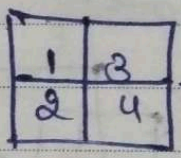
28 Monday 059-306 Wk 10

Unit - IV

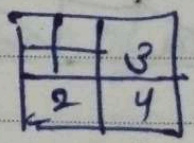
HSR → Hidden Surface Removal

* Area Subdivision Algo. :— or Area Coherence

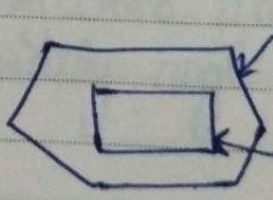
→ purposed by Warnock
→ Image space method पर depend करता है।
→ It uses divide & Conquer technique
→ This algo. will be recursive in nature because we have to divide the area into 4 parts & will process each part at a time. until we will not able to find the result.



or if we are not able to find the ans. after divide the area into 4 block then → we will divid one block into 4 parts.



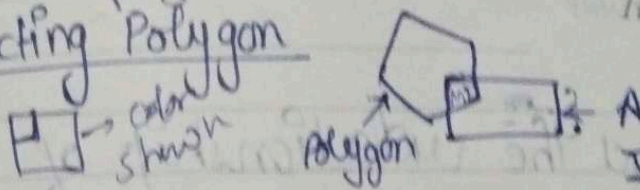
① Surrounding Polygon (only polygon will be shown)



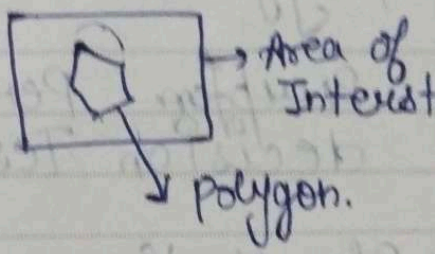
When the polygon completely covers the entire area.

The polygon intersects the area on to which it is projected.

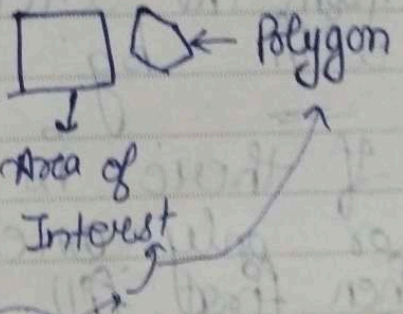
(2) Intersecting Polygon



(3) Inside or Contained
(both color is shown)



(4) Outside or Disjoint
(only bg-color is shown)



Step 1. Initilize the area to be the whole screen.

Step 2. Create the list of polygon by sorting them with their z Values. (PVPL) → Potentially visible polygon list

don't include disjoint polygons in the list because they are not visible.

March 2011

02 Wednesday
061-304
Wk 10

Step 3: Find the relationship of each polygon. (color करेंगे)

Step 4. Perform the visibility decision Test.

(a) If all the polygons are disjoint from the area, they fill area with bg color.

(b) If there is only one intersecting or only one contained polygon then first fill entire area with bg color & then fill the part of polygon contained in area with the color of polygon.

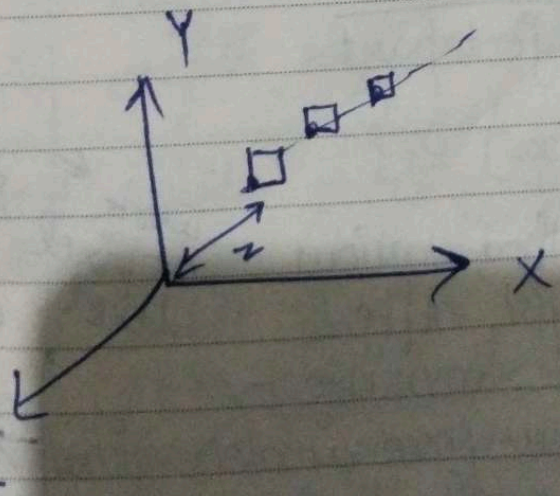
(c) If there is a single surrounding polygon but no intersecting or contained polygons then fill the area with the color of the surrounding polygon.

(d) If surrounding polygon is (more than one polygon)

closer to the view pt. then all other polygons, so that all other " are hidden by it, then fill the area with color of the surrounding polygon.
(use to terminate the algo.)

(c) If the area is pixel (x, y) & neither ~~a, b, c~~ a, b, c nor d applies. Then compute z coordinates at pixel (x, y) of all polygons in the list. Then pixel is then set to color of the polygon which is closest to the view pt.
Test are True.

Step 5. If non of the above ^ then subdivide the area & go to step 2,



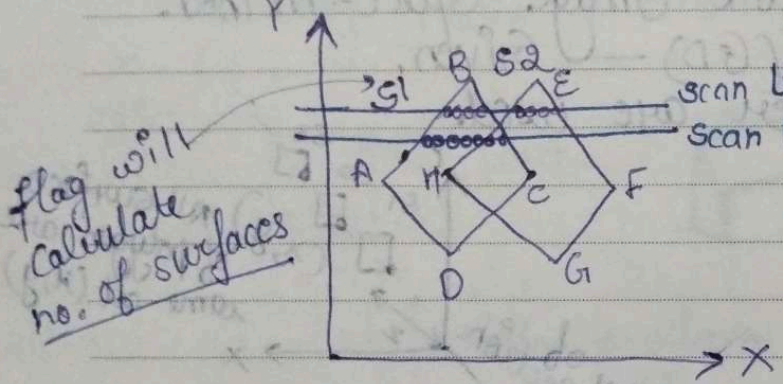
March 2011

07 Monday 066-299 Wk 11

→ also in first chapter → only fill a single polygon ⇦ scan line polygon filling algo.

✱ Scan Line Method :—

extension → a set of polygon.



Refresh buffer :- color given in this to fill the polygon.
or
frame buffer

जिन-2 को cut कर रहा है।

edge list

① Scan line 1 :- AB BC EH EF

ⓐ AB to BC flag = 1	color set होगा acc. to S1.
ⓑ BC to EH flag = 0	(because no surface area) no color or bg color

March 2011

Tuesday 08

⑥ EH to EF
flag = 1 (because one area encountered)
(color set होगा acc. to s2)

② Scan Line 2 :- Acc. to scan line there are 2 intersection points.

Active edges :- AB BC EH EF

AB → EH → flag = 1 s1 (color)
EH → BC → flag = 2 (~~s2 color~~) (z-value)
BC → EF

The min. the z-value the closer the obj. or max. the z-value the ~~cols~~ far the obj.

EH → BC → z-value of s1 < s2 z-value

BC → EF → flag = 1 (s2-color)

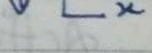
March 2011

09 Wednesday
068-297
Wk 11

✳ Algo.:-

Step 1. Initilize the each screen pixel. to a bg-color.

Step 2. Set y to the $y = y_{min}$ value in the edge list. (bottom to top)



Step 3. Repeat step 4 & 5 until no further processing can be performed.

Step 4. y-scan Loop active edges whose y min is equal to y are consider in order of increasing value of x

Step 5. x-scan loop process from left to right each active edge as follows.

(i) Invert the in or out flag of the polygon list, which contains the edge count & count the no. of active polygons such that if flag is 1 only one polygon is visible.

March 2011

Thursday 10

All the pixel values from this edge & upto the next edge are set to the color of the polygon. If flag is greater than 1 find out the visible polygon which is having smallest z-value.

The pixels from this edge & upto to next edge are set to the color of the polygon having smallest z-value.

If flag is 0, pixels from this edge & upto the next one are left unchanged.

(ii) When the last active edge is processed then procced as follows

(a) Remove those edges for which the value of y-max is equal to the present scan line value y.

If no edges remained the algo. is finished.

(b) For remaining active edge replace x by $x + \frac{1}{m}$. This is the edge

March 2011

11 Friday 070-295 Wk 11

intersecting with the next scan-line $x+1$

(c) Increment $y = y+1$ which is the next scan line & repeat step 4.

---

① y- min से x तक जाते हैं।

② then update करते रहते हैं।

③ हर scan line के acc. process करना।
acc. to flag.
flag = 0 bg-colg
flag = 1 surface का color.
flag = 2 carpet value (z min set color value of that.)

→ Eg :

Algo.

March 2011

Saturday 12
071-294
Wk 11

Intro: → list priority algo.

(*) Painter's Algo. / Priority Algo. / Depth Sort Algo. :—

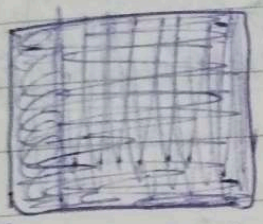
→ object-space & image space method.

→ also known as Priority or depth sort Algo.

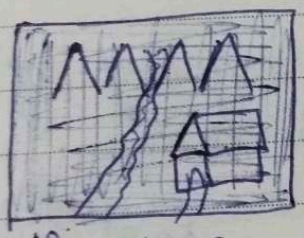
→ Technique is based on oil painting

→ Newell & Sancha (developed by)

→ Eg :-



far away.



then draw near by objects.

→ It follows back to font method.

→ depth less - nearer to viewer
depth - high - farther
high to low

Sunday 13

Algo. :- (Book)

Step (1) Sort all polygons acc. to z-coordinate

Step (2) Find ambiguities of any. find whether z coordinate overlap. split polygon if neccessary.

Step (9) Sort Answer each polygon in increasing order of z-coordinate.

March 2011

14 Monday

When two polygons below overlapped

→ Consider two surfaces S & S'
then S draw S → drawing first
S' → after that]

Tests :- Say S & S' are 2 surfaces
the following test should be
apply for checking overlapping

Test case ① No overlap of bounding rectangles (surrounding the polygon) of S & S'.

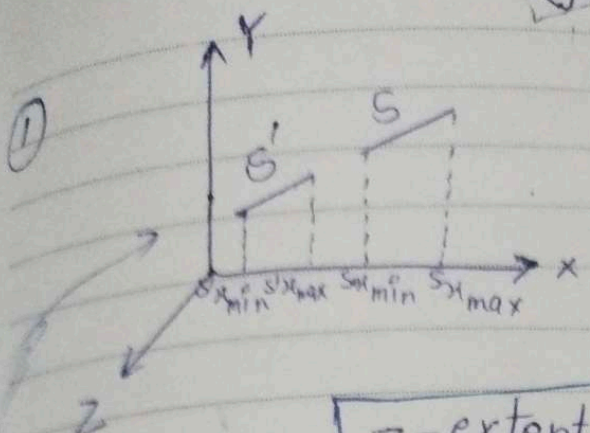
Test case ② S is lying completely behind the plane of S'

Test case ③ S' is lying completely in front of plane of S.

Test case ④ projections of S & S' don't overlap

If any of the above test fail then we have to swap S to S' & apply all the test again for S'

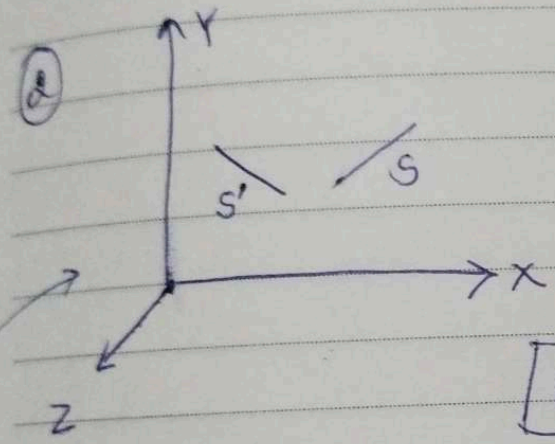
① ② ③ ④

① 

They should not be overlap

- z-extent
- y- "
- x- " $x_{max}(S) < x_{min}(S)$

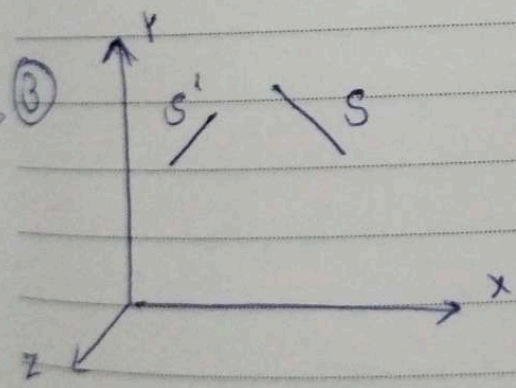
z-extent $z_{max}(S') < z_{min}(S)$

② 

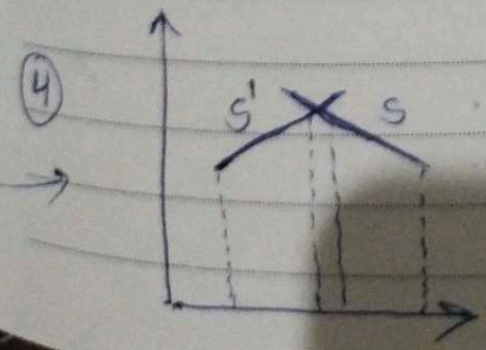
$Ax + By + C_z + D = 0$

take vertices of s & put them eq.. of S'

$Ax + By + C_z + D < 0$

③ 

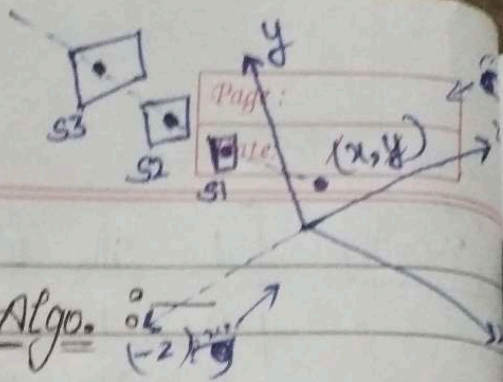
$Ax + By + C_z + D > 0$

④ 

projections should not overlap

Imp°

Unit-4



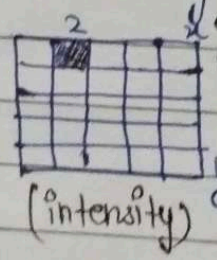
✱ Depth Buffer or z-Buffer Algo.

(i) It is used for hidden Surface Identification Removal.
(ii) Purposed by CATNULL
(iii) Based upon Image space-method.
(iv) It uses 2-buffer
(v) It compress surface Depth at each Pixel position.
(vi) The basic idea is to test z-depth of each surface to determine the closest (visible) surface.

2-Buffer

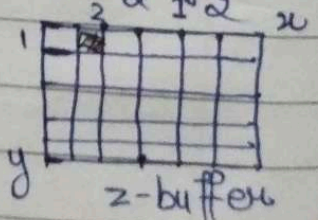
- **Frame Buffer (Refresh Buffer)** used to store intensity / color of each position (x,y)

Eg:- x y z
2 I 2



- **Depth Buffer** used to store depth of (x,y) distance / z-value from z-axis

Eg:- x y z
2 I 2



Case I

→ Sitting at +z, looking towards origin (-z)
→ Larger z value, closer to the surface.
→ Smaller z value far away the obj.

Case II

→ Sitting at (-z) & looking towards origin (+z)
→ Smaller z value → closer
→ Larger z value → farther.

Page :
Date : / /

Algo. :-

Step1. Initalize frame Buffer to Background color.

step2. Initalize z-Buffer to the depth of back clipping plane.

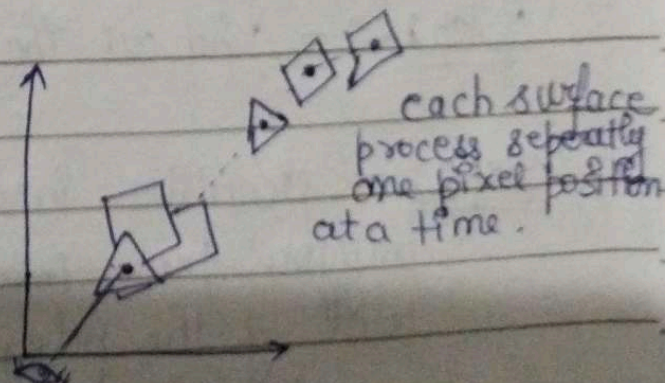
step3. Scan Convert each polygon in arbitrary order.

step4. for each pixel (x,y) calculate depth z at that pixel.

step5. Compare the new depth $[z(x,y)]$ value with that store in z-buffer $[z_{buf}(x,y)]$

step6. if $[z(x,y) < z_{buf}(x,y)]$ is less than $z_{buf}(x,y)$ then

update $z_{buf}(x,y) = z(x,y)$ &
frame buffer = color of pixel (x,y) in polygon.



Name, Address & E-mail Phones Mobile/Fax

Adv.:— ① It is easy to implement.
② It reduce speed problem.
③ It process one object at a time.
④ It is simple to use.
⑤ It displays complex surface intersections easily.
⑥ No depth sorting of objects is needed

Disadv.:— ① It requires large storage.
② It is a time-consuming process.
③ Space involved is very large.
④ It requires 2-buffers.