Name: Arpan Adhikari

Roll No. 231309

Computer Graphics

Assignment - 5

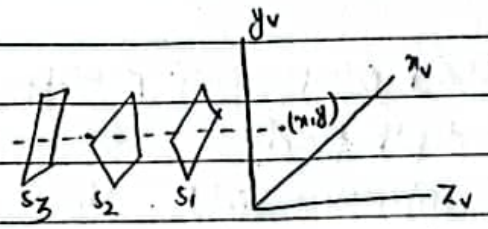
1. How can you detect hidden surface using z buffer approach? What is the limitation of z-buffer approach?

Date
Page

Depth Buffer Method / Z-Buffer (ISM)

→ Tests the z-depth of each surface to determine the closest (visible) surface

→ To override the closer surface from far ones, two buffer are used.



Frame buffer, Depth Buffer

⇓ store intensity value of color

⇓ store depth values for (x,y) position $(0 \leq \text{depth} \leq 1)$

Algorithm :

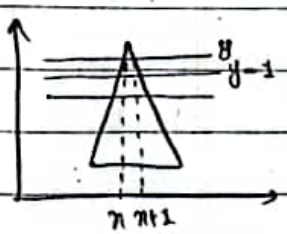
1. Set the buffer values. depth buffer (x,y) = z_{min}
   frame buffer (x,y) = background color
2. Calculate the depth 'z' for each (x,y) position on the surface
3. If z > depth buffer (x,y), compute surface color.
   Set depth buffer (x,y) = z
   frame buffer (x,y) = surface color (x,y)

After all process, depth buffer contain depth value for visible surface & frame buffer (refresh buffer) contains corresponding intensity value.

succeeding depth value can be calculated by using simple mathematics.

Eqⁿ of surface : $Ax + By + Cz + D = 0$

value of depth at (x,y) ; $z = \frac{-Ax - By - D}{C}$



For $(x+1)$, $z' = \frac{-A(x+1) - By - D}{C}$

$= \frac{-Ax - By - D}{C} - \frac{A}{C}$ so, $z' = z - \frac{A}{C}$ → constant for a surface.

A-Buffer (ISM)

→ Antialiased, area-averaged, accumulation-buffer method

→ Eliminate drawback of depth buffer (could only be used for opaque objects)

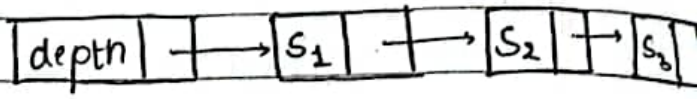
→ Each position in A-buffer has two fields.

1. depth field : stores a positive or negative real number
2. Intensity field : stores surface-intensity information or pointer valu

depth	intensity

depth value -ve:
multiple surface contributions to pixel intensity

→ maintains linked list



(Three surface overlap)
→ Transparent

depth value +ve:
single surface

depth	intensity

Intensity field:

① RGB components
② % of pixel coverage

Intensity field:

① RGB component
② % of pixel coverage
③ % of transparency
④ depth
⑤ surface identifier
⑥ other surface rendering parameters
⑦ pointer to next surface

Back Face Detection (OSM)

- Uses "Inside-Outside test"

1) A point $P(x,y,z)$ is inside a polygon surface with plane parameters A,B,C,D if
$Ax+By+Cz+D<0$.

2) Polygon is back face if $V.N > 0$

Angle bet^n V & N is 0.

3) How does the OSM approach work differently than the ISM Approaches for detecting visible surfaces in 3D?

Ans: The differences are.

Features	OSM (Object Space Model)	ISM (Image Space Model)
Defination	Operate directly on 3D objects before rasterization.	operates on the 2D image (pixels) after the 3D scene is projected onto the 2D scree
Stage of Operation	Works in object space	Works in image space.
Visibility Determination	Determines visibility by comparing objects or polygon in 3D space.	Determines visibility by Comparing depth values or attributes at each pixel
Precision	High precision as calculations are performed in 3D space.	Low precision as calculations are performed on discrete pixels.
Complexity	Computationally expensive for complex scene with many overlapping objects.	More efficient for complex scenes as calculations are limited.
Transparency Handling	Difficult to handle transparency accurately.	Can handle transparency effectively.
Anti Aliasing	Doesn't support anti-aliasing	supports.
Memory usage)	Low memory usage.	Higher memory usage.
Example	- Back face detection - Depth sorting	- A-Buffer - z-buffer

1) Differentiate between Depth Buffer & Depth Sorting Approach for detecting visible surfaces in 3D?

Ans :- the differences are :-

Features	Depth Buffer (z-buffer)	Depth sorting (Painter's Alg)
Basic Principle	compares depth values of each pixels to determine visibility	sorts polygons by depth & draw them from back to front.
Type of Algorithm	ISM	OSM
Visibility Determination	Determines visibility at the pixel level using a depth buffer.	Determines visibility by sorting polygons in 3D space & drawing them in order.
Memory usage	Requires a depth buffer to store depth values for each pixel.	Doesn't require additional buffers, but requires memory for sorting polygon
Handling Transparency	Cant handle transparency correct.	can handle transparency by drawing polygon in the correct order.
Complexity	Simple to implement & efficient for complex scenes.	More complex to implement, especially for scene with intersecting polygon
Limitations	- Requires additional memory for the depth buffer - Cant handle transparency.	- Cant handle intersecting polygon correctly - overdraw can reduce performance.

Depth-sorting Method (Painter's Algorithm) (Hybrid of OSM & ISM)

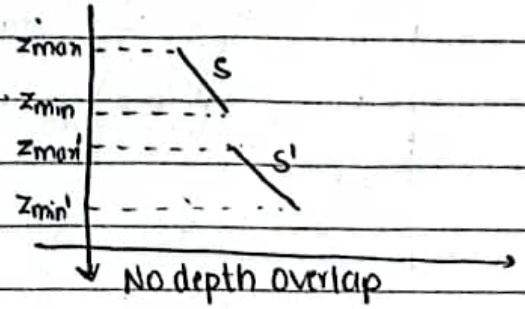
⇓

Because distant objects are made first then nearby objects

Makes use of both OSM & ISM to

(i) sort surface in decreasing depth order (using OSM & ISM)

(ii) scan converted in order with surface with greatest depth (using ISM)



→ Sort surface in descending order

→ If $z_{min} > z_{max'}$

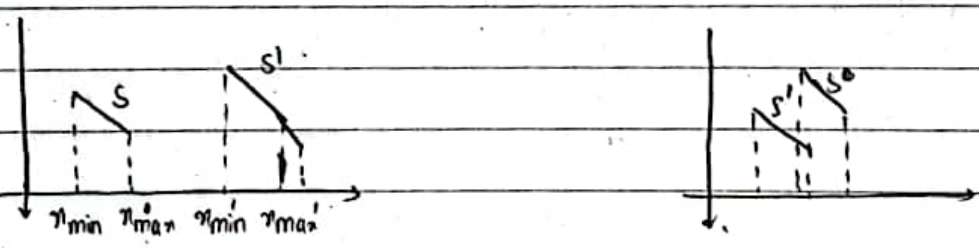
we paint S first, then S'.

If not this, Additional Test required.

If any of test true → NO reording needed

All Fail → Reordering needed.

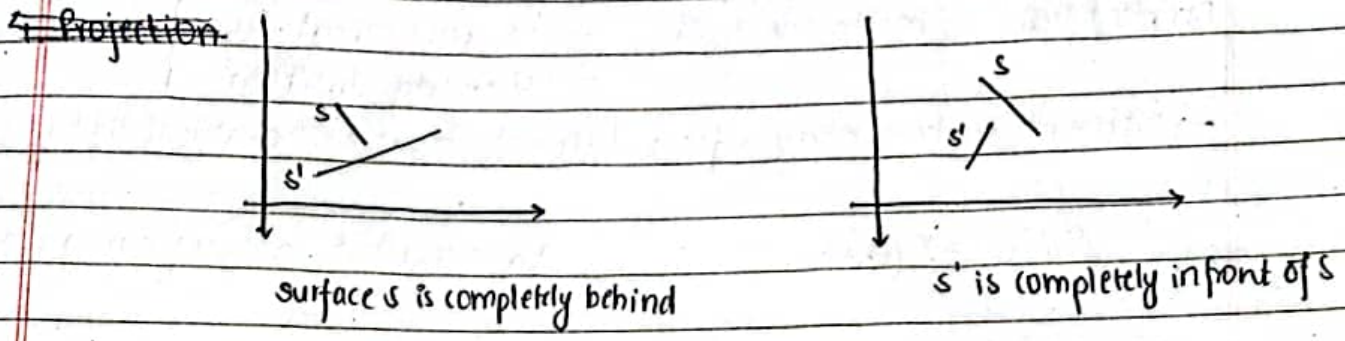
1. Bounding rectangle in XY plane for two surface don't overlap.



2. Surface S is completely behind the overlapping surface relative to the viewing position.

3. Overlapping surface S' is completely infront of S relative to viewing position.

Test ② & ③ are performed by "inside-outside test"
→ Substitute coordinates of each vertices to plane eqⁿ to determine sign or normal vector method.



surface s is completely behind

s' is completely infront of s

④ Projection of two surfaces onto view plane don't overlap.

For test ①, we can use line eqⁿ in xy plane for checking intersection, between bounding edge of two surface.

5) Explain the working mechanism of Back Face Detecting algorithm.

Ans The working mechanism is as:-

1) Calculation of the surface normal ($\vec{N}$)
2) Determining the viewing Direction ($\vec{V}$)
3) Compute the Dot product
   Dot product = $\vec{N} \cdot \vec{V}$
4) Determine visibility.
   - If dot product > 0, polygon is facing away & is not visible.
   - If negative than is facing towards viewer & is visible.

6) Why is it required to remove hidden surfaces in 3D viewing?

Ans. Hidden surface removal is essential in 3D viewing for several key reasons:

1) Visual Realism:
Ensures objects are displayed correctly with nearer with nearer surface obscuring farther ones, matching ones, matching real world depth perception.

2) Computational Efficiency :- Prevents wasting resources rendering pixels that wont be visible in final image.

3) Clarity :- Eliminate visual confusion by removing surfaces that should be blocked from view.

4) Rendering Accuracy :- Ensures proper handling of intersecting objects & complex 3D scenes.

Q.7) How does hidden surface removal algorithm bring visual realism in graphical scenes?

Ans Hidden surface removal algorithm ehnance visual realism by:

1) Depth Management
- Accuracy represents ~~patt~~ spatial relationships.
- Ensures closer objects properly occlude distant ones.

2) Light & shadow
Enables correct shadow casting
- Supports proper lightining calculations on visible surface.

3) Material Interaction
- Allows accurate rendering of reflections
- Handles ~~hand~~ transparency & translucency correctly

4) Scene Complexity
- Manages overlapping objects accurately
- Resolves intersections between 3D objects realistically.

Q.8) What are list priority algorithms? Explain the working mechanism of Depth sorting approach.

Ans Priority algorithm order surfaces /objects by some priority measure to determine visibility.

Depth sorting (Painter's Algorithm)

1) Calculate depth of each polygon's centroid
2) Sort all polygon by depth, farthest to nearest.
3) Render polygons in sorted order, with nearer polygon's overwriting farther ones.

10) How is scanline approach different from z-buffer approach for hidden surface removal?

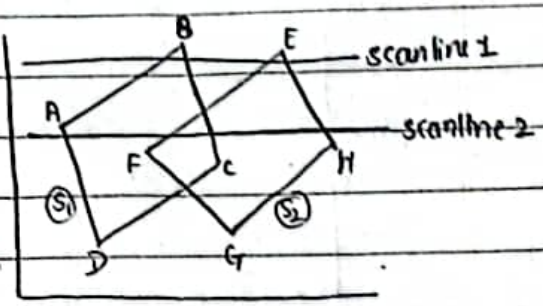
Aspect	Scan line Approach	z-Buffer Approach
Processing unit.	works line by line	works pixel by pixel.
Memory usage	Requires storage only for active scan line.	Requires full-screen depth buffer.
Sorting	Maintains active edge list & sorts by x-cordinate	No sorting required
Speed	Slower due to sorting & list maintainance.	Faster, simple depth comparision.
Accuracy	Handles polycon precisely at scan line intersection	subject to depth buffer precision.
Memory Requirement.	Lower memory usage	Higher memory usage.

Scan Line Method (ISM)

Three tables are maintained.

- **Edge Table (ET)** → contains information about all edges
- **Active Edge Table (AET)** → stores edge that intersect current scan line
- **Polygon Table (PT)** → contains info about each polygon

- Between AB & BC, flag for surface S_1 is ON. Between EF & EH, flag for surface S_2 is ON.
- NO depth calculation needed because one flag only ON at one time.



- Between EF & BC in scanline 2, both flag S_1 & S_2 ON. So, depth calculation needed.

Algorithm:

1. Set up polygon table for polygon surfaces.