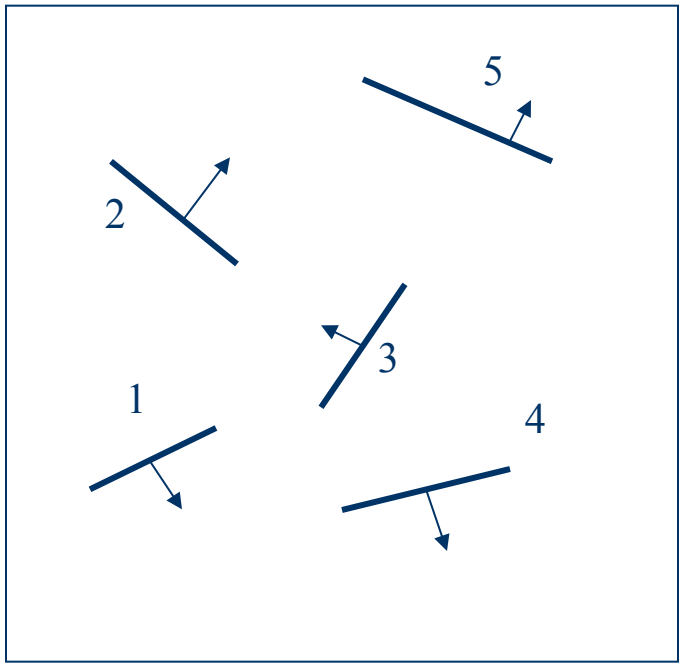
Sắp xếp theo chiều sâu Newell-Newell-Sancha

- Vòng lặp vô hạn



Cây BSP (Binary Space Partitioning)

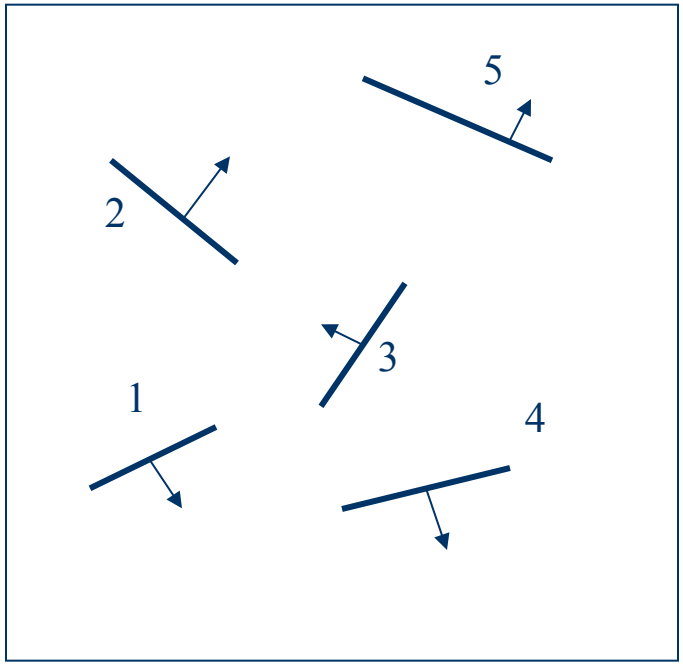
•2 bước:
-Chuyển danh sách đa giác sang dạng cấu trúc cây nhị phân (cây BSP)
-Duyệt cây BSP và vẽ các đa giác ra bộ đệm khung theo thứ tự từ sau ra trước



View of scene from above

Cây BSP

•Mặt phẳng phân tách: sao cho không có đa giác nào nằm ở nửa không gian chứa điểm nhìn bị một đa giác nằm ở nửa không gian còn lại che khuất



5 đa giác
các mũi tên chỉ về phía có điểm nhìn

Cây BSP

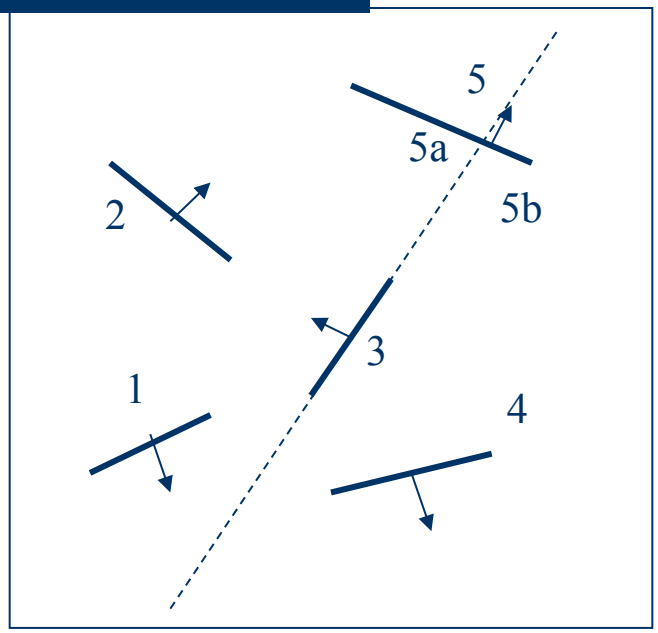
Chọn đa giác bất kỳ

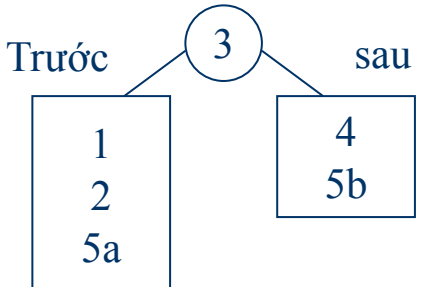
Chia cảnh vật ra 2 nửa không gian: trước và sau.

Chia những đa giác nằm ở cả hai nửa không gian.

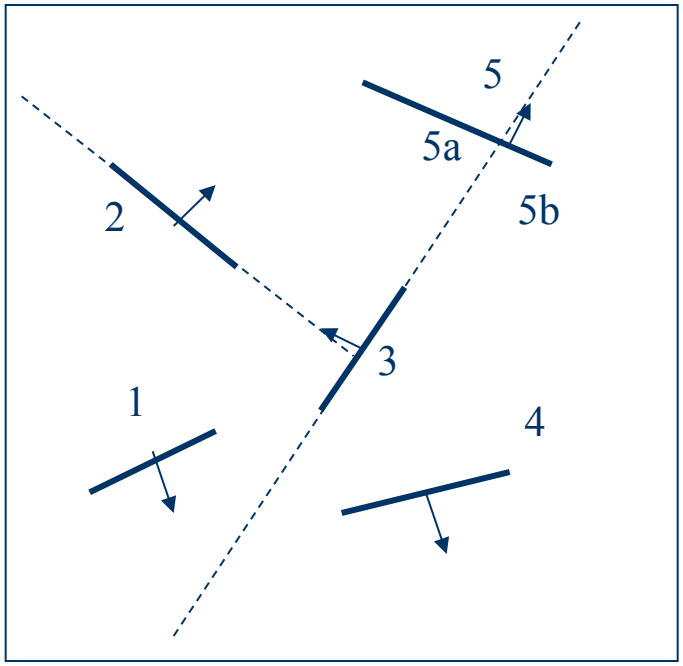
Chọn một đa giác ở mỗi nửa – chia đôi cảnh vật tiếp.

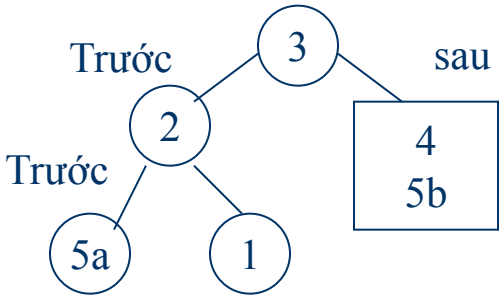
Tiếp tục chia cho đến khi mỗi phần chỉ còn một đa giác.



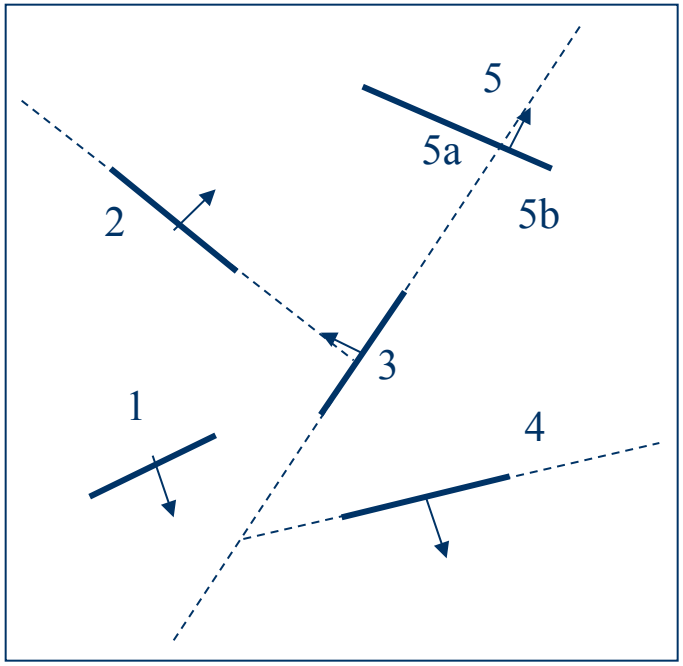


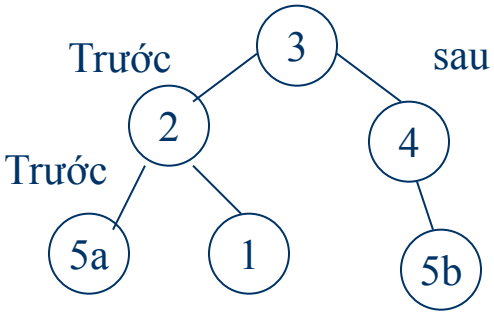
Cây BSP





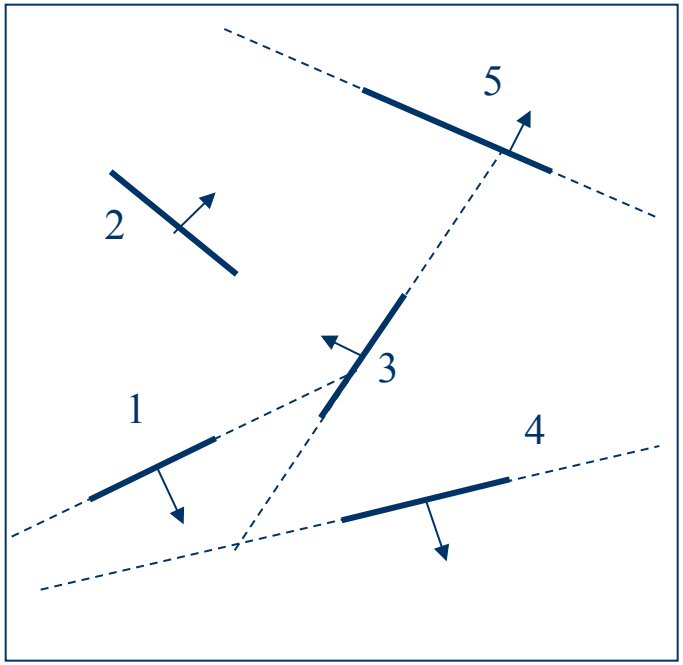
Cây BSP

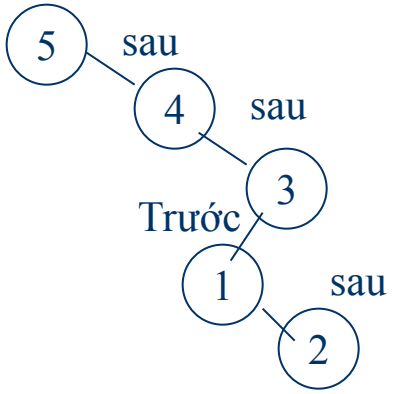




Cây BSP

Cây khác

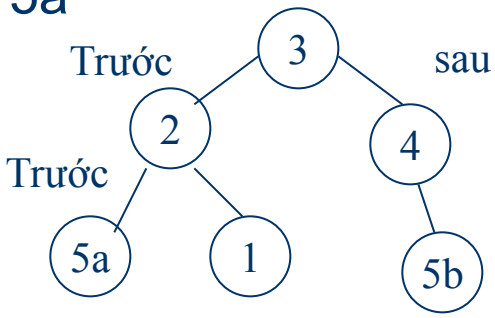




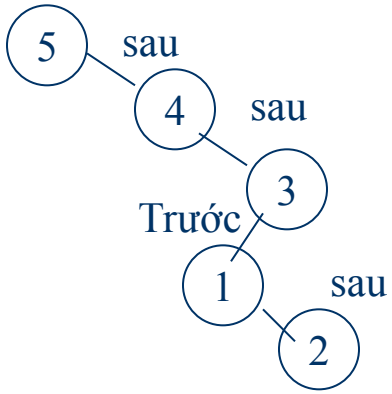
Hiện thị cây BSP

- Duyệt cây InOrder(BSP)

  5a, 2, 1, 3, 4, 5b → Thứ tự vẽ 5b, 4, 3, 1, 2, 5a



  5, 4, 1, 3, 2 →Thứ tự vẽ 2, 3,1, 4, 5



Chọn đa giác để phân chia

- Quy tắc tham lam:

Cây BSP cải tiến

- Kaplan chọn mặt phẳng tách song song với các mặt phẳng tọa độ (Thuật toán BSP trực giao – orthogonal BSP algorithm)
- Đơn giản hóa các tính toán

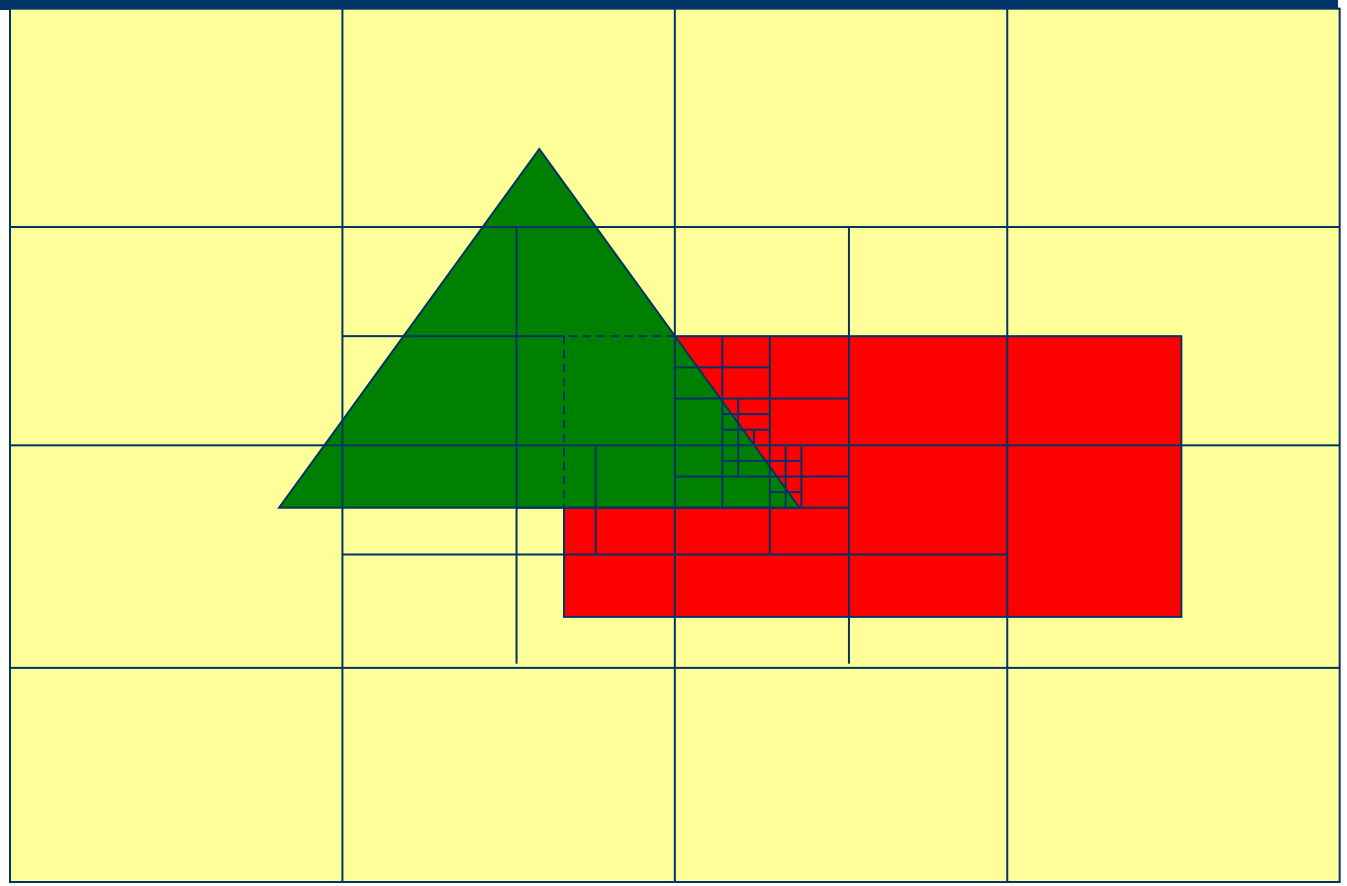
Thuật toán Warnock

- Thuật toán chính xác theo ảnh
- Tìm mảnh nhỏ HCN có cùng màu sắc/cường độ
- Thử nghiệm sau trên đa giác P bất kỳ
  1. P có tách biệt với cửa sổ không?
  2. P có bao cửa sổ không?
  3. P có giao với một phần cửa sổ không?
  4. P có nằm trong cửa sổ không?
  5. P có nằm trước các đa giác khác không?

Thuật toán Warnock

- Khởi tạo danh sách cửa sổ L (ban đầu: toàn bộ màn hình)
- Mỗi cửa sổ W trong L tìm cửa sổ thỏa mãn:
  - Tất cả đa giác tách biệt với W: vẽ W với màu nền
  - Chỉ có P giao với W: vẽ phần giao với màu của P còn lại là màu nền.
  - Tìm đa giác bao phủ W và đa giác đó nằm trước tất cả các đa giác khác giao với W: vẽ cửa sổ với màu của đa giác
- Trường hợp khác: chia cửa sổ thành 4 rồi cho vào L. Lặp lại cho đến khi kích thước cửa sổ là 1 điểm ảnh.

Ví dụ về thuật toán Warnock



Thuật toán Warnock

Thử nghiệm trên đa giác P bất kỳ

1. P có tách biệt với cửa sổ không?

Sử dụng hộp bao

Thuật toán Warnock

Thử nghiệm trên đa giác P bất kỳ

2. P có nằm trong cửa sổ không?

Thay tọa độ các đỉnh của cửa sổ và công thức cạnh đa giác

Thuật toán Warnock

Thử nghiệm trên đa giác P bất kỳ

3. P có giao với một phần cửa sổ không?

   cạnh của đa giác có giao với cạnh cửa sổ

Thuật toán Warnock

Thử nghiệm trên đa giác P bất kỳ

4. P có bao cửa sổ không?

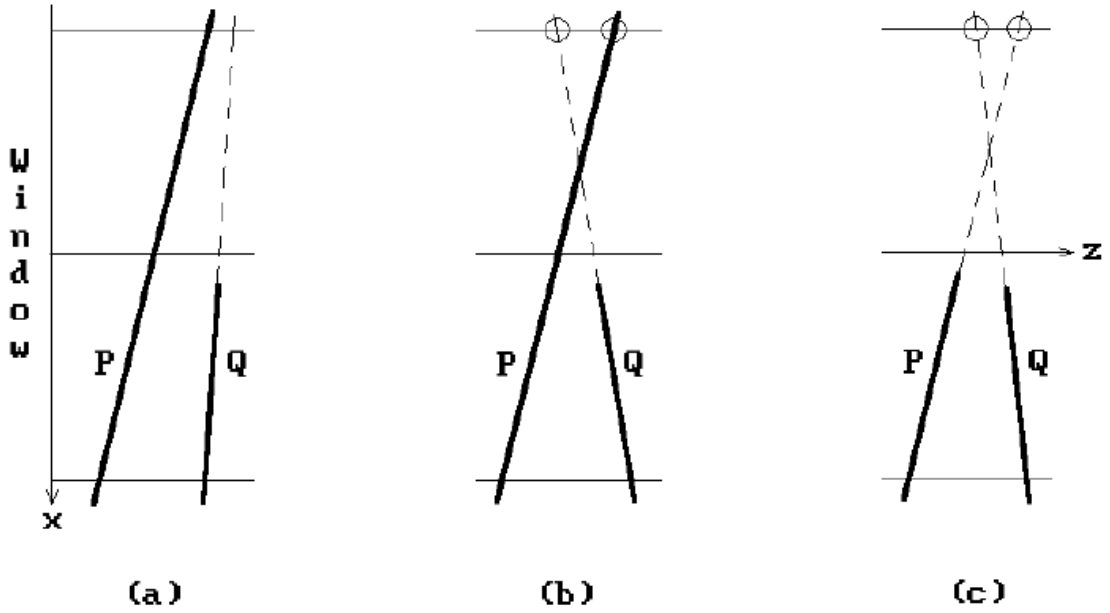
   Kẻ tia từ một đỉnh cửa sổ tính điểm giao với đa giác đang xét:

- Số điểm giao chẵn: 2 hình tách biệt nhau
- Còn lại: đa giác bao cửa sổ

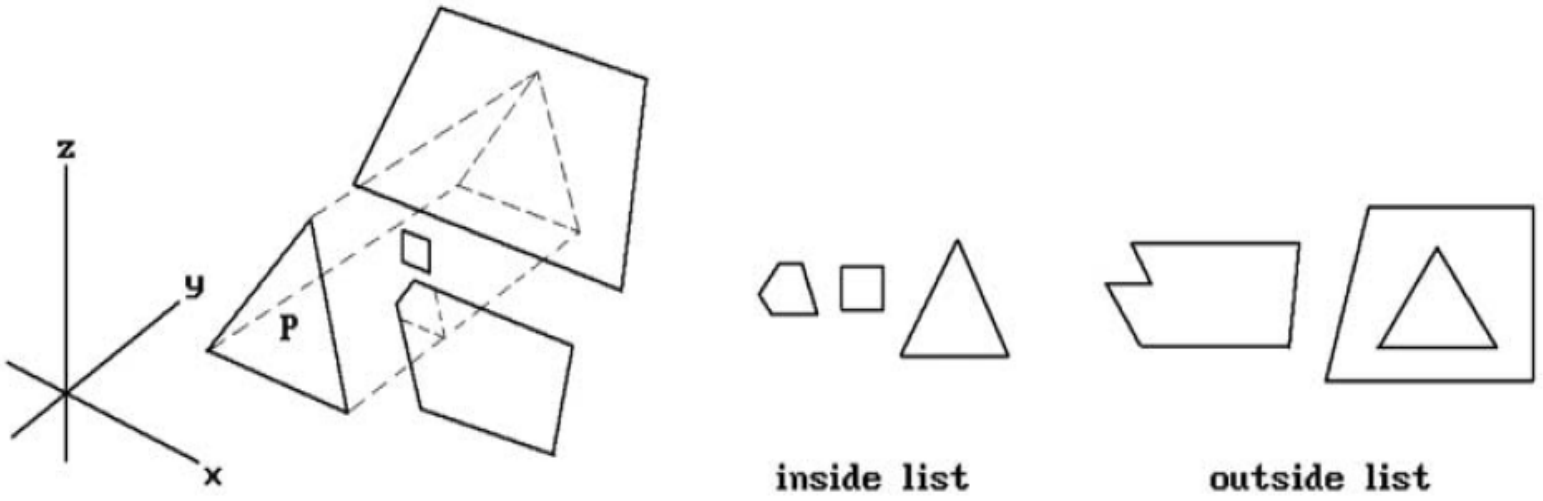
Thuật toán Warnock

Thử nghiệm trên đa giác P bất kỳ

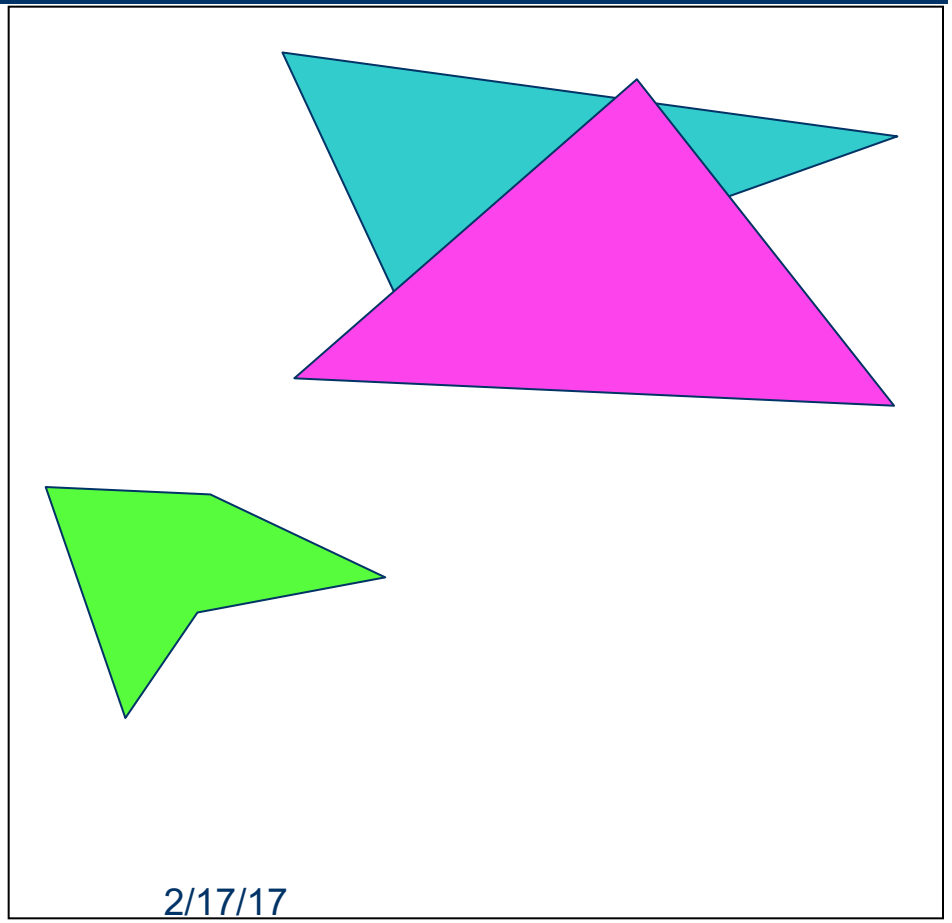
5. P có nằm trước các đa giác khác không?



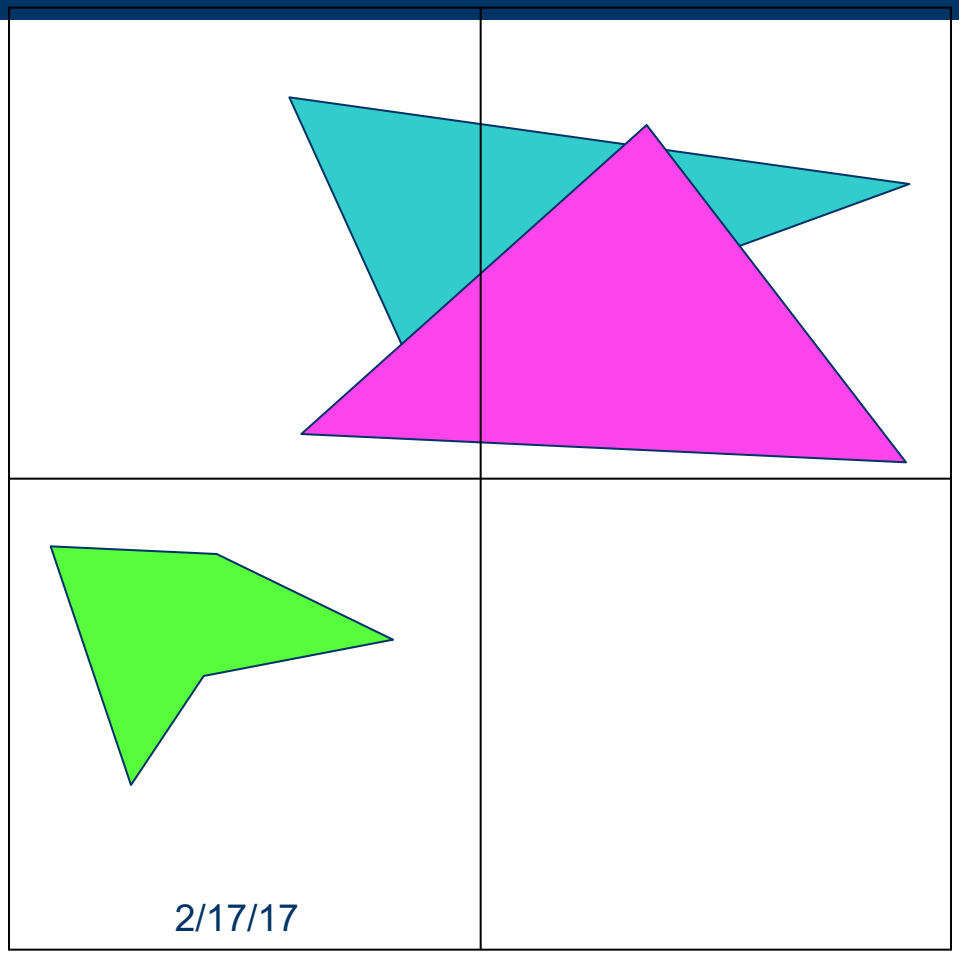
Thuật toán Weiler – Atherton



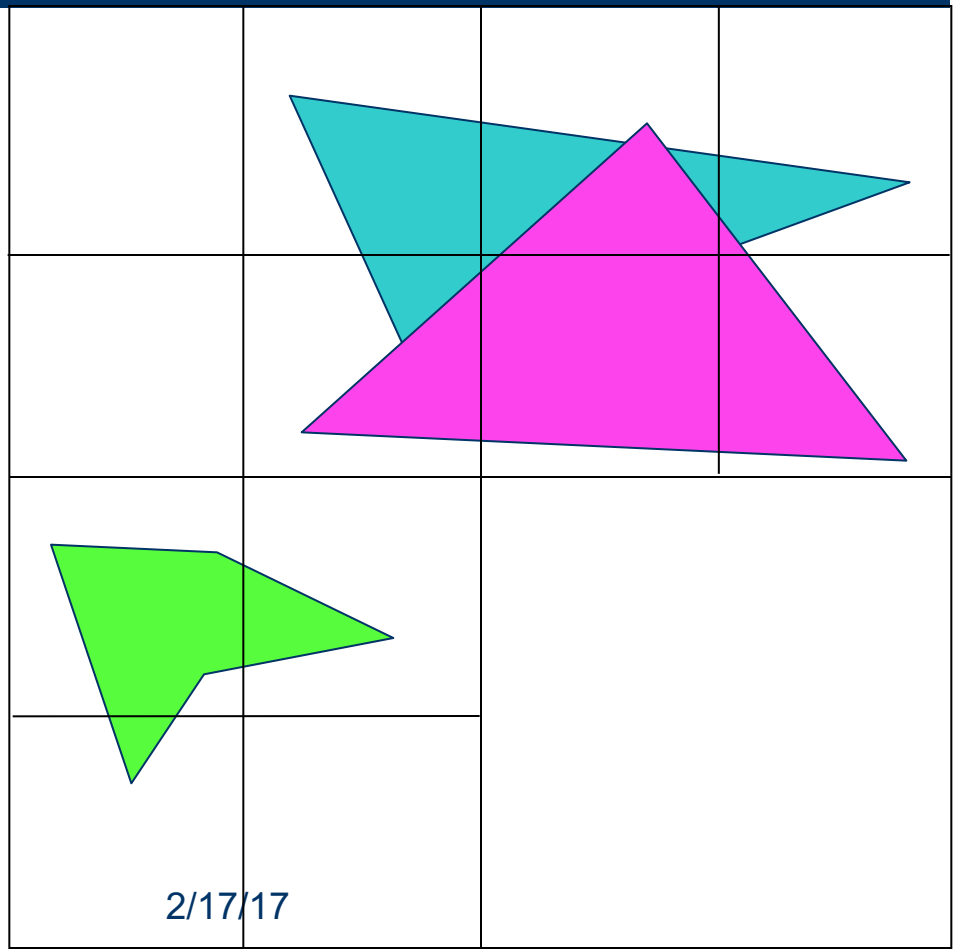
Thuật toán Warnock



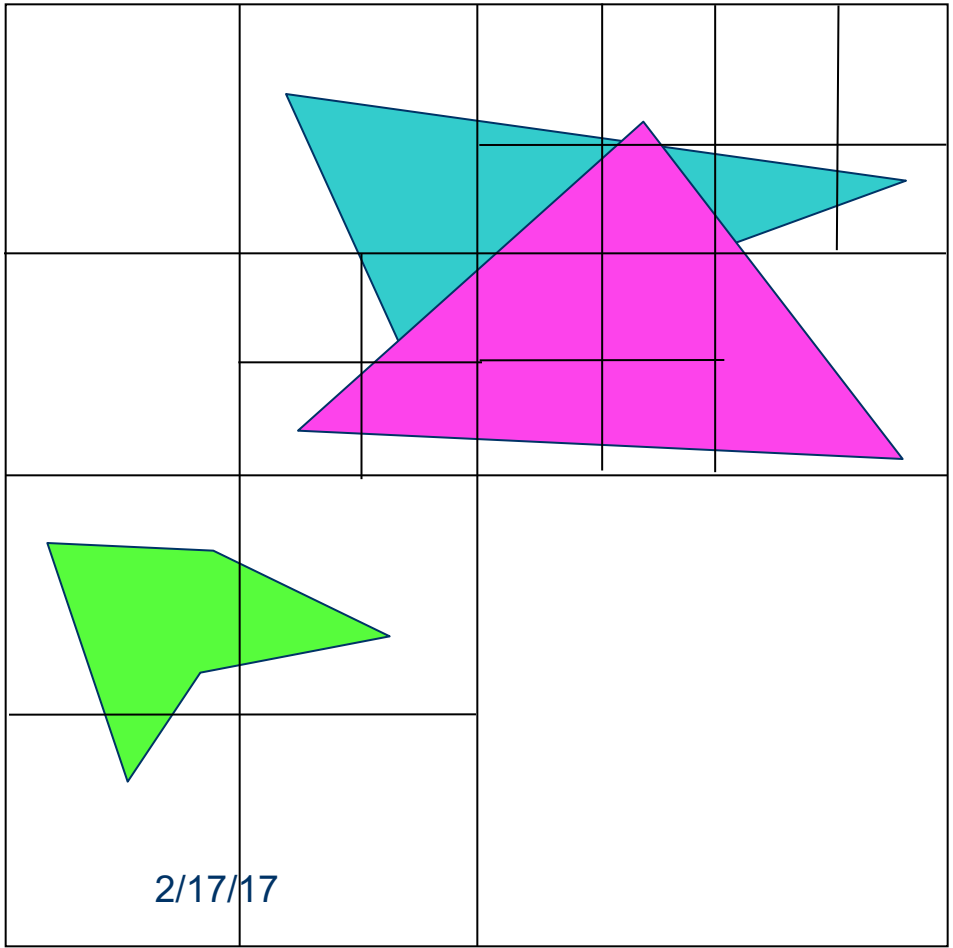
Thuật toán Warnock



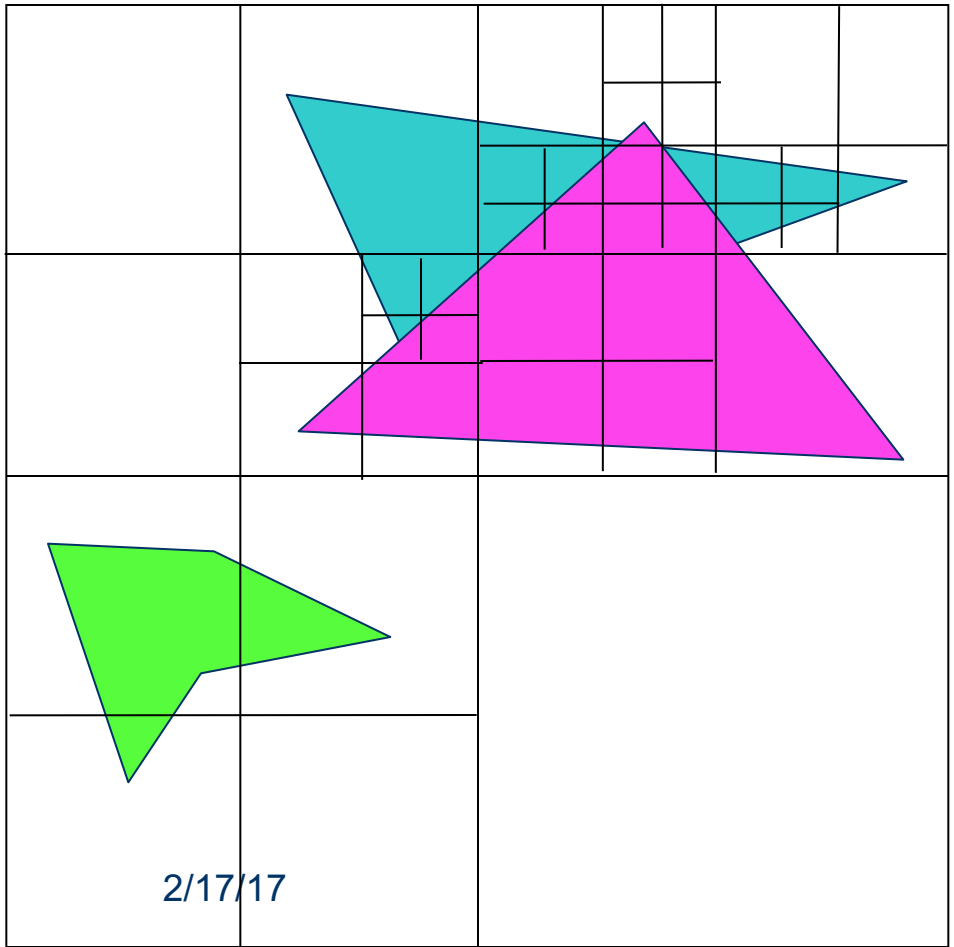
Thuật toán Warnock



Thuật toán Warnock

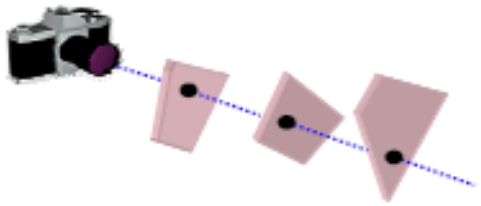


Thuật toán Warnock



Thuật toán bộ đệm Z

- Lưu lại thông tin về độ sâu hiện thời của mỗi điểm
- Nội suy z trong quá trình tính toán.
- Lưu trữ một ma trận độ sâu tương ứng với ảnh đầu ra.
- Mỗi khi xử lý một đa giác, so sánh với các giá trị z đang lưu trữ.
- Lưu lại giá trị màu của những điểm gần nhất.



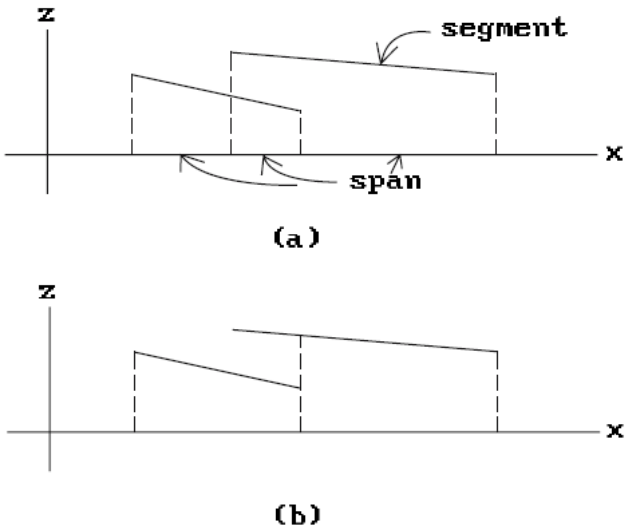
Cài đặt thuật toán bộ đệm Z

Bộ đệm Z: Mảng 2 chiều chứa các số thực
Kích thước giống bộ đệm khung

- Khởi tạo bộ đệm ảnh với màu nền.
- Khởi tạo bộ đệm Z với z = giá trị max. của mặt phẳng clipping.
- Cần tính giá trị z cho mỗi điểm
  - Bằng cách nội suy từ các đỉnh đa giác.
- Cập nhật cả bộ đệm ảnh và bộ đệm độ Z.

Kết hợp với thuật toán dòng quét

Đoạn (segment): giao của mặt với dòng quét
Nhịp (span)



Kết hợp với thuật toán dòng quét

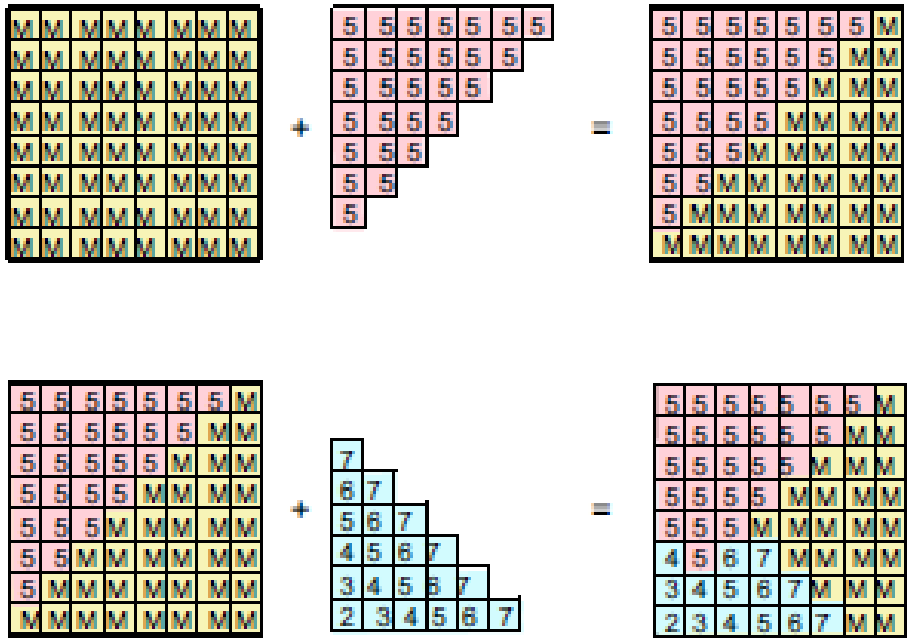
Mảng tương ứng với một dòng quét

Ví dụ: độ phân giải màn hình 1024x768

-Z buffer: 786000 bit (5.25Mb)

-Scanline Z buffer: 1024 bit (16kb)

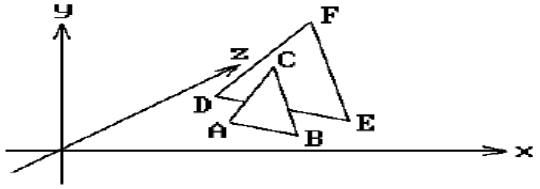
Ví dụ


M M M M M M M M
M M M M M M M M
M M M M M M M M
M M M M M M M M
M M M M M M M M
M M M M M M M M
M M M M M M M M
M M M M M M M M
+
5 5 5 5 5 5 5
5 5 5 5 5 5
5 5 5 5 5
5 5 5 5
5 5 5
5 5
5
=
5 5 5 5 5 5 5 M
5 5 5 5 5 5 M M
5 5 5 5 5 M M M
5 5 5 5 M M M M
5 5 5 M M M M M
5 5 M M M M M M
5 M M M M M M M
M M M M M M M M
5 5 5 5 5 5 5 M
5 5 5 5 5 5 M M
5 5 5 5 5 M M M
5 5 5 5 M M M M
5 5 5 M M M M M
5 5 M M M M M M
5 M M M M M M M
M M M M M M M M
+
7
6 7
5 6 7
4 5 6 7
3 4 5 6 7
2 3 4 5 6 7
=
5 5 5 5 5 5 5 M
5 5 5 5 5 5 M M
5 5 5 5 5 M M M
5 5 5 5 M M M M
5 5 5 M M M M M
4 5 6 7 M M M M
3 4 5 6 7 M M M
2 3 4 5 6 7 M M

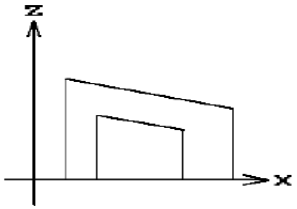
Thuật toán dòng quét Watkins

- Thuật toán dòng quét trong không gian ảnh
- Nhịp hiện tại:
  + đầu mút phải “biến đổi”
  + đầu mút trái cố định

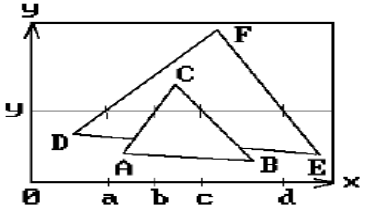
Thuật toán dòng quét Watkins



(a)



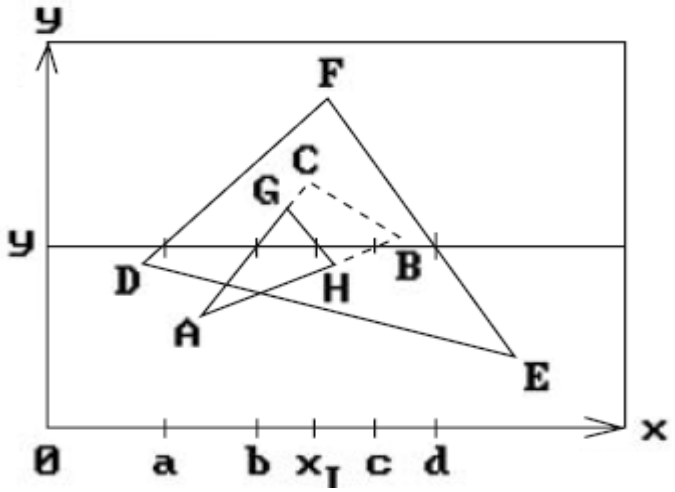
(b)



(c)

Label	polyCount	spanLeft	spanRight	ABC.active	DEF.active
LA	0	0	a	F	F
LB	1	0	a	F	T
LA	1	a	b	F	T
LB	2	a	b	T	T
LA	2	b	c	T	T
LB	1	b	c	F	T
LA	1	c	d	F	T
LB	0	c	d	F	F

Thuật toán dòng quét Watkins



Label	polyCount	spanLeft	spanRight	ABC.active	DEF.active	spanStack
LA	0	0	a	F	F	nil
LB	1	0	a	F	T	nil
LA	1	a	b	F	T	nil
LB	2	a	b	T	T	nil
LA	2	b	c	T	T	nil
LC	2	b	x_I	T	T	c
LD	2	x_I	c	T	T	nil
LB	1	x_I	c	F	T	nil
LA	1	c	d	F	T	nil
LB	0	c	d	F	F	nil

Cây BSP.

- Cần một lượng lớn tính toán khi bắt đầu
  - Chia đa giác
- Nhanh chóng để xác định tính hữu hình khi cây được tạo ra.
- Có thể được sử dụng để tính toán sự hữu hình chính xác cho bất kỳ cảnh vật nào.

$\Rightarrow$ Hiệu quả khi các vật thể không thay đổi trong cảnh vật.

Hiệu năng của thuật toán Warnock

- Chia không gian màn hình (độ phân giải màn hình, r = w*h), thuật toán lai giữa không gian vật thể và không gian ảnh, tốt với một số lượng nhỏ đối tượng, chính xác.
- Bộ nhớ làm việc: O(n)
- Bộ nhớ lưu trữ: O(n lg r)
- Thời gian để xác định tính hữu hình : O(n*r)
- Vẽ thừa: không

Hiệu năng của thuật toán BSP

- Xây dựng cây và duyệt cây (thuật toán thứ tự trong không gian vật thể – tốt với một số lượng nhỏ các đối tượng, chính xác)
- Bộ nhớ làm việc: O(1), O(lg n)
- Bộ nhớ lưu trữ: $O(n^2)$
- Thời gian để xác định tính hữu hình: $O(n^2)$
- Vẽ thừa: không

Hiệu suất của Z-buffer

- Tốt khi vẽ những cảnh phức tạp, không hoàn toàn chính xác nhưng dễ cài đặt
- Bộ nhớ làm việc: O(1)
- Bộ nhớ lưu trữ: O(1)
- Thời gian để xác định tính hữu hình: O(n)
- Vẽ thừa: tối đa

Tại sao thuật toán bộ đệm z lại thông dụng?

Lợi điểm

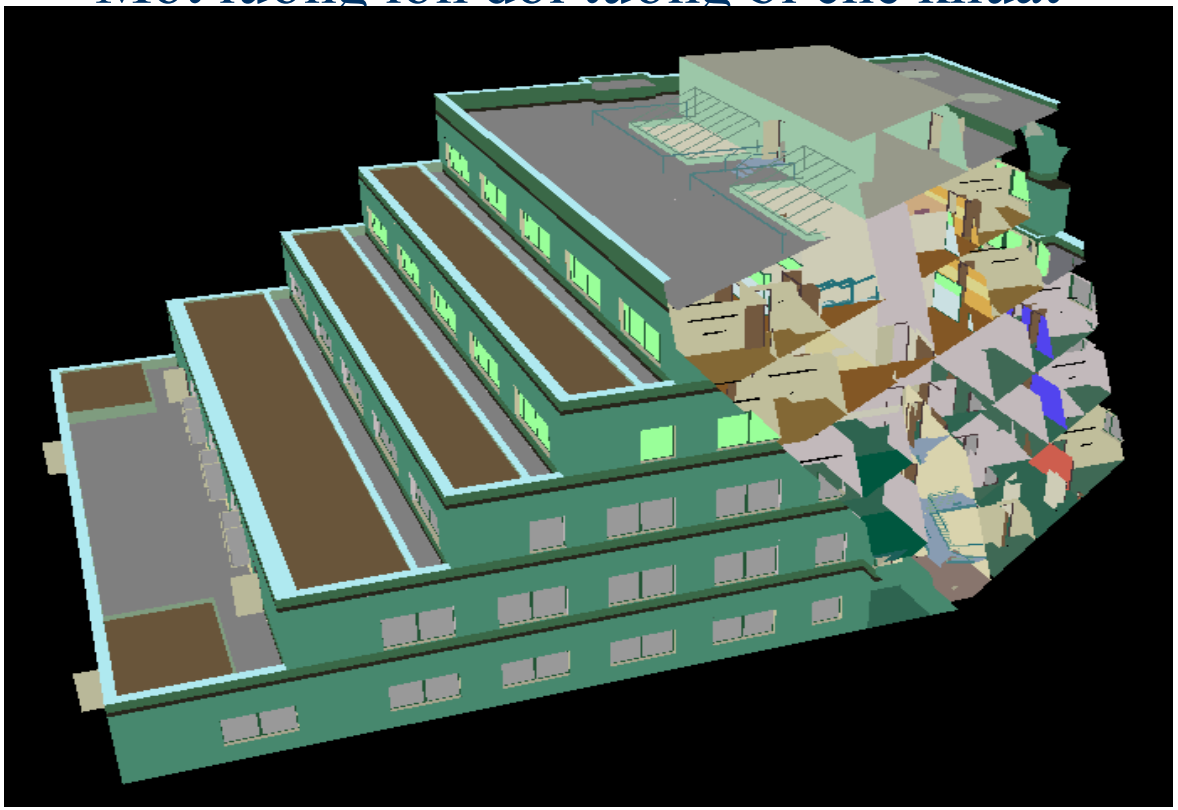
- Dễ dàng cài đặt trên phần cứng.
  - Kết hợp với thuật toán đường quét.
  - Bộ nhớ cho z-buffer không quá đắt
- Xử lý được nhiều loại đối tượng hình học – không chỉ đa giác.
- Có thể xử lý cảnh vật phức tạp đến bất cứ mức nào
- Không cần tính toán phần giao giữa các đối tượng.

Nhược điểm

- Tốn thêm bộ nhớ và băng thông
- Tốn thời gian tính toán những đối tượng ẩn

Ví dụ. Cảnh kiến trúc

Một lương lớn đối tương bi che khuất



Sự che khuất ở các mức độ khác nhau

Tổng kết

- Xác định mặt quay vào trong
- Thuật toán z-buffer
- Thuật toán BSP
- Thuật toán Warnock
- Thuật toán Watkins