# UNIT-4



## Learning Objective

### Shading

- Shading, Illumination Model for diffused Reflection
- Effect of ambient lighting & distances
- Specular Reflection Model
- Computing Reflection Vector, Curved Surfaces
- Polygonal Approximations
- Gourard Shading, Phong Model.

### Hidden Surface Removal

- Floating Horizon Method
- Depth Buffer (Z-Buffer, A-Buffer) Method
- Scan Line Method, Depth Sorting Method
- BSP- tree Method
- Area Subdivision Method

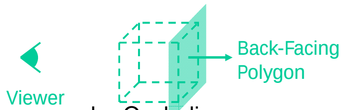


## Hidden Surface Removal

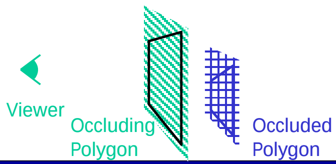
- Motivation
- Introduction
- Algorithms for HSR
  - Back-face detection
  - Scan Line Method
  - Floating Horizon Method
  - Depth Buffer (Z-Buffer, A-Buffer) Method
  - Painter's algorithm or Depth Sorting Method
  - BSP- tree Method
  - Ray casting
  - Area subdivision
- Tradeoffs

## Motivation

- Surfaces may be Back-Facing:



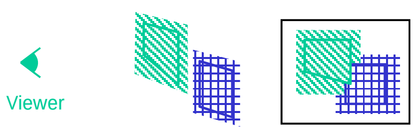

- Surfaces may be Occluding:



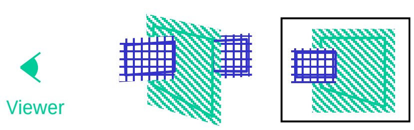



## Motivation

- Surfaces may be Overlapping:

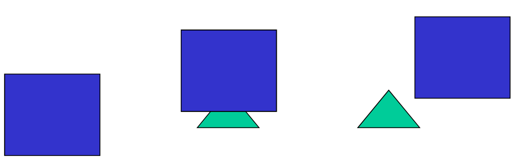



- Surfaces may be Intersecting:





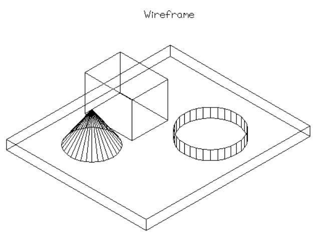

## Occlusion: Full, Partial, None



**Full** **Partial** **None**

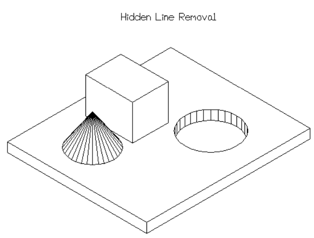
- The rectangle is closer than the triangle
- Should appear in front of the triangle

## No Lines Removed



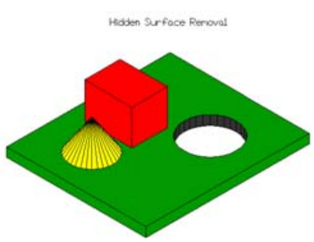



## Hidden Lines Removed






## Hidden Surfaces Removed



## Introduction

- Identifying visible parts of a scene from a viewpoint
- Numerous algorithms
  - More memory - storage
  - More processing time – execution time
  - Only for special types of objects - constraints
- Deciding a method for a particular application
  - Complexity of the scene
  - Type of objects
  - Available equipment
  - Static or animated scene



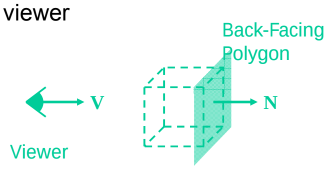
## Classification of Visible-Surface Detection Algorithms

- ***Object-space methods*** vs. ***Image-space methods***
  - Object definition directly vs. their projected images
  - Most visible-surface algorithms use image-space methods
  - Object-space can be used effectively in some cases
    - ✓ Ex) Line-display algorithms
- Object-space methods
  - Compares objects and parts of objects to each other
- Image-space methods
  - Point by point at each pixel position on the projection plane



## Back-Face Detection

- An Object space method
- Do not Render any surface oriented away from viewer




Polygon is back-facing to the viewer if $V \bullet N > 0$

## Floating Horizon Method

- Algorithm is usually implemented in image space
- **Fundamental Concept:** The technique is to convert 3D problem to equivalent 2D problem by intersecting 3D surface with a series of parallel cutting planes at constant values of the coordinate in the view direction. It could be x, y or z. The function F(x,y,z)=0 is reduced to a planar curve in each of these parallel planes

  $$y = f(x, z)$$

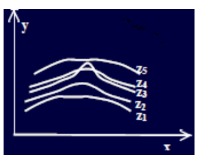
- It is assumed that the curves are single valued functions of independent variables.
- The result is projected on to the z=0 plane
- The algorithm first sorts the z=constant planes in increasing distance from the view point beginning from z=0 (closest to viewpoint)



## Floating Horizon Method

### Floating Horizon Algorithm

*With z=constant plane closest to the viewpoint, the curve in each plane is generated (for each x coordinate in image space the appropriate y value is found).*






## Floating Horizon Method

- **Upper Horizon**: If at any given value of x, the y value of the curve in the current plane is larger than the y value for any previous curve at that z value, then the curve is visible, otherwise hidden
- **Lower Horizon**: If at any given value of x, the y value of the curve in the current plane is smaller than the minimum y value for any previous curve at that z plane then the curve is visible, else hidden
- **Functional interpolation**: The algorithm assumes the value of y is available at every x location. However, if it is not available (crossing of curves), a linear interpolation of known values is calculated to fill the upper and lower floating horizon arrays.
- **Aliasing**: If the function contains very narrow regions (small increments of x) then the algorithm yields incorrect results. The effect is generally caused by computing the function for visibility at a resolution less than that of image space resolution. The problem is overcome by taking more points to evaluate the function in narrow regions.

## Scan Line Method

- An image space method for removing hidden surfaces
- As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible.
- Active edge list is maintained for each scan line which will contain only edges that cross current scan line, sorted in order of increasing x values.
- We define a flag for each surface that is set on or off to indicate whether a position is inside or outside of the surface
- Scan lines are processed from left to right. At the left most boundary of a surface, the surface flag is turned on; and at the rightmost boundary, it is turned off.
- Coherence along scan lines



## Depth Buffer Method

**Z-buffer(Depth Buffer)**

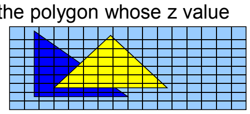
Basic Z-buffer idea:

-Rasterize every input polygon

-For every pixel in the polygon interior, calculate its corresponding z value

-Track depth values of closest polygon (smallest z) so far

-Paint the pixel with the colour of the polygon whose z value is the closest to the eye.





## Depth Buffer Method

**Algorithm**

1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x,y)

   depth(x,y) = 0 refresh(x,y)= $I_{background}$

2.For each position in each polygon surface, compare depth values to previously stored values in the depth buffer

   - Calculate the depth z for each(x,y)
   - If z > depth(x,y), then set

   depth (x,y) = z refresh(x,y) = $I_{surface(x,y)}$

## Hidden Surface Removal Cont..

**A-Buffer**

- An extension of the ideas in the depth-buffer method
- The origin of this name
  - At the other end of the alphabet from "z-buffer"
  - Antialiased, area-averaged, accumulation-buffer
  - Surface-rendering system developed by 'Lucasfilm'
    - ✓ REYES(Renders Everything You Ever Saw)
- A drawback of the depth-buffer method
  - Can't accumulate intensity values for more than one surface



## Hidden Surface Removal Cont..

- Each position in the buffer can reference a linked list of surfaces
  - Several intensities can be considered at each pixel position
  - Object edges can be antialiased
- Each position in the A-buffer has two fields
  - Depth field
    - ✓ Stores a positive or negative real number
  - Intensity field
    - ✓ Stores surface-intensity information or a pointer value



## Hidden Surface Removal Cont..

- If the depth field is positive
  - The number at that position is the depth of single surface
  - The intensity field stores the RGB
- If the depth field is negative
  - Multiple-surface contributions to the pixel
  - The intensity field stores a pointer to a linked list of surfaces
  - Data for each surface in the linked list
    - RGB intensity components
    - Opacity parameters(percent of transparency)
    - Depth
    - Percent of area coverage
    - Surface identifier
    - Pointers to next surface

## Depth Sorting Method

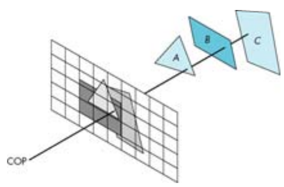
- Also called painter's algorithm
- It gets its name from the process which an artist renders a scene using oil paints. First, the artist will paint the background colors of the sky and ground. Next, the most distant objects are painted, then the nearer objects, and so forth. Note that oil paints are basically opaque, thus each sequential layer completely obscures the layer that its covers.
- Painter's Algorithm:
  - Sort surfaces in order of decreasing depth
  - Scan convert surfaces in order starting with ones of greatest depth, reordering as necessary based on overlaps



## Depth Sorting Method

The objects are then painted from back-to-front.

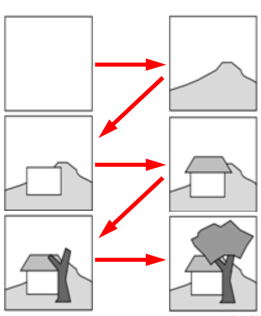
Painter's method






## Depth Sorting Method

- Main Idea
  - A painter creates a picture by drawing background scene elements before foreground ones
- Requirements
  - Draw polygons in back-to-front order
  - Need to **sort** the polygons by depth order to get a correct image



from Shirley

## Hidden Surface Removal Cont..

**BSP-trees**

Binary Space Partioning represented by a binary tree:

- efficient method for VSD
- is painting from back to front (cp. Painter's method)
- particularly useful when the Viewing Reference Point (VRP) is changing but the scene is fix (cp. moving a camera)

**Planes in BSP**

The method is based on indentifying surfaces (objects) that are *behind* or *in front of* the dividing plane relative to VRP in each step.



## Hidden Surface Removal Cont..

Each plane is subdividing the space in two sets of objects; one set of objects is behind and the other set is in front of the plane (relative to VRP)

If an object is intersected by a plane, it is divided into two separate objects

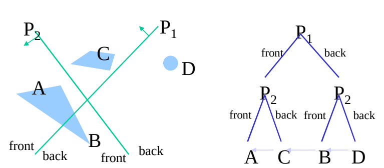
Further subdivisions will successively create a binary tree where:

- objects are terminal nodes
- left subtrees represent "front" objects
- right subtrees represent "behind" objects



## Hidden Surface Removal Cont..

BSP example



## Hidden Surface Removal Cont..

**Area Subdivision Method:**

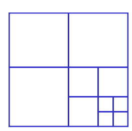
- Takes advantage of area coherence
  - Locating view areas that represent part of a single surface
  - Successively dividing the total viewing area into smaller rectangles
    - ✓ Until each small area is the projection of part of a single visible surface or no surface
  - Require tests
    - ✓ Identify the area as part of a single surface
    - ✓ Tell us that the area is too complex to analyze easily
- Similar to constructing a ***quadtree***



## Hidden Surface Removal Cont..

**Process :**

- Staring with the total view
  - Apply the identifying tests
  - If the tests indicate that the view is sufficiently complex
    - ✓ Subdivide
  - Apply the tests to each of the smaller areas
    - ✓ Until belonging to a single surface
    - ✓ Until the size of a single pixel

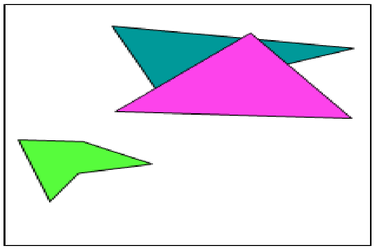




## Hidden Surface Removal Cont..

- At each stage of the algorithm, examine the areas:
  1. If no polygons lie within an area, the area is filled with the background color
  2. If only one polygon is in part of the area, the area is first filled with the background color and then the polygon is scan converted within the area.
  3. If one polygon surrounds the area and it is in front of any other polygons, the entire area is filled with the color of the surrounding polygon.
  4. Otherwise, subdivide the area and repeat the above 4 tests.

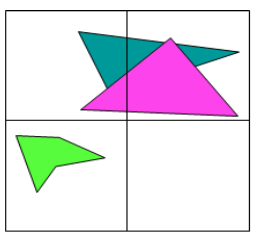
## Hidden Surface Removal Cont..



Initial scene



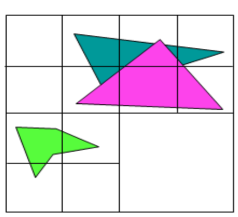
## Hidden Surface Removal Cont..



First subdivision

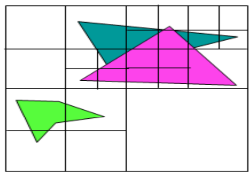


## Hidden Surface Removal Cont..



Second subdivision

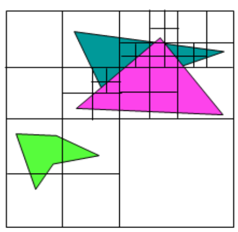
## Hidden Surface Removal Cont..



Third subdivision



## Hidden Surface Removal Cont..



Fourth subdivision

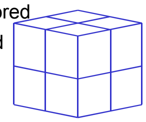


## Hidden Surface Removal Cont..

- Subdivision continues until:
  - All areas meet one of the four criteria
  - An area is pixel size
    - ✓in this case, the polygon with the closest point at that pixel determines the pixel color

## Hidden Surface Removal Cont..

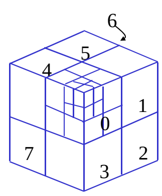
- **Octree Method**
- Extension of ***area-subdivision method***
- Projecting octree nodes onto the viewplane
  - Front-to-back order ←→ Depth-first traversal
    - ✓ The nodes for the front suboctants of octant 0 are visited before the nodes for the four back suboctants
    - ✓ The pixel in the framebuffer is assigned that color if no values have previously been stored
      - Only the front colors are loaded





## Hidden Surface Removal Cont..

- Map the octree onto a quadtree of visible areas
  - Traversing octree nodes from front to back in a recursive procedure
  - The quadtree representation for the visible surfaces is loaded into the framebuffer






## Hidden Surface Removal Cont..

**Ray-Casting Method**

- Based on geometric optics methods
  - Trace the paths of light rays
    - ✓ Line of sight from a pixel position on the viewplane through a scene
    - ✓ Determine which objects intersect this line
    - ✓ Identify the visible surface whose intersection point is closest to the pixel
  - Infinite number of light rays
    - ✓ Consider only rays that pass through pixel positions
      - Trace the light-ray paths backward from the pixels
- Effective visibility-detection method
  - For scenes with curved surfaces

## Image-Space Method vs Object-Space Method

- Image-Space Method
  - Depth-Buffer Method
  - A-Buffer Method
  - Scan-Line Method
  - *Area-Subdivision Method*
- Object-Space Method
  - Back-Face Detection
  - BSP-Tree Method
  - *Area-Subdivision Method*
  - Octree Methods
  - Ray-Casting Method



## Shading

Shading is a process used in drawing for depicting levels of darkness on paper by applying media more densely or with a darker shade for darker areas, and less densely or with a lighter shade for lighter areas.

In computer graphics, Shading refers to the process of altering the color of an object/surface/polygon in the 3D scene, based on its angle to lights and its distance from lights to create a photorealistic effect. Shading is performed during the rendering process.



## Shading Cont..

**Illumination Model for diffused Reflection**

An Illumination model is a formula in variables associated to the surface properties and light conditions to calculate the intensity of light reflected from a point on a surface. Based on standard lighting conditions in a scene.

## Shading Cont..

**Model 1: (Effect of ambient lighting & distances)**

An object may be visible even if it is not directly exposed to a light source. That is because some light Is always scattered from the nearby illuminated objects and surroundings and is known as ambient light. This light is diffused and non-directional in nature and it is assumed to be incident with uniform intensity on all objects in a scene.

So the simplest model determining the intensity ($I_{amb}$) of light reflected from a surface under ambient light condition (incident light intensity $I_a$) is given by



## Shading Cont..

$$I_{amb}=I_a k_a$$

Where '$k_a$' is the ambient-reflection coefficient of the surface material and this being a material property causes different materials looking different under same ambient light condition.



## Shading Cont..

Model2(Diffused Light)

Unlike in case of ambient lighting where distributed light source is assumed, intensity profile across a surface changes when exposed to point light source. This is because light from the source is incident at different angle at different points of the surface. The model which represents such diffused reflection is based on the **Lambert's cosine law** and is given by,

$$I_{diff}=I_1 k_d \cos\theta$$

## Shading Cont..

Where $I_{diff}$ is the reflected light intensity, $I_1$ is the incident light intenisty, $k_d$ is the diffuse reflection coefficient and θ is the angle between the incident light direction and the surface normal at the point of incidence.

If the unit vectors representing the surface normal and incident light directions are 'N' and 'L' respectively the previous equation is rewritten as:

$$I_{diff}=I_1k_d(N.L).$$



## Shading Cont..

**Model3(Phong Specular Reflection Model)**

Shiny surfaces like polished metal etc, that are not diffuse reflected. Such reflection is identified by a highlight or bright spot on the surface when viewed from particular direction(s). Most interestingly such spots appear to move over the surface with movement of the viewpoint. This is because shiny surface reflect light unequally in different directions.



## Shading Cont..

For an ideal reflector like a perfect mirror, light is reflected only in the direction of pure reflection **R**. But for shiny surfaces that exhibit specular reflection, light is reflected with varied intensity in multiple directions **V** within a limiting angle, say (plus minus) ± alpha, around **R**. This reflection domain forms a cone with **R** being the axis and semi-vertical angle being alpha. So in case of mirror reflected light can be seen only when viewed along **R** i.e. when **V** is coincident with **R** implying alpha=0. But for other shiny surfaces reflected light can be viewed not only along **R** but also over a

## Shading Cont..

finite range of viewing positions aroung **R** (deviated by angle ±alpha not equal to 0,<=90degree, >=90degree).

However the intensity of specularly reflected light is strongest along **R** and falls off rapidly with increase of alpha. This is the reason why the centre of the specular highlight is brightest. This characteristic of specular reflection is mathematically modeled by expressing the reflected light intensity (Ispec) proportional to $\cos^{n}{}_{s}$alpha where ns account for the shininess of the reflected surface. This empirical model known as **Phong Model** is given by:



## Shading Cont..

$I_{spec}=I_1 k_s \cos^{n}{}_{s}\text{alpha}$

Where $\mathbf{I_1}$ is the incident light intensity and $\mathbf{k_s}$ is the coefficient of specular reflection. If **V** and **R** are unit vectors along the viewpoint and specular reflection direction respectively then alpha is the angle between **V** and **R**. cos alpha can be replaced by **R.V** the above equation is written as

$Ispec=I_1 k_s (R.V)^{n}{}_{s}$



## Shading Cont..

**Curved Surfaces**

There are various effective methods for determining the visibility for objects with curved surfaces.

Octree is the one of the effective method for determining the visibility for objects with curved surfaces, once the representation has been established from the input definition of the object, all visible surfaces are identified with the same processing procedures. No special considerations need be given to different kinds of curved surfaces.

## Polygon-Rendering Methods

1. Constant –Intensity Shading
2. Gouraud shading
3. Phong Shading



## Constant –Intensity Shading

- Also know as flat shading
- A single intensity is calculated for each surface
- Valid if the light source & the viewer are at infinity, so that N.L and the attenuation function are constant over the surface
- The polygon is NOT an approximation of a curved surface.



## Polygon Mesh Shading

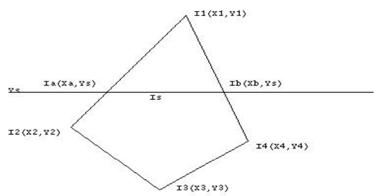
- When approximate a curved surface by a polygonal mesh, using a finer mesh turns out to be ineffective
- When the light source is not at infinity, or the viewer is local, flat shading is not quite right
- Solution: interpolated shading
  - **Gouraud shading** (smooth shading)
  - **Phong shading** (normal interpolation)

## Gouraud Shading

- Also called smooth shading, intensity interpolation shading or color interpolation shading
- Each polygon is shaded by linear interpolation of vertex intensities along each edge and then between edges along each scan line.
- The interpolation along edges can easily be integrated with the scan-line visible-surface algorithm.



## Gouraud Shading cont…




The interpolation equations are as follows:

$$I_a = \frac{1}{y_1 - y_2}[I_1(y_s - y_2) + I_2(y_1 - y_s)]$$

$$I_b = \frac{1}{y_1 - y_4}[I_1(y_s - y_4) + I_4(y_1 - y_s)]$$

$$I_s = \frac{1}{x_b - x_a}[I_a(x_b - x_s) + I_b(x_s - x_a)]$$



## Gouraud Shading cont…

- For computational efficiency these equations are often implemented as incremental calculations. The intensity of one pixel can be calculated from the previous pixel according to the increment of intensity:

$$\Delta I_s = \frac{\Delta x}{x_b - x_a}(I_b - I_a)$$

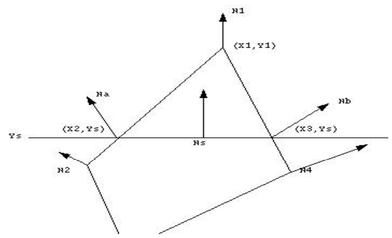
$$I_{s,n} = I_{s,n-1} + \Delta I_s$$

## Phong Shading

- Also known as normal-vector interpolation shading, interpolates the surface normal vectors rather than the color.
- Interpolation occurs across a polygon span on a scan line, between starting and ending normals for the span.
- Colors of the pixels are calculated using the interpolated normals.



## Phong Shading cont..




•Phong Shading overcomes some of the disadvantages of Gouraud Shading.
•The first stage in the process is the same as for the Gouraud Shading - for any polygon we evaluate the vertex normals.



## Phong Shading cont..

- These two vectors **Na** and **Nb** are then used to interpolate **Ns**.
- we thus derive a normal vector for each point or pixel on the polygon that is an approximation to the real normal on the curved surface approximated by the polygon.
- **Ns** , the interpolated normal vector, is then used in the intensity calculation. The vector interpolation tends to restore the curvature of the original surface that has been approximated by a polygon mesh.

## Phong Shading cont..

We have equations:

$$N_a = \frac{1}{y_1 - y_2}[N_1(y_s - y_2) + N_2(y_1 - y_s)]$$

$$N_b = \frac{1}{y_1 - y_4}[N_1(y_s - y_4) + N_4(y_1 - y_s)]$$

$$N_s = \frac{1}{x_b - x_a}[N_a(x_b - x_s) + N_b(x_s - x_a)]$$

## Conclusion

In Shading we are able to under stand Illumination Model for diffused Reflection, Effect of ambient lighting & distances, Specular Reflection Model, Curved Surfaces, Polygonal Approximations and Gourard Shading, Phong Model.

Also, Hidden Surface Removal or Visible Surface Detection comprises of following techniques.

- Floating Horizon Method
- Depth Buffer (Z-Buffer, A-Buffer) Method
- Scan Line Method, Depth Sorting Method
- BSP- tree Method
- Area Subdivision Method

## Review Questions Cont..

**Short answer type Questions**

Q1. What do you mean by Shading?

Q2. Give Lambert's cosine law for reflection?

Q3. Explain Phong Model?

Q4. Explain Warnock's Algorithm?

Q5. Where do we need Douglas-Peucker algorithm? Explain

Q6. What do you mean by Floating Horizon Method for HSR?

## Review Questions cont..

**Long answer type Questions**

Q1. Write and Explain various Illumination Model for diffused Reflection.

Q2. What do you mean by HSR? Explain various techniques of VSD?

Q3. Explain the A and Z buffer Techniques of HSR? Also, Differentiate the above two techniques.

Q4. Discuss Painter's Method with the help of an example.

Q5. Explain Polygonal Approximations with the help of an example.



## Suggested Reading/References

[1]. Donnald Hearn and M. Pauline Baker, "Computer Graphics", PHI.

[2]. Foley James D, "Computer Graphics", AW 2nd Ed.

[3]. Rogers, "Procedural Element of Computer Graphics", McGraw Hill.

[4]. Newman and Sproul, "Principal of to Interactive Computer Graphics", McGraw Hill.