SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | Approved by AICTE

COMPUTER GRAPHICS & MULTIMEDIA SYSTEMS
SCS1302

UNIT IV – PART I

Syllabus

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY FACULTY OF COMPUTING

SCS1302	COMPUTER GRAPHICS AND MULTIMEDIA SYSTEMS	L	T	P	Credits	Total Marks
		3	0	0	3	100

COURSE OBJECTIVES

- To gain knowledge to develop, design and implement two and three dimensional graphical structures.
- To enable students to acquire knowledge of Multimedia compression and animations.
- To learn creation, Management and Transmission of Multimedia objects.

UNIT 1 BASICS OF COMPUTER GRAPHICS **9 Hrs.**

Output Primitives: Survey of computer graphics - Overview of graphics systems - Line drawing algorithm - Circle drawing algorithm - Curve drawing algorithm - Attributes of output primitives - Anti-aliasing.

UNIT 2 2D TRANSFORMATIONS AND VIEWING **8 Hrs.**

Basic two dimensional transformations - Other transformations - 2D and 3D viewing - Line clipping - Polygon clipping - Logical classification - Input functions - Interactive picture construction techniques.

UNIT 3 3D CONCEPTS AND CURVES **10 Hrs.**

3D object representation methods - B-REP , sweep representations, Three dimensional transformations. Curve generation - cubic splines, Beziers, blending of curves- other interpolation techniques, Displaying Curves and Surfaces, Shape description requirement, parametric function. Three dimensional concepts – Introduction - Fractals and self similarity- Successive refinement of curves, Koch curve and peano curves.

Syllabus

UNIT 4 METHODS AND MODELS **8 Hrs.**

Visible surface detection methods - Illumination models - Halftone patterns - Dithering techniques - Polygon rendering methods - Ray tracing methods - Color models and color applications.

UNIT 5 MULTIMEDIA BASICS AND TOOLS **10 Hrs.**

Introduction to multimedia - Compression & Decompression - Data & File Format standards - Digital voice and audio - Video image and animation. Introduction to Photoshop - Workplace - Tools - Navigating window - Importing and exporting images - Operations on Images - resize, crop, and rotate - Introduction to Flash - Elements of flash document - Drawing tools - Flash animations - Importing and exporting - Adding sounds - Publishing flash movies - Basic action scripts - GoTo, Play, Stop, Tell Target

Max. 45 Hours

TEXT / REFERENCE BOOKS

1. Donald Hearn, Pauline Baker M., "Computer Graphics", 2nd Edition, Prentice Hall, 1994.
2. Tay Vaughan ,"Multimedia", 5th Edition, Tata McGraw Hill, 2001.
3. Ze-Nian Li, Mark S. Drew ,"Fundamentals of Multimedia", Prentice Hall of India, 2004.
4. D. McClelland, L.U.Fuller ,"Photoshop CS2 Bible", Wiley Publishing, 2005.
5. James D. Foley, Andries van Dam, Steven K Feiner, John F. Hughes, "Computer Graphics Principles and Practice, 2nd
6. Edition in C, Audison Wesley, ISBN - 981 -235-974-5
7. William M. Newman, Roberet F. Sproull, " Principles of Interactive Computer Graphics", Second Edition, Tata McGraw-Hill Edition.

Course Objective(CO)

CO1: Construct lines and circles for the given input.

CO2: Apply 2D transformation techniques to transform the shapes to fit them as per the picture definition.

CO3: Construct splines, curves and perform 3D transformations

CO4: Apply colour and transformation techniques for various applications.

CO5: Analyse the fundamentals of animation, virtual reality, and underlying technologies.

CO6: Develop photo shop applications

VISIBLE SURFACE DETECTION METHODS

Basic classification

- Two basic classifications –based on either an object or projected image , which is going to be displayed

i) **Object-space Methods-** compares objects and parts of objects to each other within a scene definition to determine which surfaces are visible

ii) **Image-space Methods-** visibility is determined point-by-point at each pixel position on the projection plane

Image-space Method is the most commonly used method

Back-face detection

- **Back-Face detection**, also known as **Plane Equation method,** is an object space method in which objects and parts of objects are compared to find out the visible surfaces.
- Let us consider a triangular surface that whose visibility needs to decided.
- The idea is to check if the triangle will be facing away from the viewer or not.
- If it does so, discard it for the current frame and move onto the next one.
- Each surface has a normal vector. If this normal vector is pointing in the direction of the center of projection, then it is a front face and can be seen by the viewer.
- If this normal vector is pointing away from the center of projection, then it is a back face and can not be seen by the viewer.

Back-face detection

- To find the back faces of a polyhedron.
- Consider a polygon surface with parameters A,B,C and D.
- A point (x,y,z) is inside the polyhedrons' backface only if

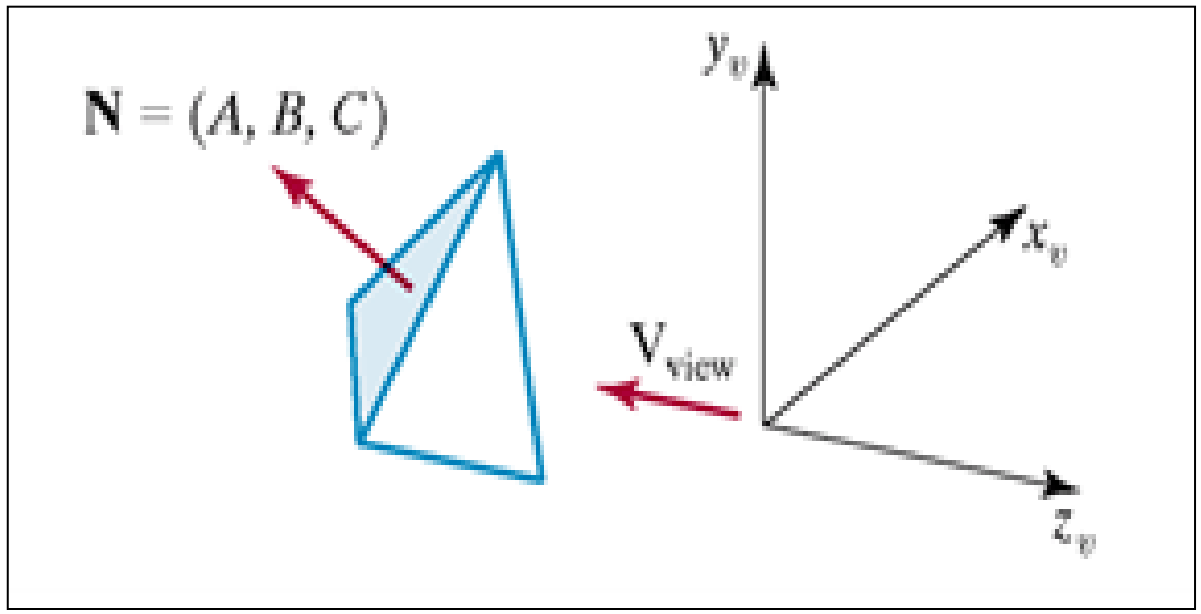
$$Ax+By+Cz+D<o$$

- We can simply say that the z component of the polygon's normal is less than zero, then the point is on the back face.

Back-face detection

- Object space algorithm: Back-Face removal
- No faces on the back of the object are displayed.
- In general - about half of objects faces are back faces
- Algorithm will remove about half of the total polygons in the image from further visibility tests.

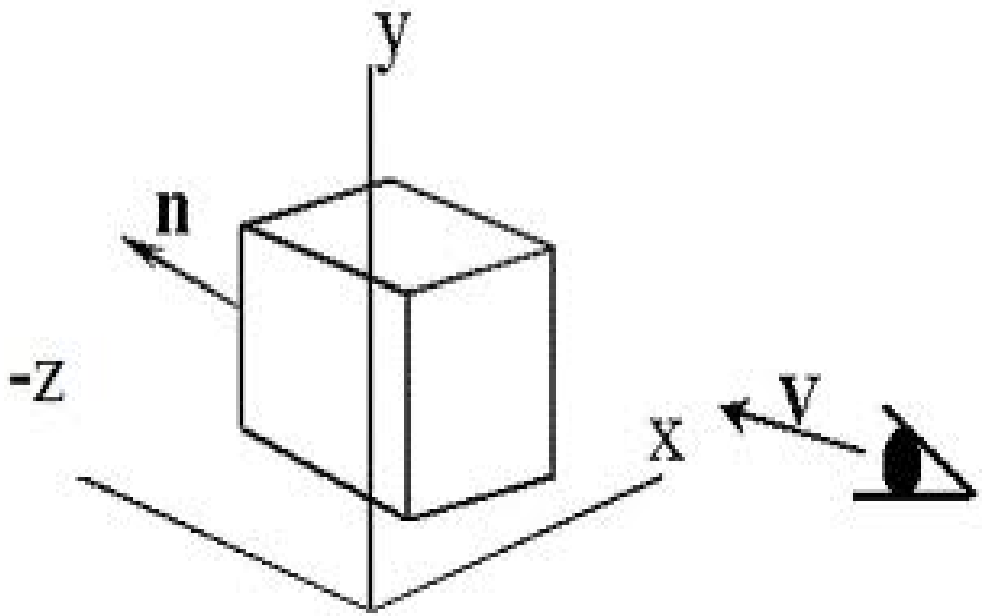
NOTE: Right Handed coordinate System



- If we take V as the vector in the viewing direction from the eye and N as the normal vector to the polygon's surface, then we can state the condition as

$$V.N>0$$

then that face is the back face.



Back-face detection

- Thus, for the right-handed system, if the Z component of the normal vector is negative, then it is a back face. If the Z component of the vector is positive, then it is a front face.

Algorithm for right-handed system :

```
1) Compute N for every face of object.
2) If (C.(Z component) < 0)
        then a back face and don't draw
   else
        front face and draw
```

Back-face detection

- The Back-face detection method is very simple. For the left-handed system, if the Z component of the normal vector is positive, then it is a back face. If the Z component of the vector is negative, then it is a front face.

Algorithm for left-handed system :

```
1) Compute N for every face of object.
2) If (C.(Z component) > 0)
        then a back face and don't draw
   else
        front face and draw
```

Back-face detection

Recalling the polygon surface equation :

```
Ax + By + Cz + D < 0
```

While determining whether a surface is back-face or front face, also consider the viewing direction. The normal of the surface is given by :

```
N = (A, B, C)
```

A polygon is a back face if $V_{view}.N > 0$. But it should be kept in mind that after application of the viewing transformation, viewer is looking down the negative Z-axis. Therefore, a polygon is back face if :

```
(0, 0, -1).N > 0
or if C < 0
```

Viewer will also be unable to see surface with C = 0, therefore, identifying a polygon surface as a back face if : C <= 0.

If $\mathbf{n} \bullet \mathbf{V} > 0$ then polygon is backface

$$\mathbf{n} \bullet \mathbf{V} = |\mathbf{n}| \, |\mathbf{V}| \cos \theta$$

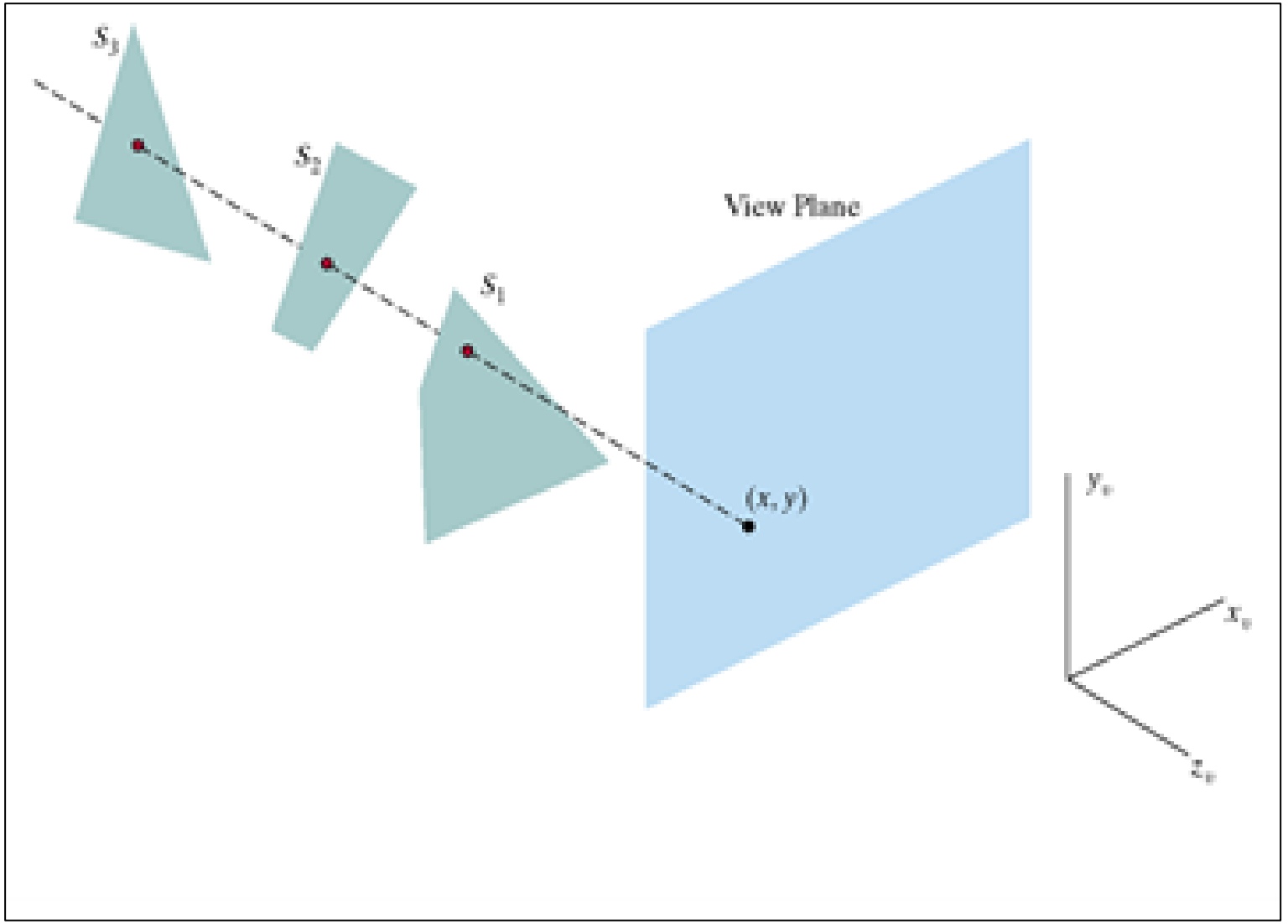
$$\cos \theta = \mathbf{n} \bullet \mathbf{V} / (|\mathbf{n}| \, |\mathbf{V}|)$$

where $|\mathbf{n}|$ and $|\mathbf{V}|$ are the magnitudes of the vector

$$|\mathbf{V}| = \text{sqrt}(V_x^2 + V_y^2 + V_z^2)$$

Depth buffer method (or) z-buffer method

- Image Space Method.
- Compares surface depths at each pixel position throughout the scene on the projection plane.
- It is usually applied to images containing polygons.
- Very fast method.



Depth buffer algorithm

- For each projected (x, y) pixel position of a polygon, calculate the depth z (if not already known)
- If z < depth(x, y), compute the surface colour at that position and set

  depth(x, y) = z

  frameBuff(x, y) = surfColour(x, y)

- At a surface position (x,y) the depth is calculated from the plane equation:

$$z = \frac{-Ax - By - D}{C}$$

- the depth of the next position z' becomes, for each successive position along scanline:

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

$$z' = z - \frac{A}{C}$$

Depth buffer algorithm

Possible algorithm:

1. Begin at top vertex of polygon
2. Recursively calculate x-coordinate values down left edge of polygon
3. Subsequent x-values for each scanline calculated from starting x-value

$$x' = x - \frac{1}{m}$$

and depth values become

$$z' = z + \frac{A/m + B}{C}$$

- **Advantage**
  Always works and is simple to implement => used in hardware
- **Disadvantages**
  - Large memory requirements

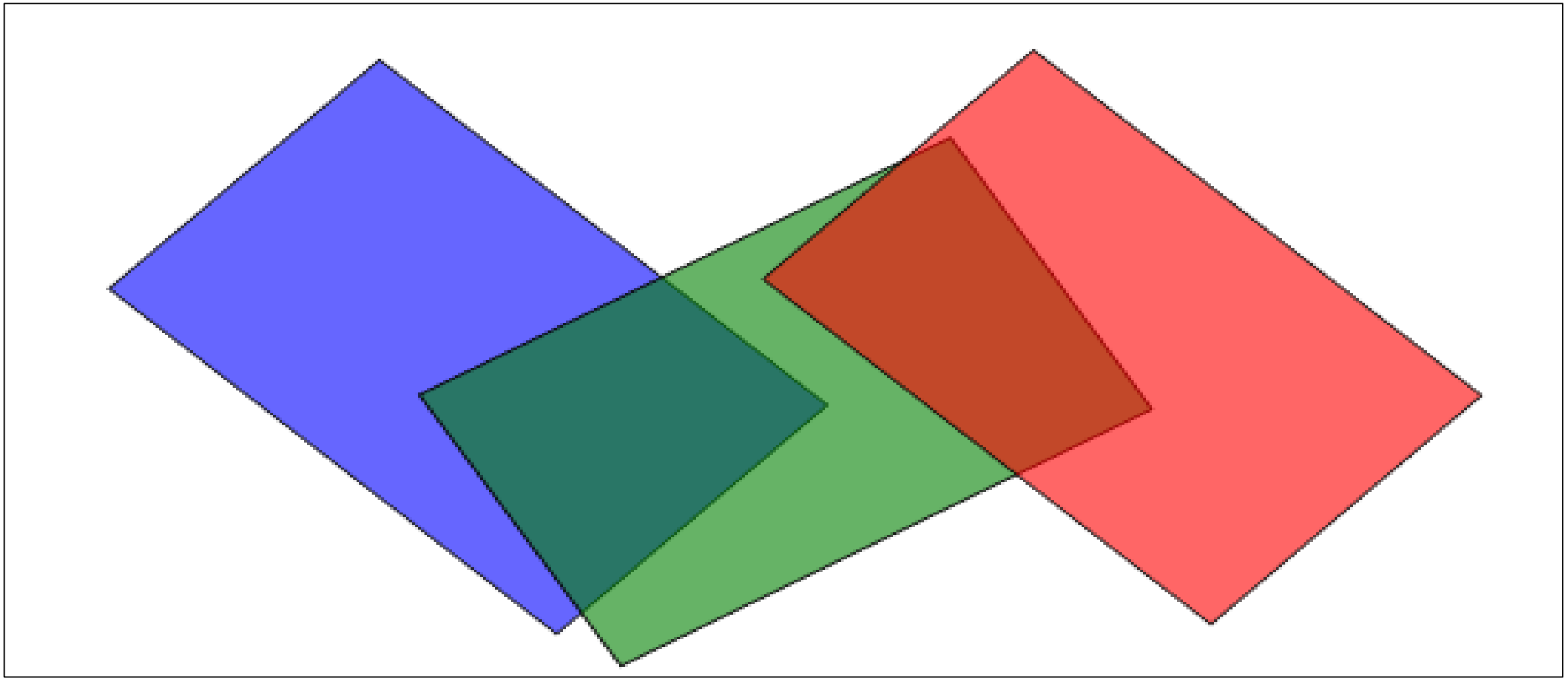
Disadvantages

- The depth buffer Algorithm is not always practical because of the enormous size of depth and intensity arrays.
- Generating an image with a raster of 500 x 500 pixels requires 2, 50,000 storage locations for each array.
- Even though the frame buffer may provide memory for intensity array, the depth array remains large.
- To reduce the amount of storage required, the image can be divided into many smaller images, and the depth buffer algorithm is applied to each in turn.
- For example, the original 500 x 500 faster can be divided into 100 rasters each 50 x 50 pixels.
- Processing each small raster requires array of only 2500 elements, but execution time grows because each polygon is processed many times.
- Subdivision of the screen does not always increase execution time instead it can help reduce the work required to generate the image. This reduction arises because of coherence between small regions of the screen.

A-buffer Method

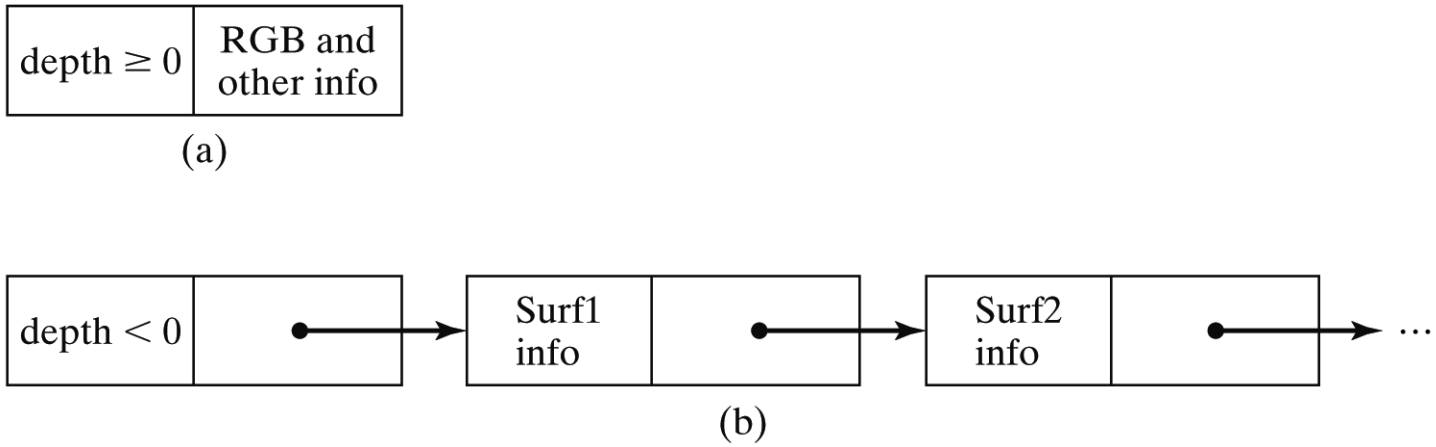
- Extension of Depth-buffer -> **accumulation buffer(A-Buffer)**
- **A-** Antialiased, Area-averaged, Accumulation – Buffer.
- Drawback of Depth-buffer-can find one visible surface at each pixel position. Cannot accumulate intensity values for more than one surface.
- Each buffer position can reference a linked-list of surfaces.

A-buffer



A-buffer

- Each position has 2 fields.
- Depth field – stores a positive or negative real number
- Intensity field- stores surface intensity information or a pointer value.

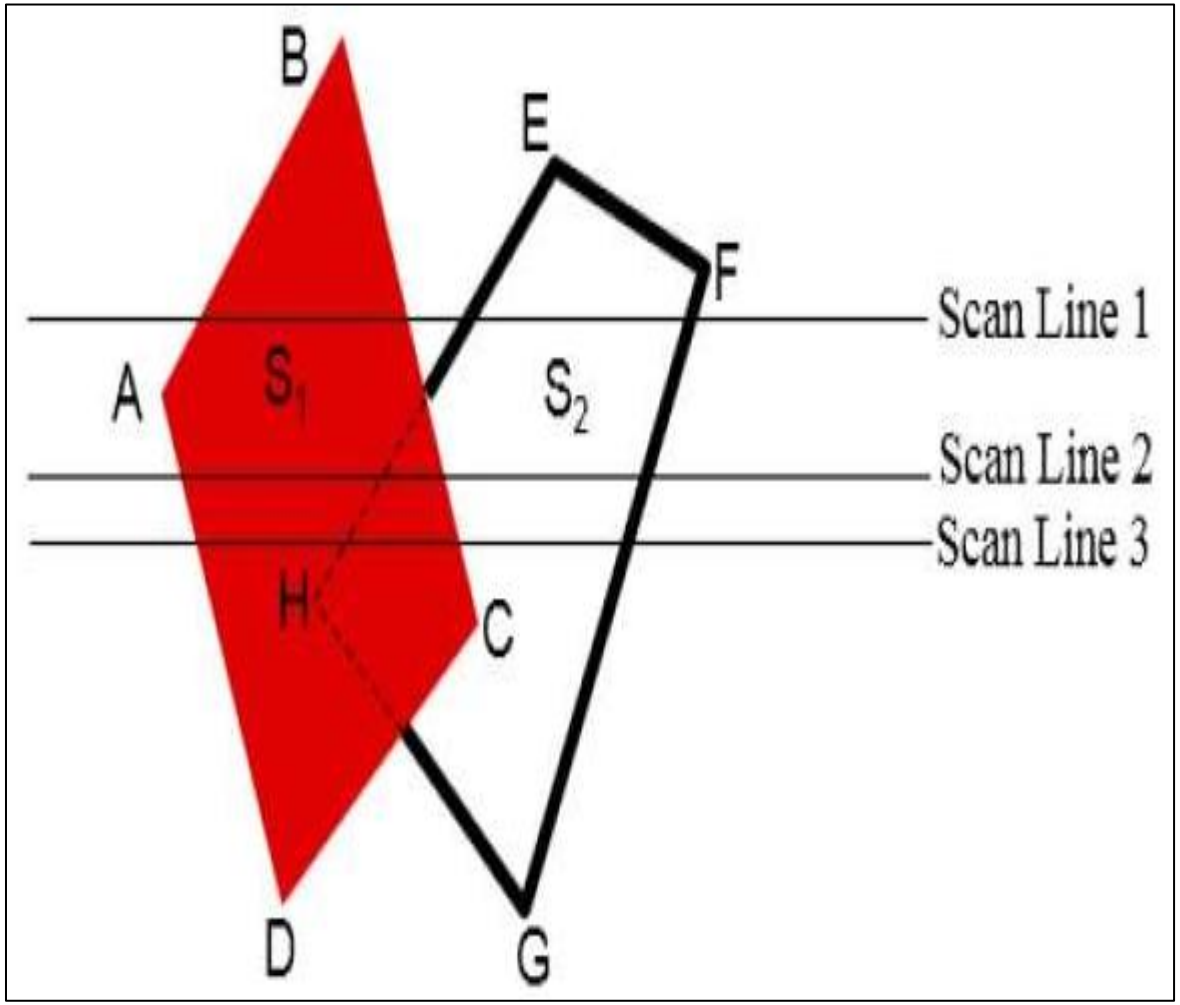


A-buffer

- If depth is >= 0(single surface), then the surface data field stores the depth of that pixel position as before
- If depth < 0 (multiple surfaces)then the data filed stores a pointer to a linked list of surface data.
- Surface information in the A-buffer includes:
  - RGB intensity components
  - Opacity parameter
  - Depth
  - Percent of area coverage
  - Surface identifier
  - Other surface rendering parameters
- The algorithm proceeds just like the depth buffer algorithm
- The depth and opacity values are used to determine the final colour of a pixel

Scan-line method

- Idea is to intersect each polygon with a particular scanline. Solve hidden surface problem for just that scanline.
- Requires a depth buffer equal to only one scanline
- The cost of tiling scene is roughly proportional to its depth complexity
- Efficient way to tile shallowly-occluded scenes



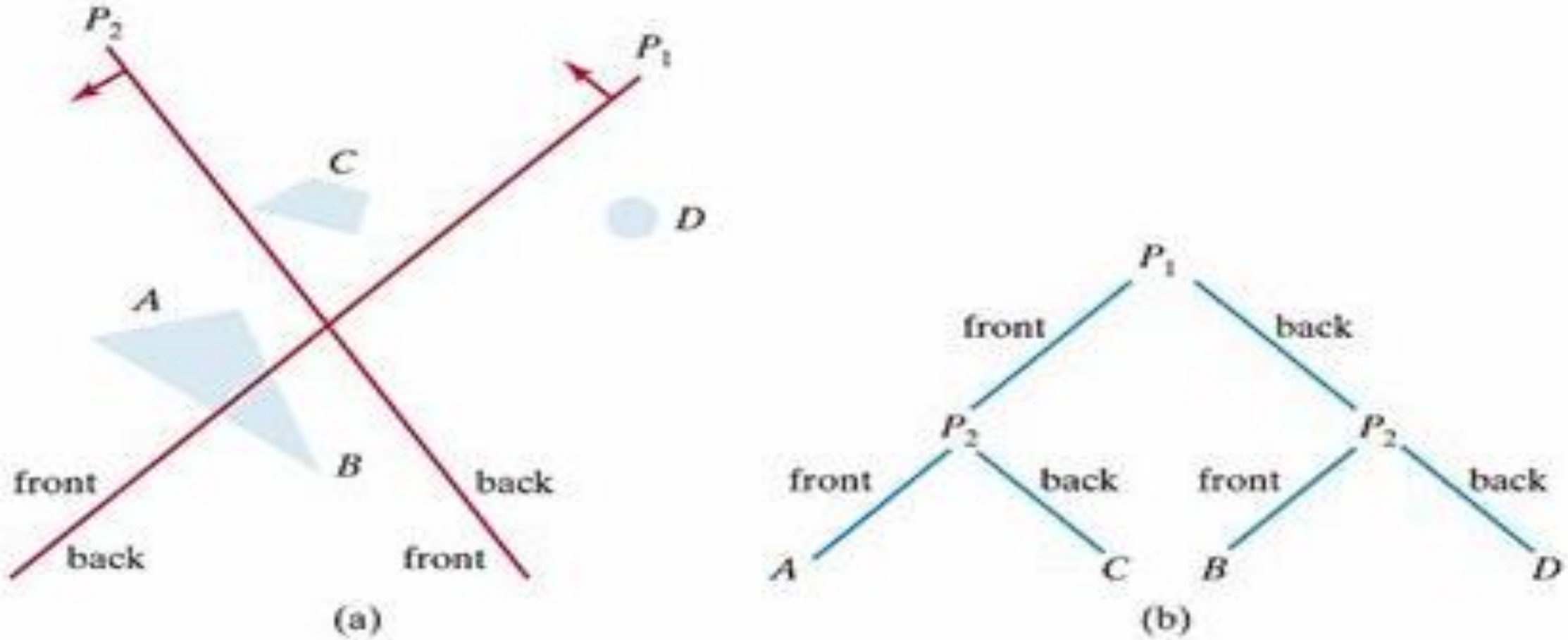
Scan-line method

- It is an image-space method to identify visible surface. This method has a depth information for only single scan-line.
- In order to require one scan-line of depth values, we must group and process all polygons intersecting a given scan-line at the same time before processing the next scan-line.
- Two important tables, **edge table** and **polygon table,** are maintained for this.
- To facilitate the search for surfaces crossing a given scan-line, an active list of edges is formed.

- The active list stores only those edges that cross the scan-line in order of increasing x.
- Also a flag is set for each surface to indicate whether a position along a scan-line is either inside or outside the surface.
- Pixel positions across each scan-line are processed from left to right.
- At the left intersection with a surface, the surface flag is turned on and at the right, the flag is turned off.
- You only need to perform depth calculations when multiple surfaces have their flags turned on at a certain scan-line position.

BSP Tree Method

- A binary space partitioning tree (bsp–tree) is a binary tree whose nodes contain polygons.
- Binary space partitioning, or BSP, divides space into distinct sections by building a tree representing that space.
- Used to sort polygons.
- A BSP takes the polygons and divides them into two groups bychoosing a plane, usually taken from the set of polygons,
 and divides the world into two spaces.
- It decides which side of the plane each polygon is on, or it may also be on the plane.
- If a polygon intersects the splitting plane it must be split into two separate polygons, one on each side of the plane.
- The tree is built by choosing a partitioning plane and dividing the remaining polygons into two or three lists: Front, Back and On lists – done by comparing the normal vector of the plane with that of each polygon.



A region of space (a) is partitioned with two planes P_1 and P_2 to form the BSP tree representation shown in (b).

BSP Tree Method

- For each node in a bsp-tree the polygons in the left subtree lie behind the polygon at the node while the polygons in the right
subtree lie in front of the polygon at the node.
- Each polygon has a fixed normal vector, and front and back are measured relative to this fixed normal.
- Once a bsp-tree is constructed for a scene, the polygons are rendered by an in order traversal of the bsp-tree.
- Recursive algorithms for generating a bsp-tree and then using the bsp-tree to render a scene are presented below.

Algorithm for Generating a BSP–Tree

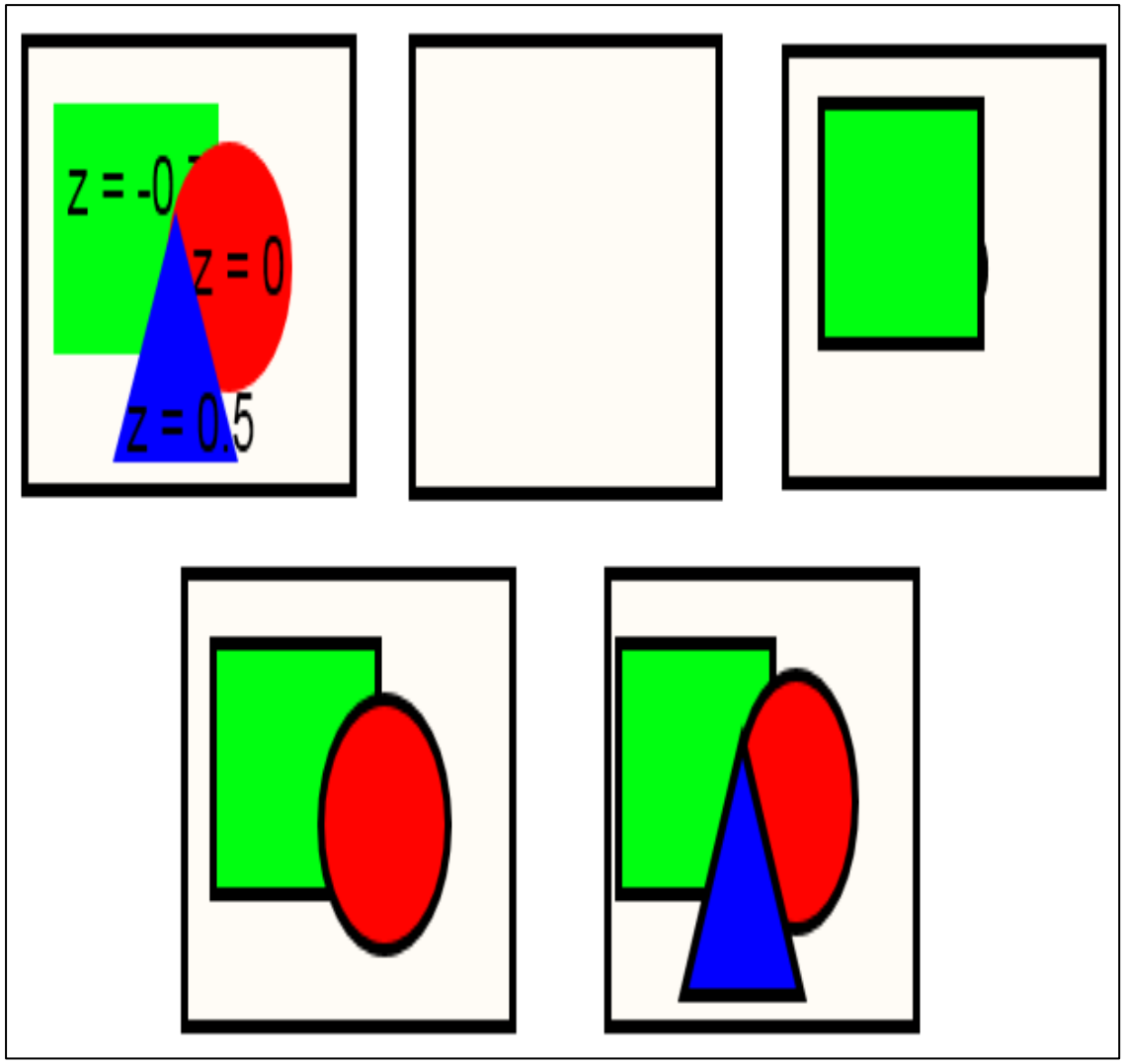
1.Select any polygon (plane) in the scene for the root.
2.Partition all the other polygons in the scene to the back (left subtree) or the front (right subtree).
3.Split any polygons lying on both sides of the root (see below).
4.Build the left and right subtrees recursively.

***BSP-Tree Rendering Algorithm* (In order tree traversal)**

If the eye is in front of the root, then
Display the left subtree (behind)
Display the root
Display the right subtree (front)
If eye is in back of the root, then
Display the right subtree (front)
Display the root
Display the left subtree (back)

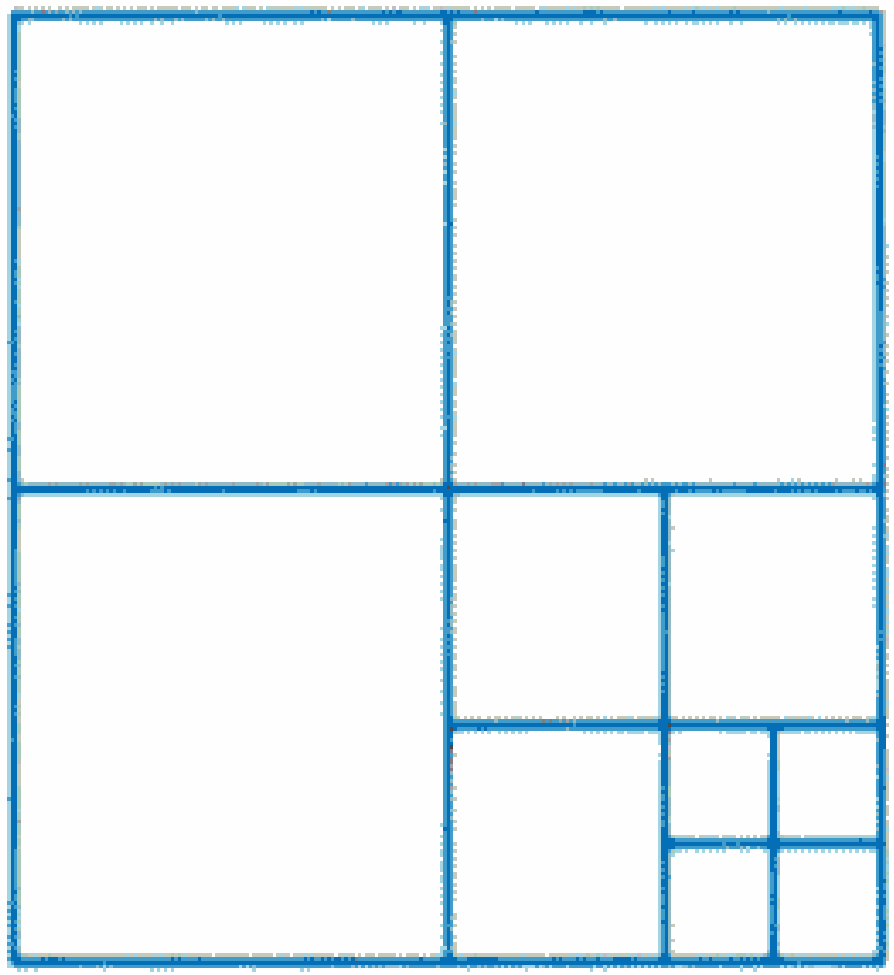
Depth- sorting or painter’s algorithm

- Depth sorting method uses both image space and object-space operations. The depth-sorting method performs two basic functions –
- First, the surfaces are sorted in order of decreasing depth.
- Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.
- The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the **painter's algorithm**.



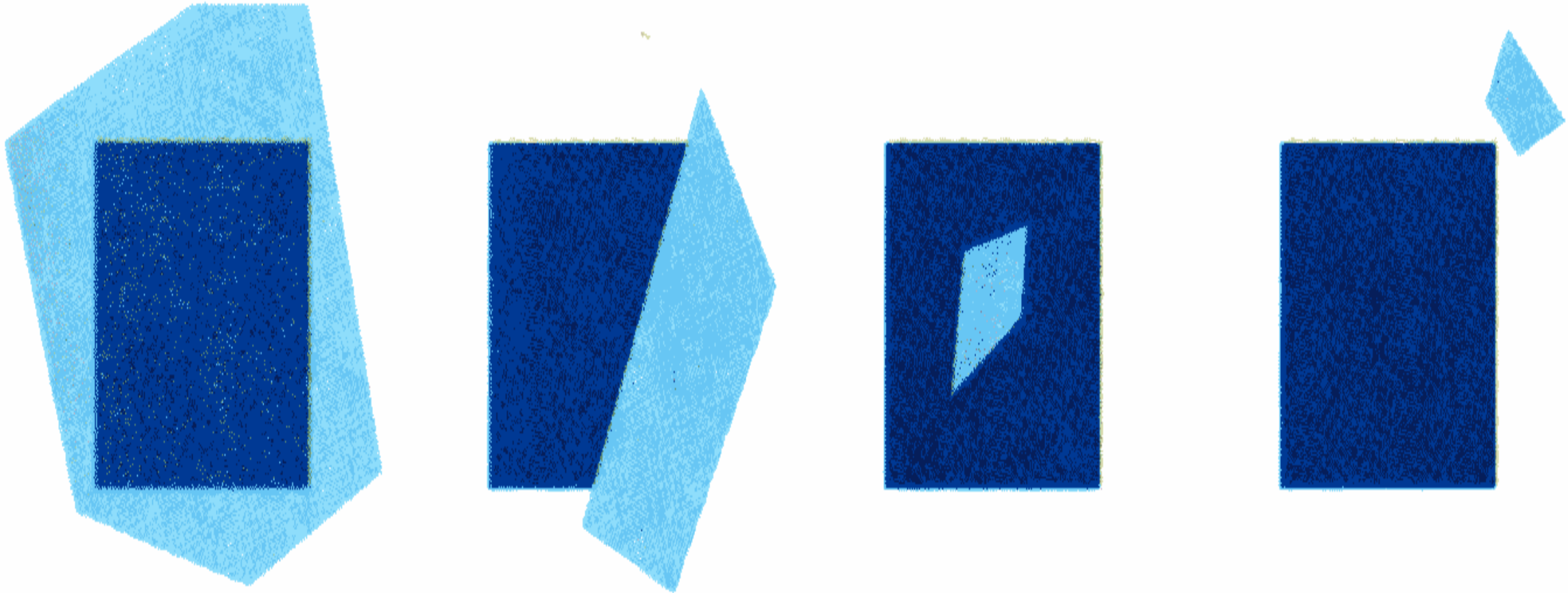
Area subdivision method

- The area-subdivision method takes advantage by locating those view areas that represent part of a single surface.
- Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.
- Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel.
- An easy way to do this is to successively divide the area into four equal parts at each step.
- There are four possible relationships that a surface can have with a specified area boundary.



Dividing a square area into equal-sized quadrants at each step.

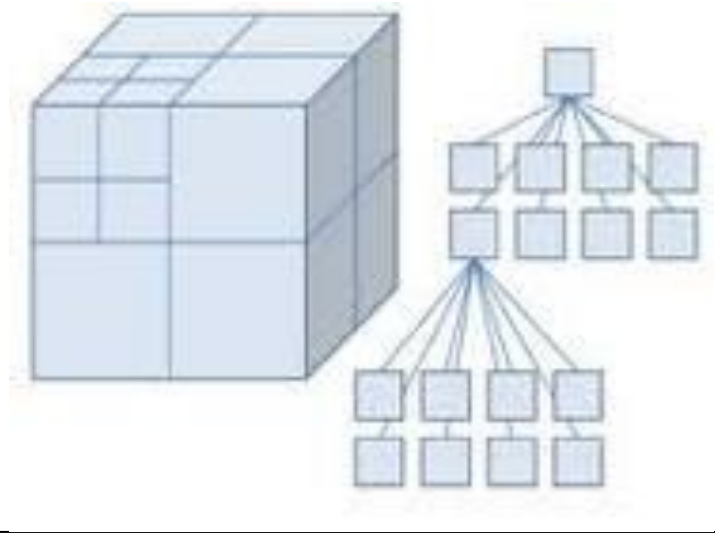
- The tests for determining surface visibility within an area can be stated in terms of these four classifications. No further subdivisions of a specified area are needed if one of the following conditions is true −
- All surfaces are outside surfaces with respect to the area.
- Only one inside, overlapping or surrounding surface is in the area.
- A surrounding surface obscures all other surfaces within the area boundaries.

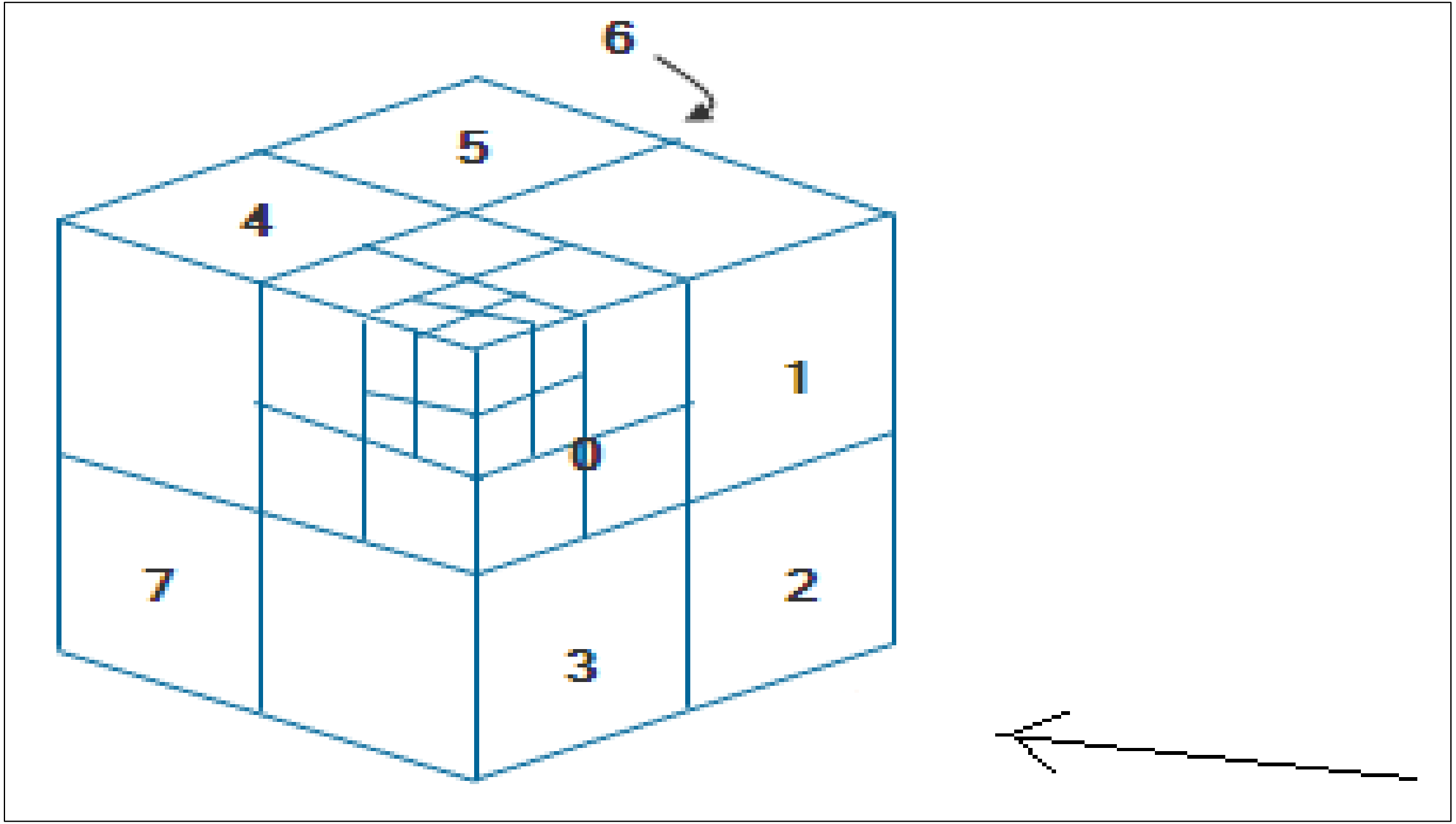


Possible relationships between polygon surfaces and a rectangular area.

Octree methods

- 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.




6
5
4
1
0
7
2
3

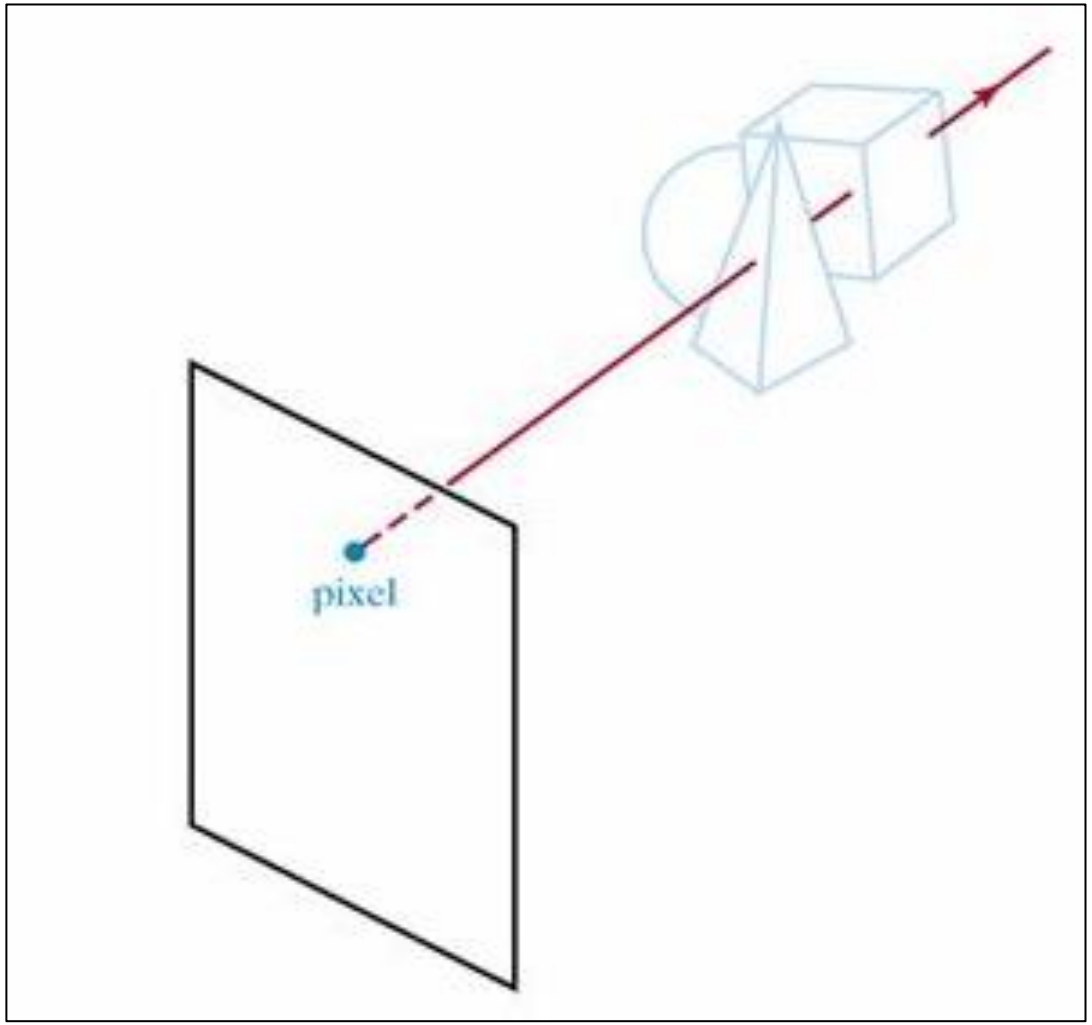
Ray-casting method

- 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.

Ray-casting method

· The ray casting algorithm for hidden surfaces employs no special data structures.
· A ray is fired from the eye through each pixel on the screen in order to locate the polygon in the scene closest to the eye.
· The color and intensity of this polygon is displayed at the pixel.

Ray casting is easy to implement for polygonal models
because the only calculation required is the intersection of a line with a plane.

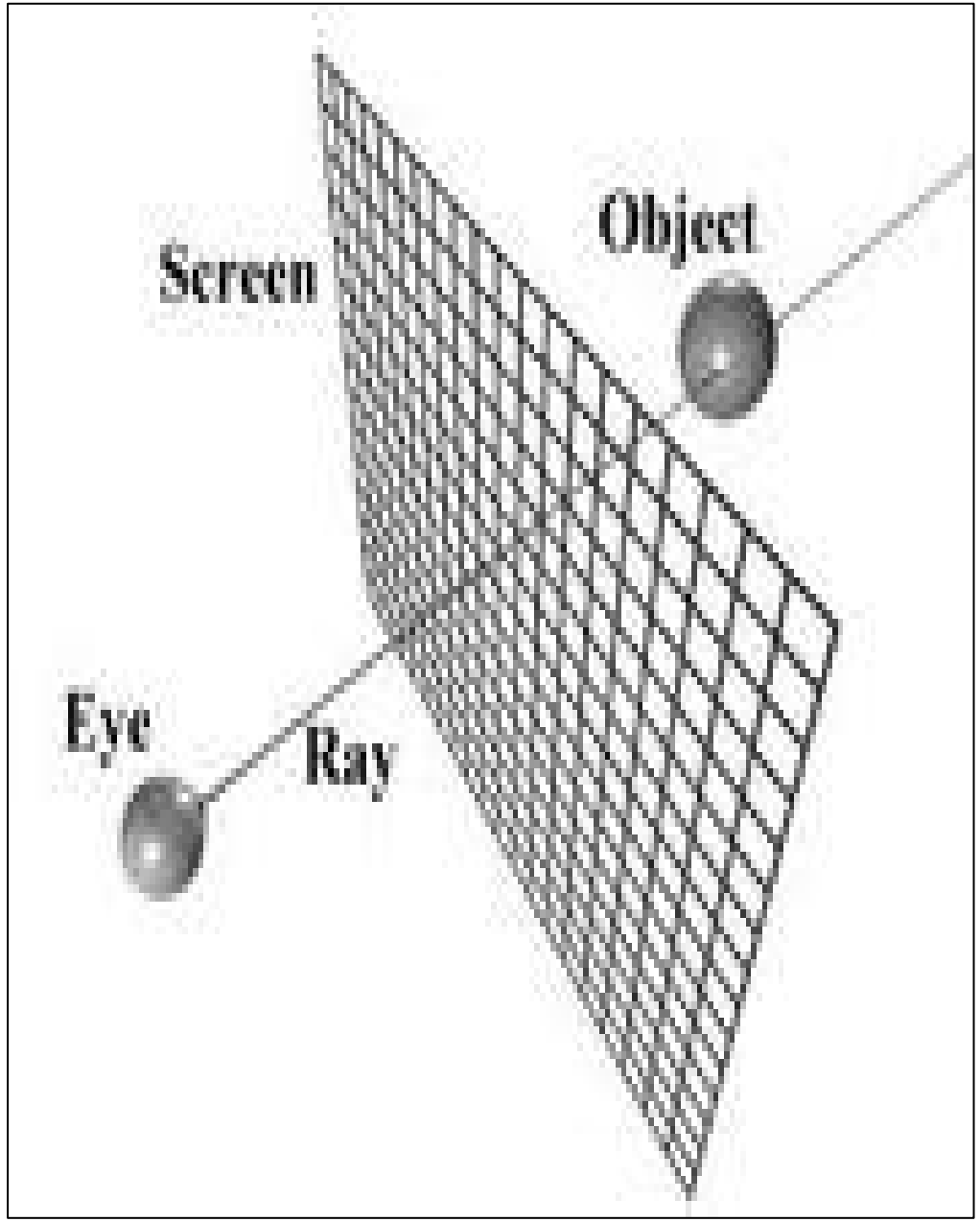


Ray Casting Algorithm

Algorithm

1.Through each pixel, fire a ray to the eye:
2.Intersect the ray with each polygonal plane.
3.Reject intersections that lie outside the polygon.
4.Accept the closest remaining intersection -- that is, the intersection with the smallest value of the parameter along the line.

- The main advantage of the ray casting algorithm for hidden surfaces is that ray casting can be used even with non-polygonal surfaces.
- All that is needed to implement the ray casting algorithm for hidden surfaces is a line/surface intersection algorithm for each distinct surface type.
- The main disadvantage of ray casting is that the method is slow. Ray casting is a brute force technique that makes no use of pixel coherence.


Screen
Object
Eye
Ray

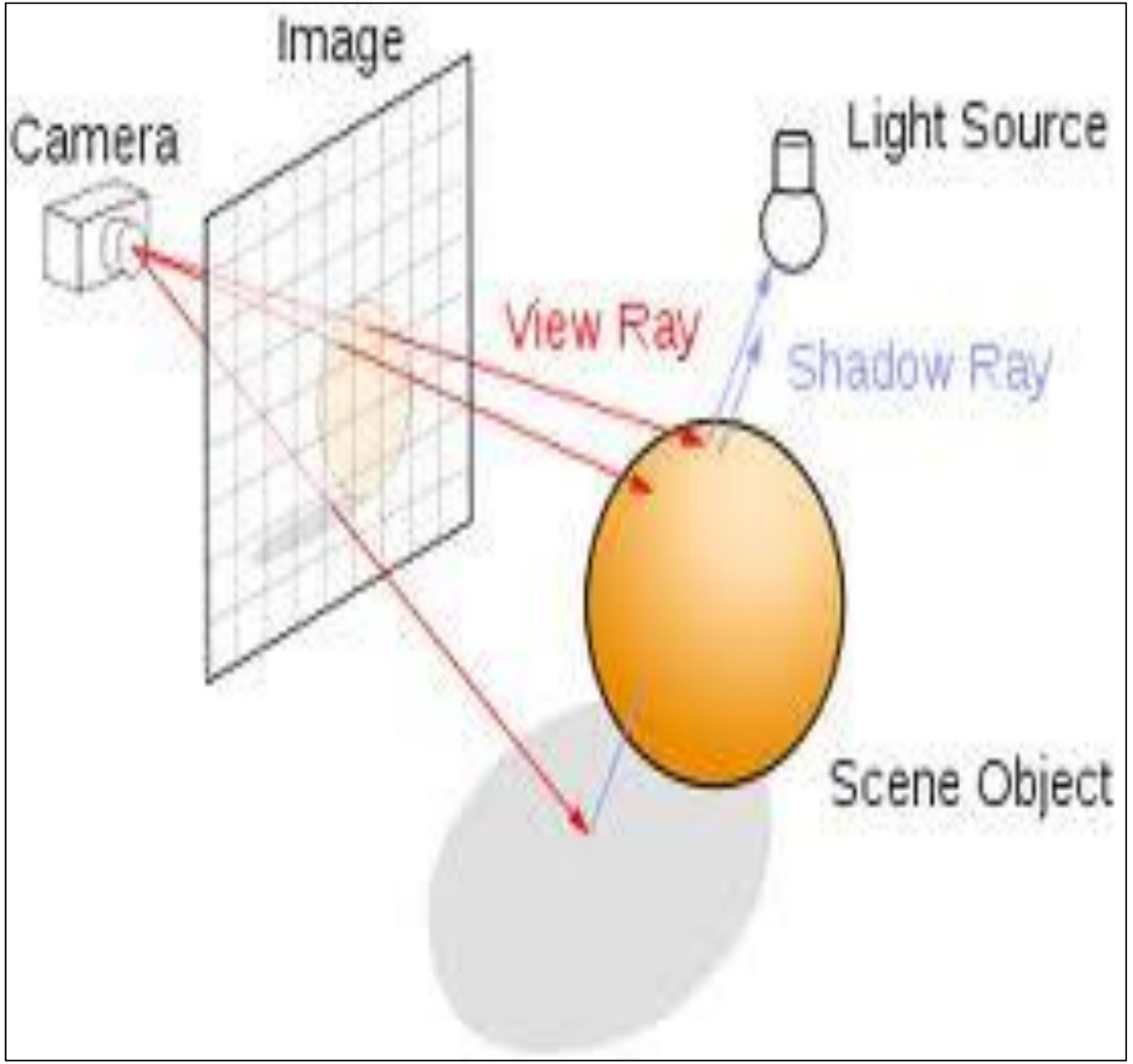


Image
Camera
Light Source
View Ray
Shadow Ray
Scene Object

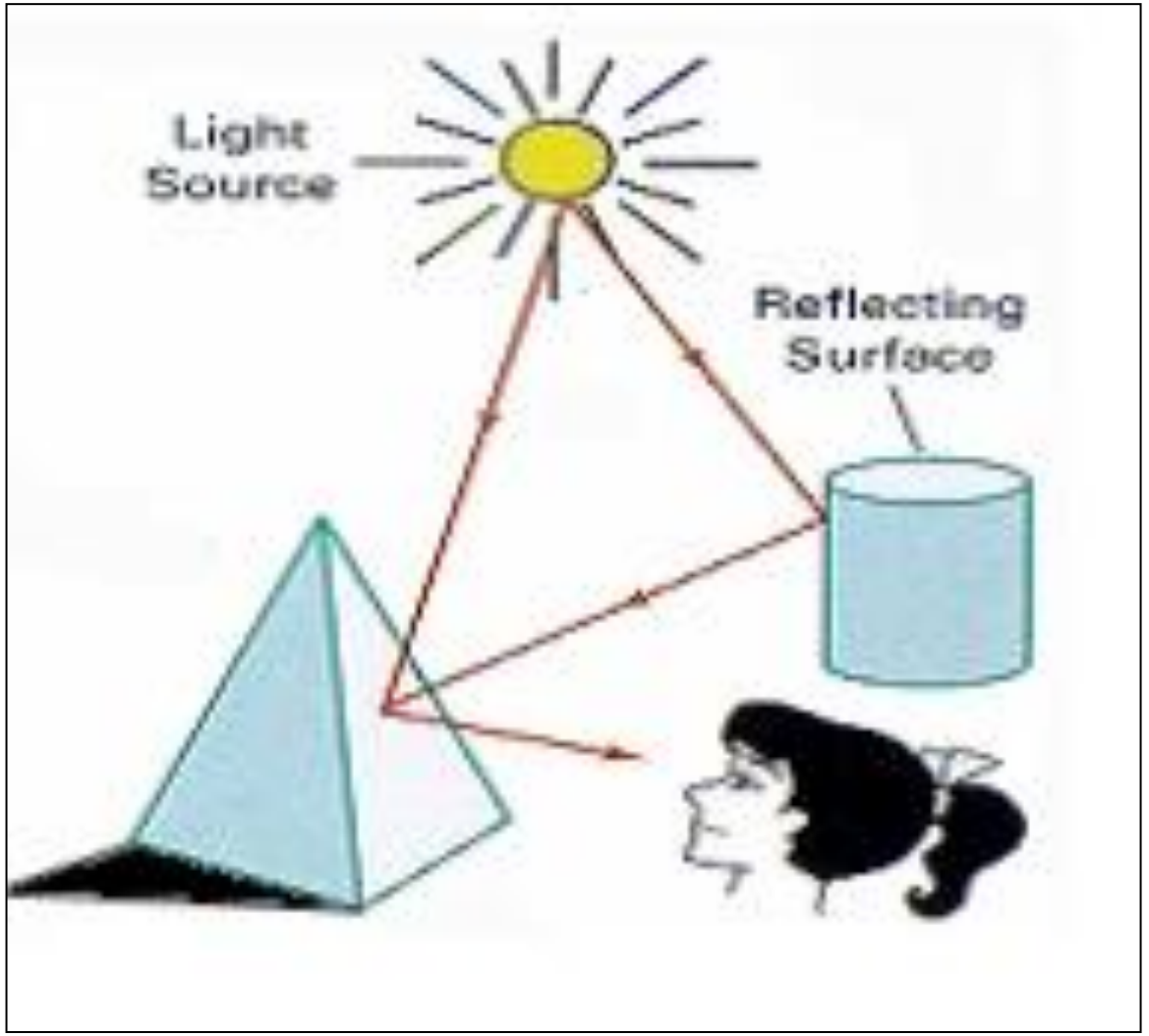
ILLUMINATION MODELS

ILLUMINATION MODELS

- An illumination model (lighting model) is used to calculate the color of the illuminated positions of an object.
- The procedure for applying a lighting model to obtain a pixel colors for all projected surface positions is called **surface rendering**.
- In computer graphics, illumination models are usually easy to compute approximations of physical laws that describe surface-effect lighting.
- A surface-rendering algorithm uses the intensity calculations from an illumination model to determine the light intensity for all projected pixel positions for the various surfaces in a scene.
- Surface rendering can be performed by applying the illumination model to every visible surface point.

ILLUMINATION MODELS

- **Illumination model**, also known as Shading model or Lightning model, is used to calculate the intensity of light that is reflected at a given point on surface. There are three factors on which lightning effect depends on:
    1. Light Source
    2. Surface
    3. Observer



1. LIGHT SOURCES

- Light Emitting sources – Emits the light
- Light Reflecting sources – Reflecting surfaces such as walls of a room.

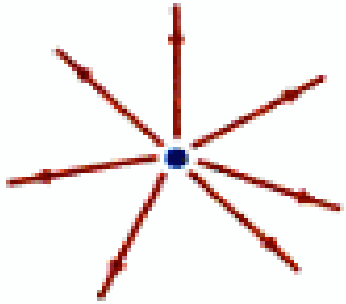
1. LIGHT SOURCES: Point Source and Distributed Source

1.Point Sources – The source that emit rays in all directions (A bulb in a room).

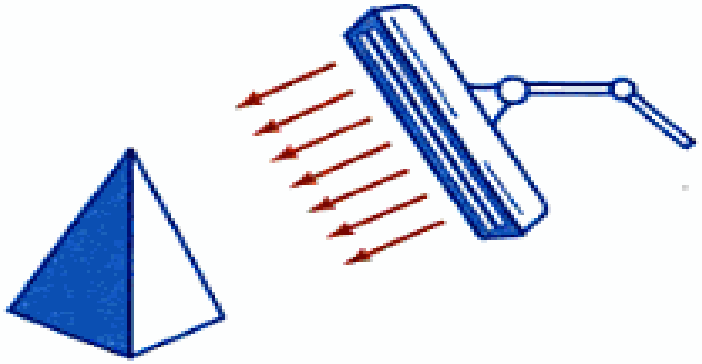
2.Parallel Sources – Can be considered as a point source which is far from the surface (The sun).

3.Distributed Sources – Rays originate from a finite area (A tubelight).

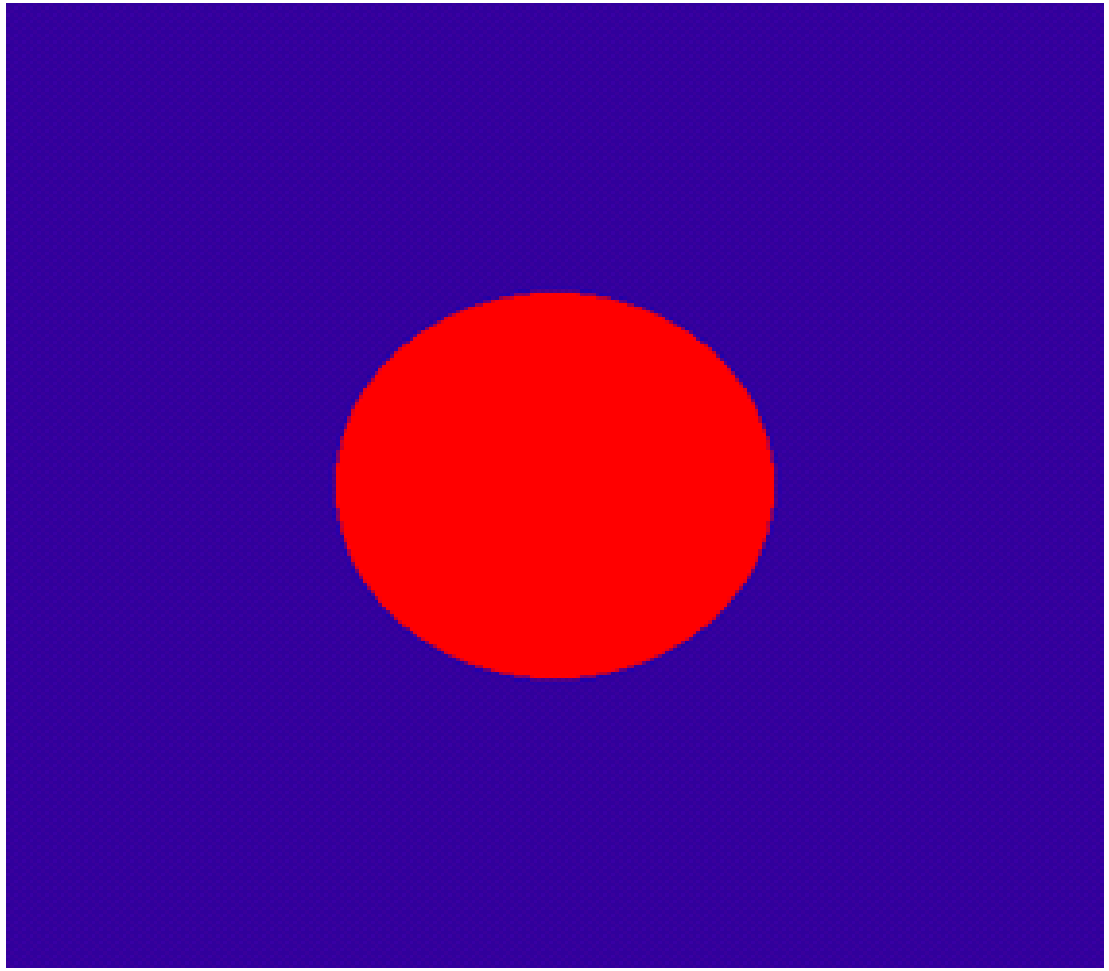
Point Source

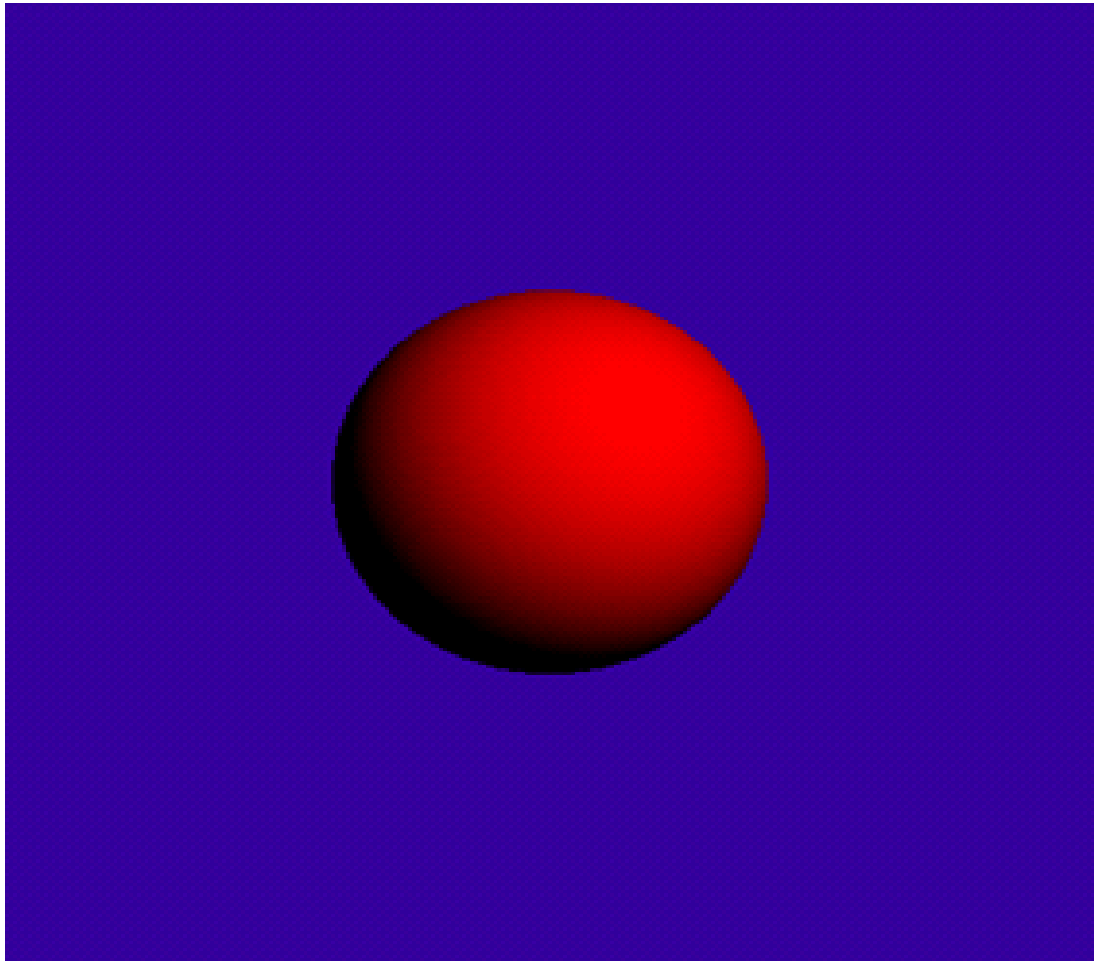


Distributed Source



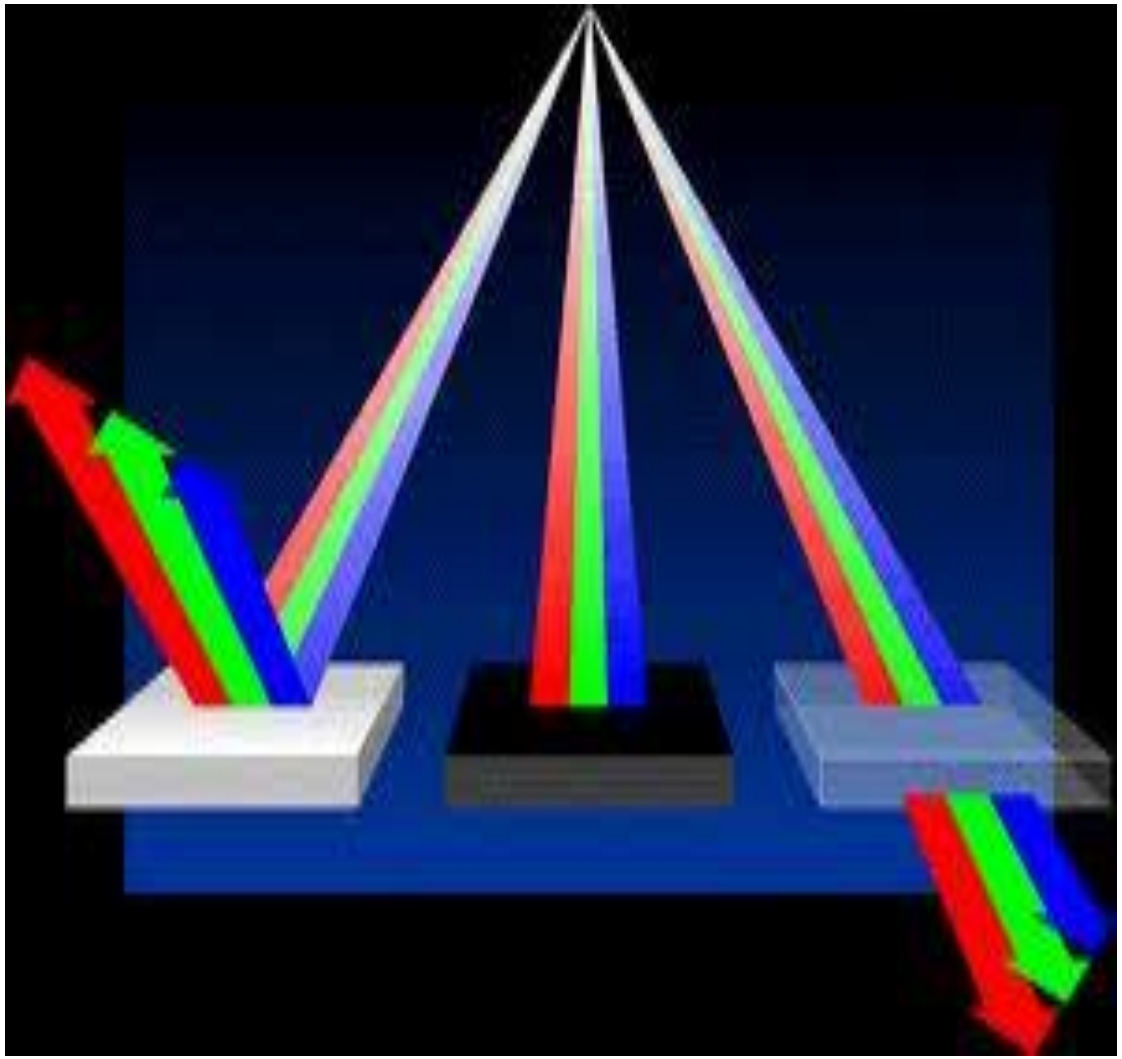
3D Object without light and with light





2. Surface - Opaque, dull & transparent surfaces

- When light falls on a surface part of it is reflected and part of it is absorbed.
- Now the surface structure decides the amount of reflection and absorption of light.
- The position of the surface and positions of all the nearby surfaces also determine the lightning effect.



3. Observer

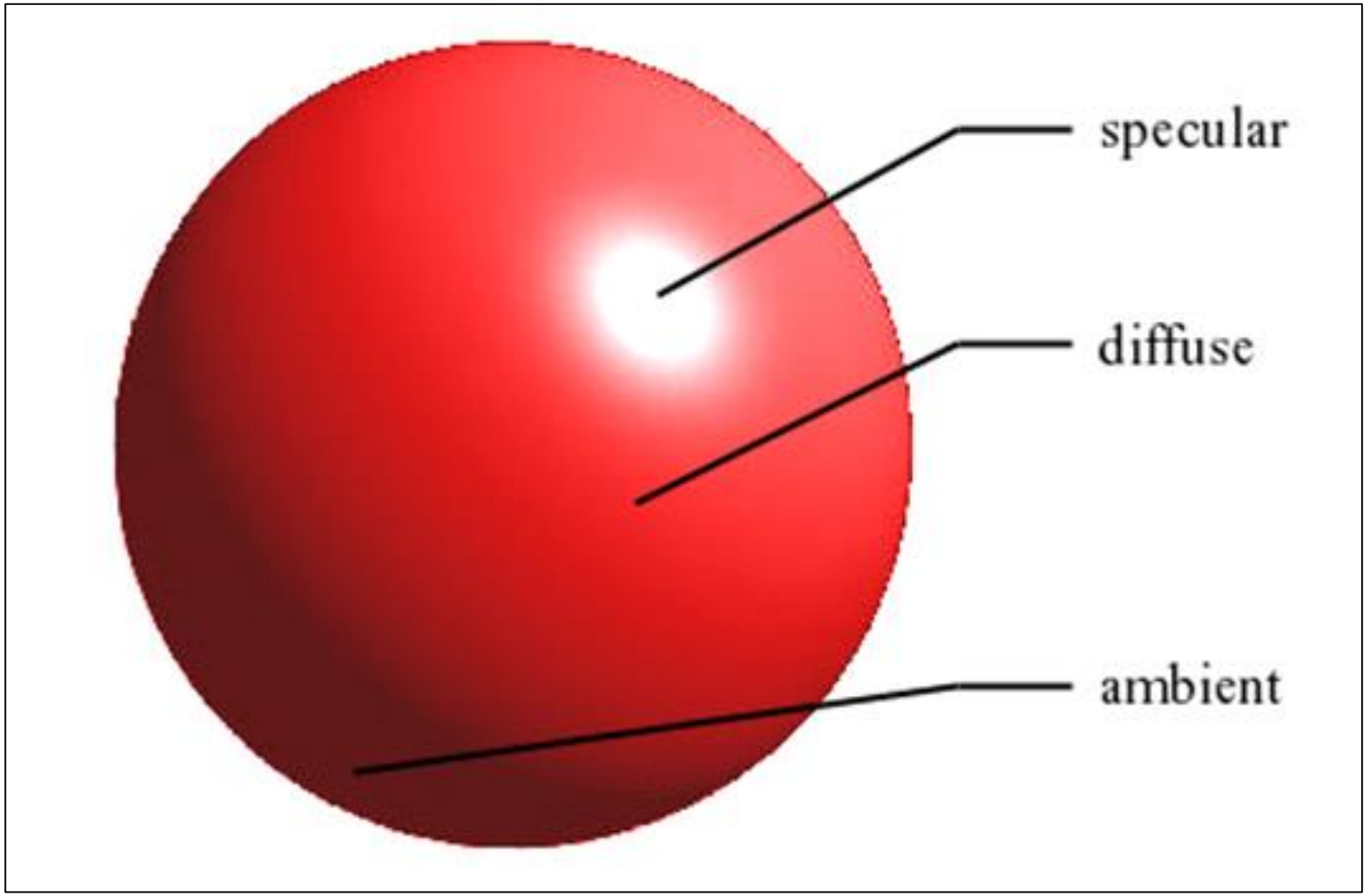
- The observer’s position and sensor spectrum sensitivities also affect the lightning effect.

Basic illumination Models

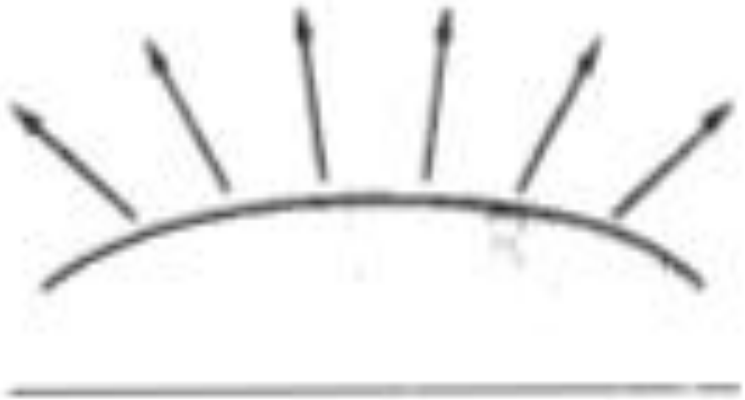
- ***Ambient*** – surface exposed to **indirect light** reflected from nearby objects.
- ***Diffuse*** – reflection from **incident** light with equal intensity in all directions. Depends on surface properties.

  Diffuse reflection occurs on the surfaces which are rough or grainy. In this reflection the brightness of a point depends upon the angle made by the light source and the surface.
- ***Specular*** – Light sources create highlights or **bright spots.**

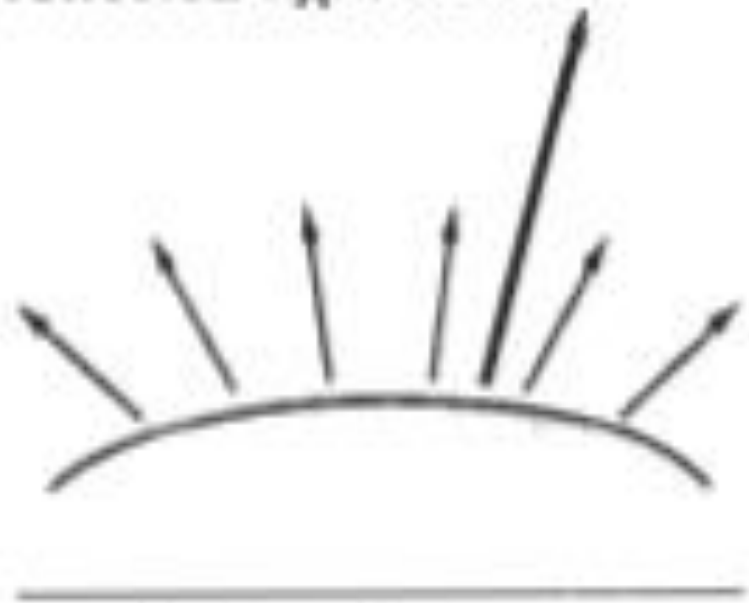
COMPONENTS OF REFLECTIONS



Surfaces that are rough, or grainy, tend to scatter the reflected light in all directions. This scattered light is called **diffuse reflection**.



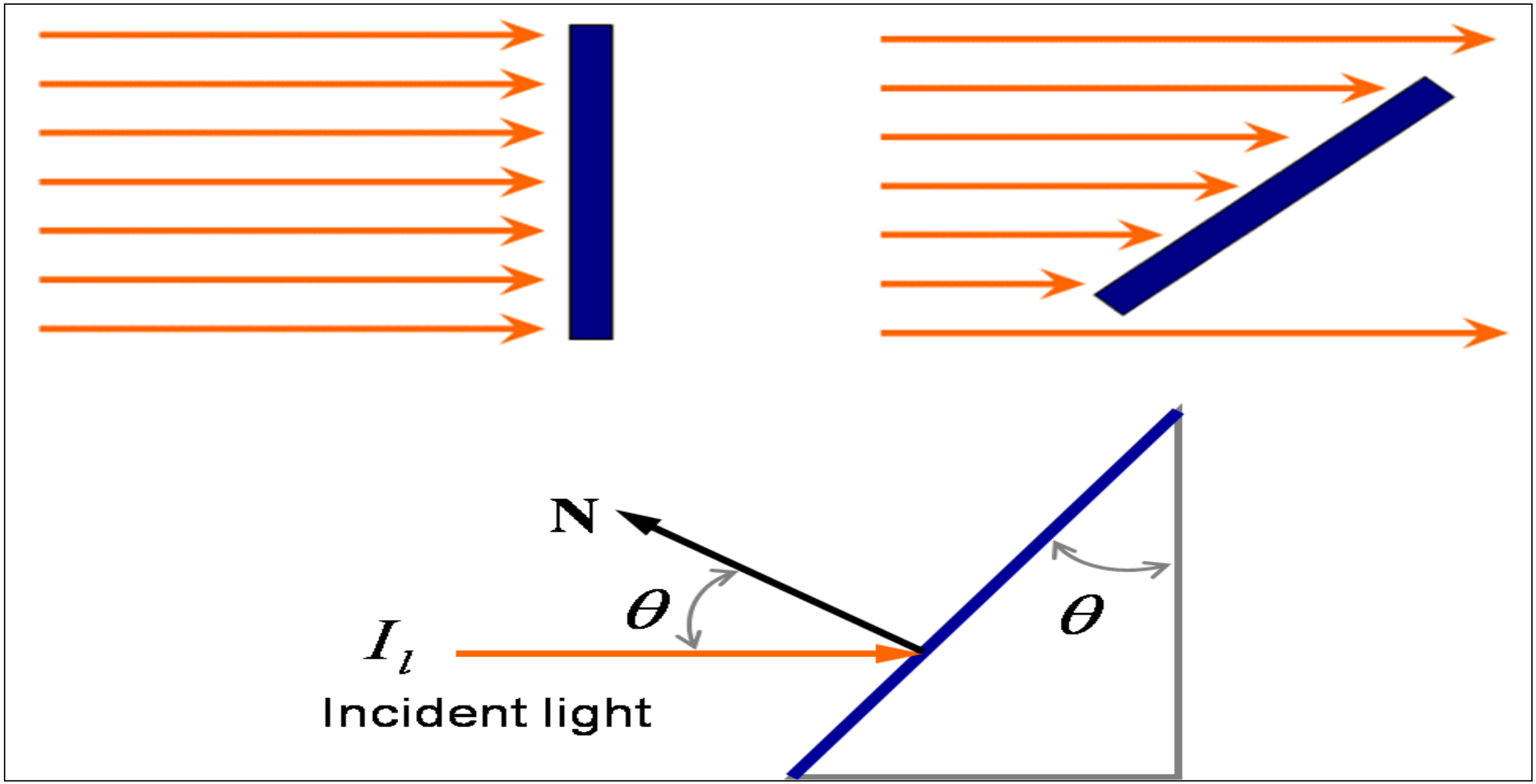
Diffuse reflections from a surface.

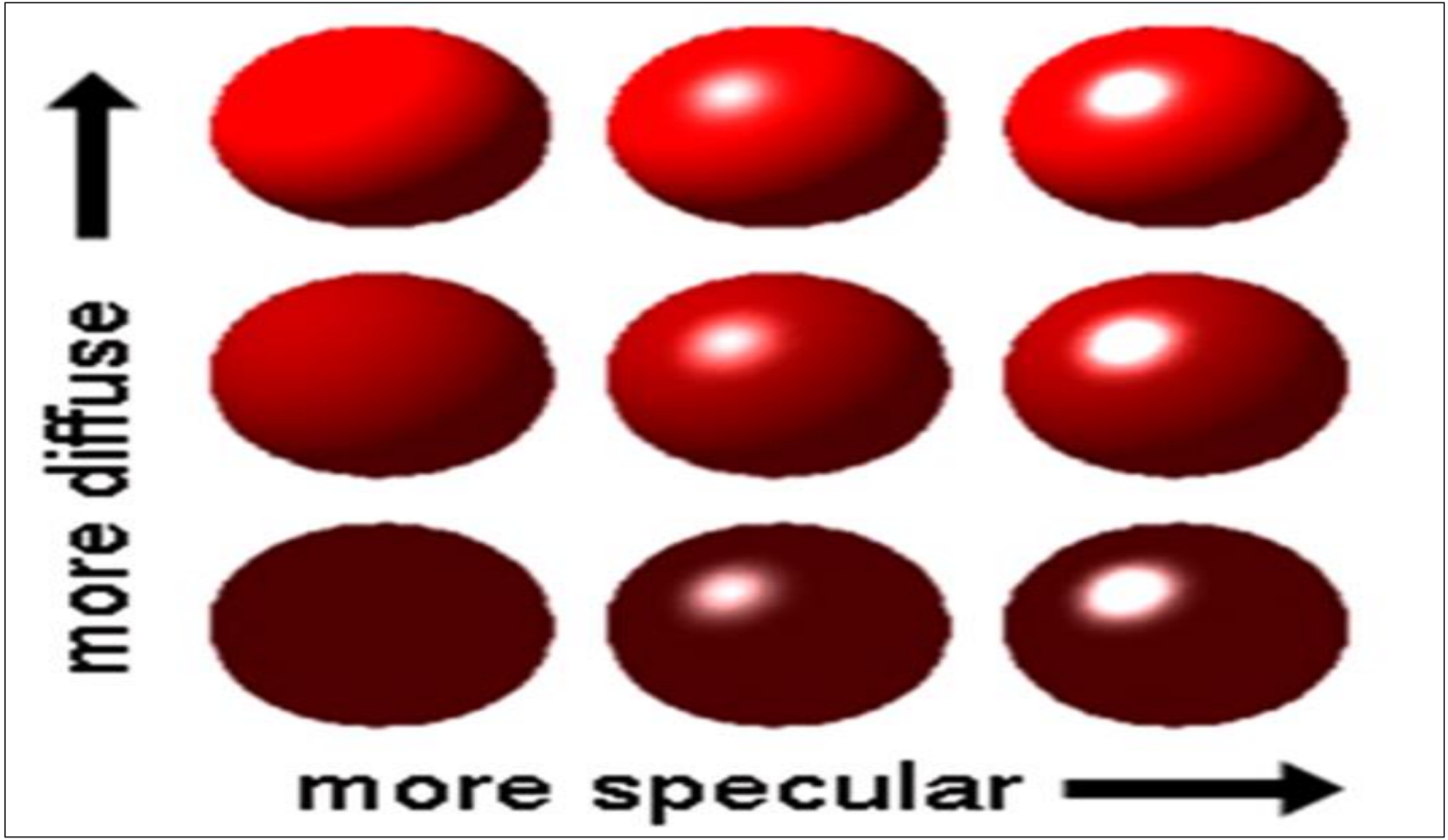


Specular reflection superimposed on diffuse reflection vectors.

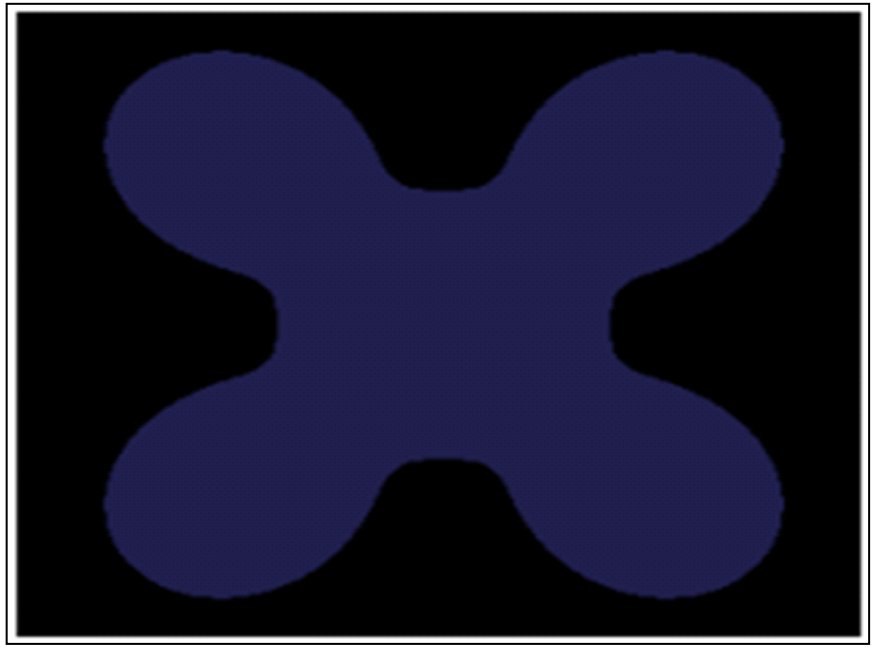
In addition to diffuse reflection, light sources create highlights, or bright spots, called **specular reflection**. This highlighting effect is more pronounced on shiny surfaces than on dull surfaces.

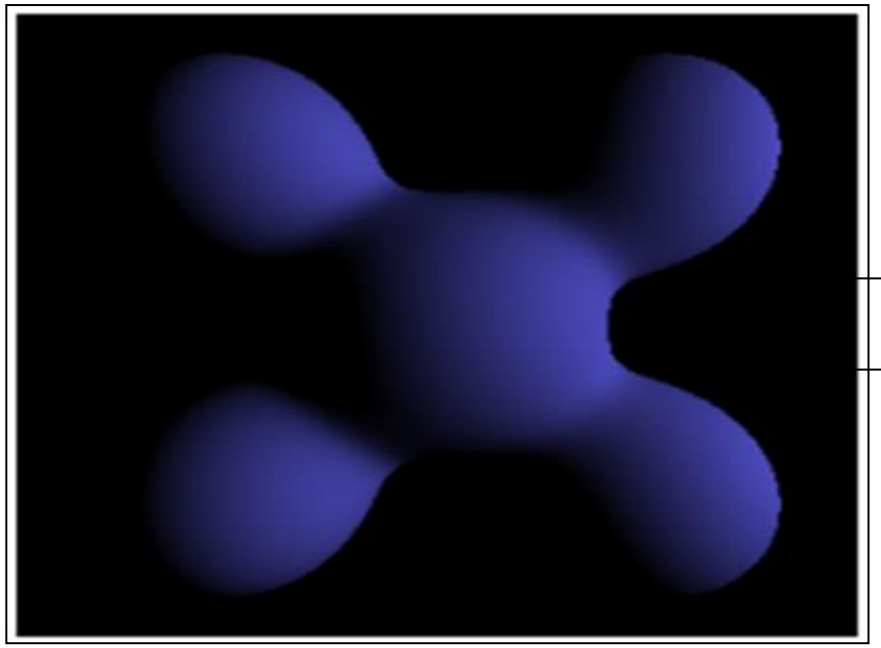
- The amount of incident light depends on the orientation of the surface relative to the light source direction.




more diffuse
more specular

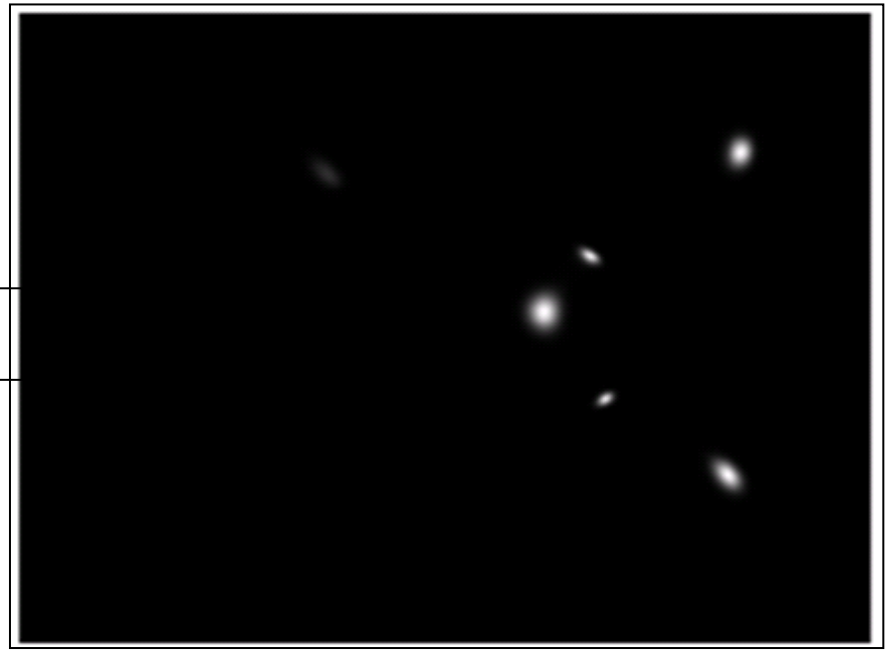
ambient

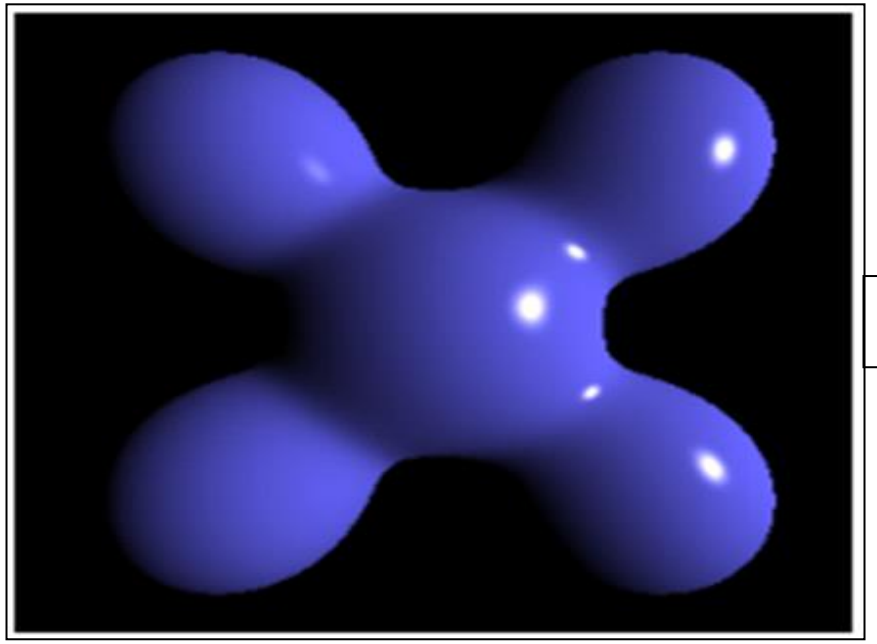




diffuse

specular





final

Lambert’s cosine Law

- We assume that the diffuse reflections from the surface are scattered with equal intensity in all directions, independent of the viewing directions.
- Such surfaces are sometimes referred to as ideal diffuse reflectors. They are also called Lambertian reflectors, since radiated light energy from any point on the surface is governed by Lambert’s cosine law.
- **Lambert’s cosine Law**: Radiant energy from any small surface, in any direction say θ, relative to the surface normal is proportional to $\cos \theta$

Reflection parameters

If the ambient light of the scene is I_a and k_a is a surfce ambient reflection parameter, the contribution to surface reflection at any point is $I_{\text{ambient}} = k_a I_a$.

For a light source I_l and surface reflection parameter k_d (independent of ambient relection parameter k_a), the diffused reflection is $I_{l,\text{diffuse}} = k_d I_l \cos\theta$.

Specular Reflection

- **Phong Model** is an empirical model for Specular Reflection which provides us with the formula for calculation the reflected intensity I_{spec}.

$$I_{spec} = W(\Theta) I_l \cos^n(\phi) = K_s I_l \cos^n(\phi)$$

Where,

L: direction of light source

N: normal to the surface

R: direction of reflected ray

V: direction of observer

Θ: Angle between L and R

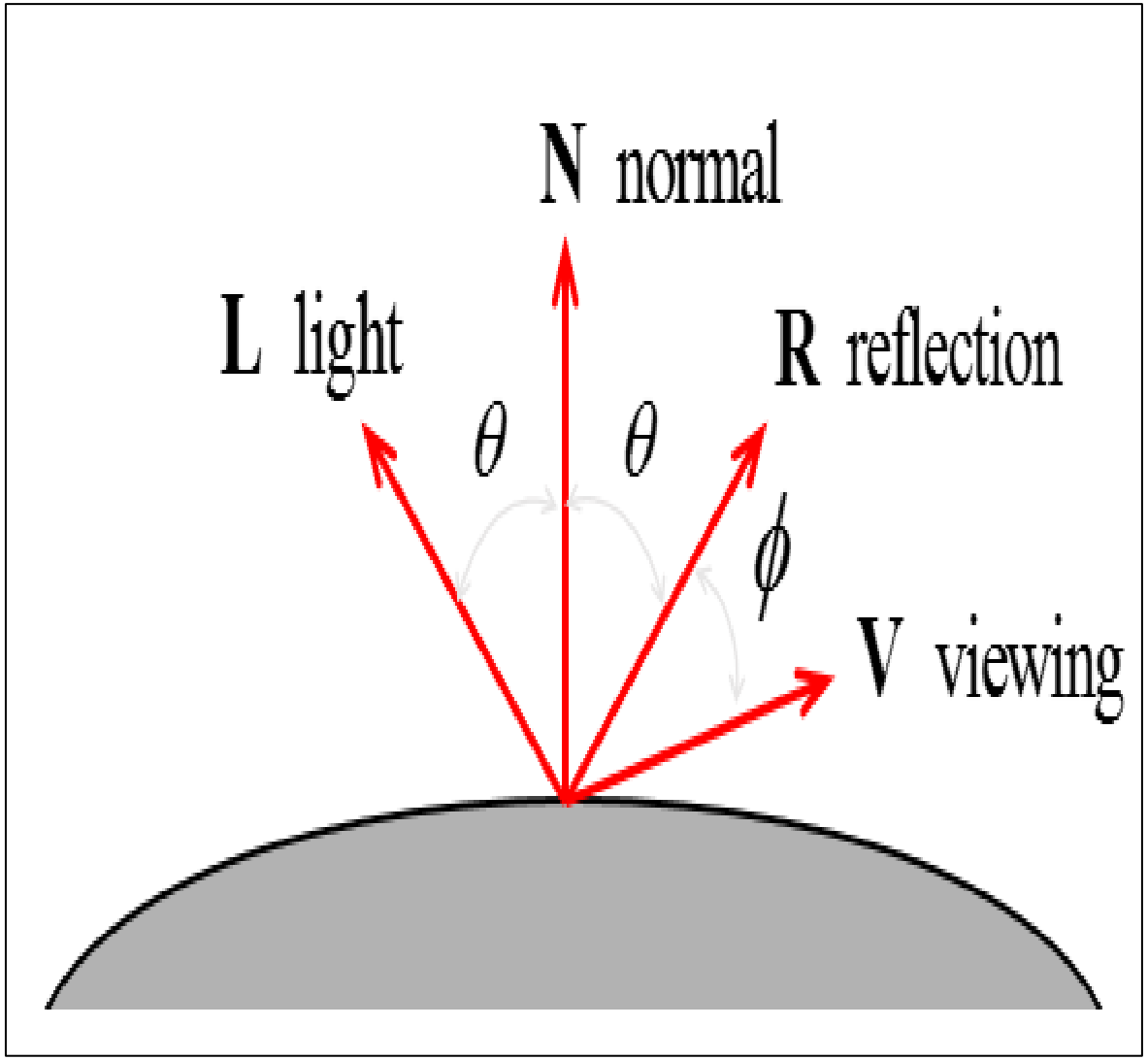
ϕ: Angle between R and V

SPECULAR REFLECTION

- Models shiny and glossy surfaces (like metal,plastic, etc..) with **highlights**
- Reflectance intensity changes with reflected angle
- An ideal specular surface (mirror) reflects light exclusively in one direction: **R**
- Glossy objects are not ideal mirrors and reflect in the immediate vicinity of *R*

Specular Reflection

- In this figure, we use R to represent the unit vector in the direction of ideal specular reflection;
- L to represent the unit vector directed toward the point light source; and V as the unit vector pointing to the viewer from the surface position.
- Angle Θ is the viewing angle relative to the specular-reflection direction R.
- For an ideal reflector (perfect mirror), incident light is reflected only in the specular-reflection direction. In this case, we would only see reflected light when vectors V and R coincide (Θ = 0).

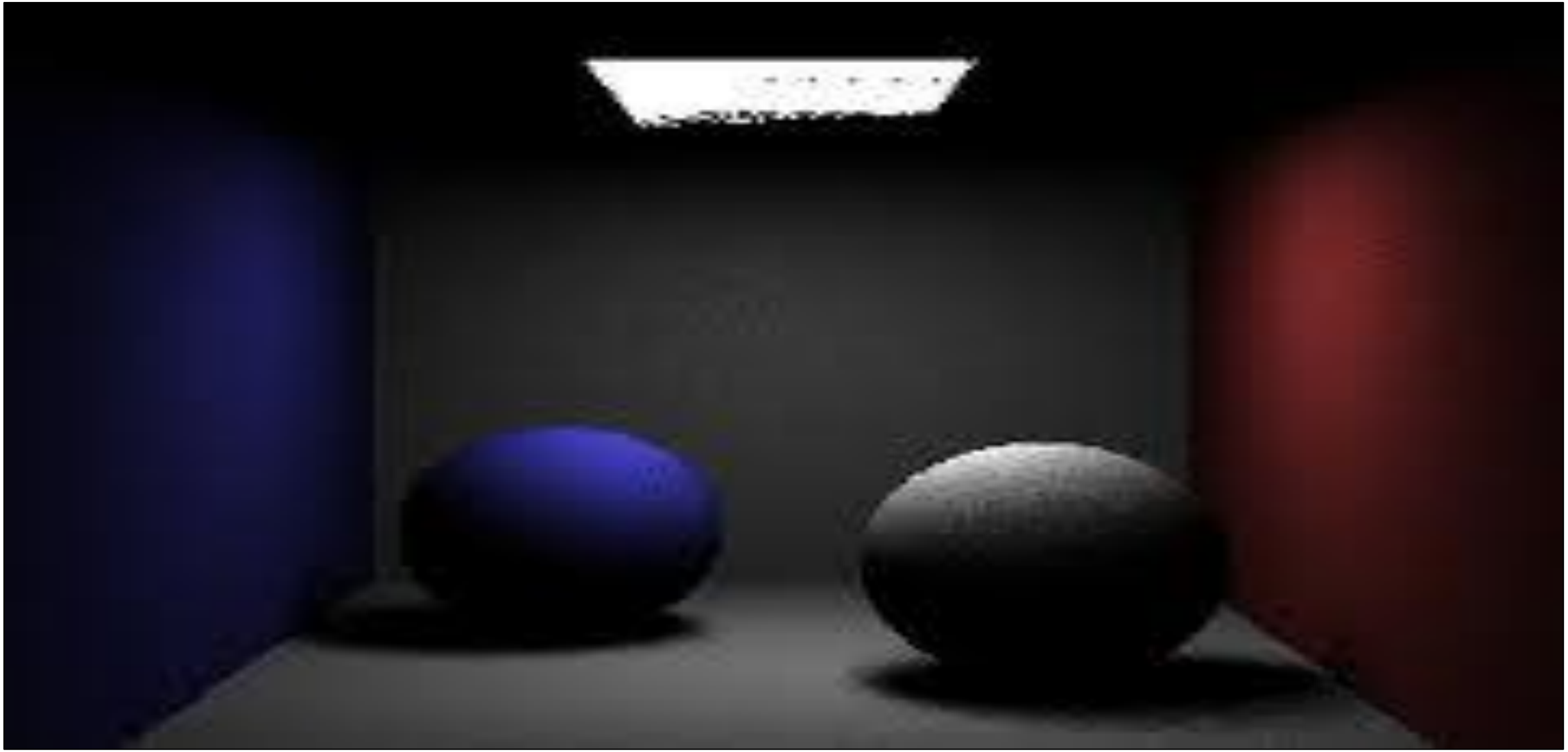


PHONG MODEL - **PHONG BUI TOUNG**

- A model for approximating the non ideal reflection is the Phong specular-reflection model where V is the unit vector in the direction of the viewer and R is the mirror reflection direction.
- This is an empirical model, which is not based on physics, but physical observation. Phong observed that for very shiny surfaces the specular highlight was small and the intensity fell off rapidly, while for duller surfaces it was larger and fell off more slowly. He decided to let the reflected intensity be a function of $(\cos a)^n$ with n >= 200 for a shiny surface and n small for a dull surface. For a perfect reflector n equals infinity, and for a piece of cardboard n equals 0 or 1. In the diagram below we can see how the function $(\cos a)^n$ behaves for different values of n.

$$I_{spec} = W(\Theta) I_l \cos^n(\phi) = K_s I_l \cos^n(\phi)$$

Combined specular and diffuse reflections



Warn Model

- The Warn model provides a method for simulating studio lighting effects by controlling light intensity in different directions.
- Light sources are modeled as points on a reflecting surface, using the Phong model for the surface points.
- Then the intensity in different directions is controlled by selecting values for the Phong exponent.
- In addition, light controls and spotlighting, used by studio photographers can be simulated in the Warn model.
- Flaps are used to control the amount of light emitted by a source in various directions.

Difference between ambient, diffuse, and specular reflection.

Ambient

- Approximates the effect of inter-reflections
- Sourceless – constant over entire surface
- Does not depend on surface normal
- Does not vary based on viewpoint

Diffuse

- Models rough surfaces (e.g. paper or drywall) – where light scatters equally in all directions
- Has a point or directional source
- Depends on surface normal – brightest where the surface is oriented toward the light, and falls off to zero at 90°
- Does not vary based on viewpoint

Specular

- Models highlights from smooth, shiny surfaces (e.g. opaque plastic)
- Has a point or directional source
- Depends on surface normal
- Depends on viewpoint

he Phong model puts these three terms together:

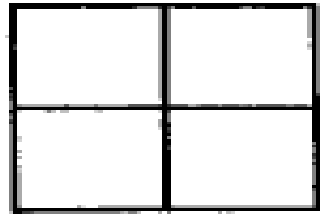
$$I_a k_a + \sum^{lights} \left[I_i k_{diff} (N \cdot L) + I_i k_{spec} (R \cdot V)^n \right]$$

HALFTONING PATTERNS AND DITHERING TECHNIQUES

https://www.youtube.com/watch?v=S3JzlOr1eH4
https://www.youtube.com/watch?v=nK6pYXRHTVU
https://www.youtube.com/watch?v=cLnuoQ6pIKk

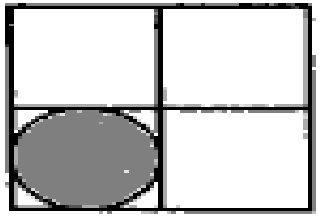
Halftones

- The process of generating a binary pattern of black and white dots from an image is termed **halftoning**.
- In traditional newspaper and magazine production, this process is carried out photographically by projection of a transparency through a 'halftone screen' onto film.
- The pictures produced by halftoning process are called **halftones**. The screen is a glass plate with a grid etched into it.
- Different screens can be used to control the size and shape of the dots in the halftoned image.
- In computer graphics, halftone reproductions are approximated using rectangular pixel regions, say 2x 2 pixels or 3x 3 pixels.
- These regions arc called halftone patterns or pixel patterns. figure (e) shown the halftone patterns to create number of intensity levels.



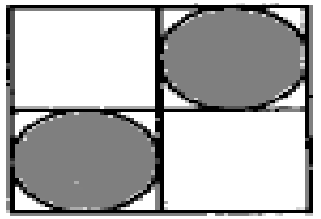
0

$0 \leq I < 0.2$



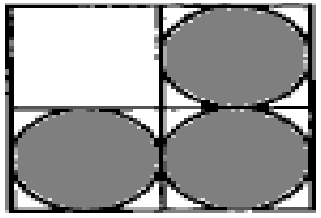
1

$0.2 \leq I < 0.4$



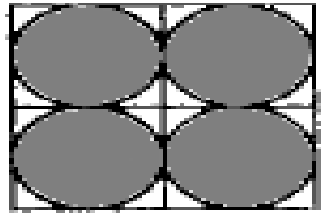
2

$0.4 \leq I < 0.6$



3

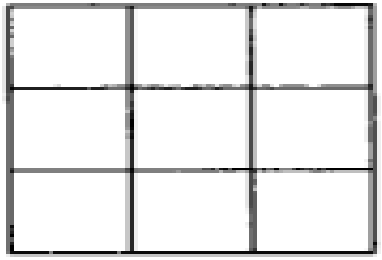
$0.6 \leq I < 0.8$



4

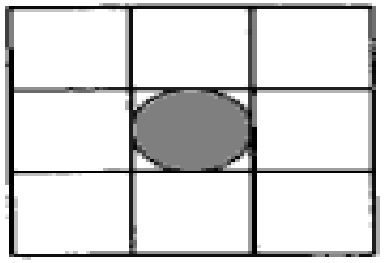
$0.8 \leq I < 1.0$

2*2 Pixel patterns for creating five intensity levels



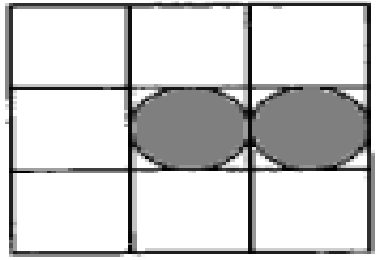
0

$0 \leq I < 0.1$



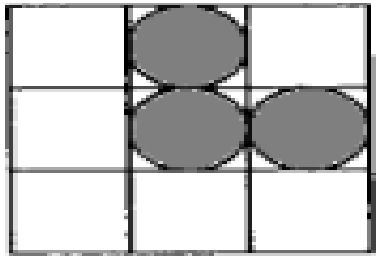
1

$0.1 \leq I < 0.2$



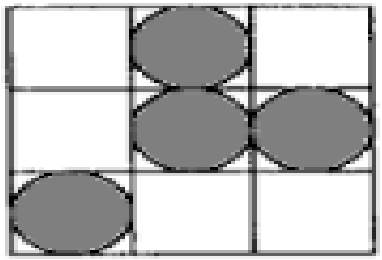
2

$0.2 \leq I < 0.3$



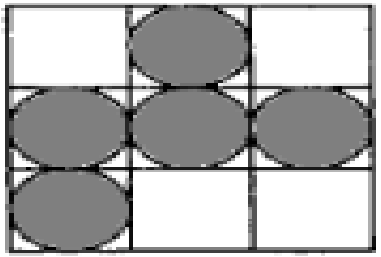
3

$0.3 \leq I < 0.4$



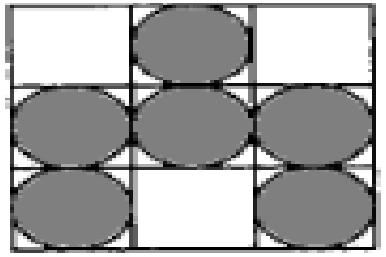
4

$0.4 \leq I < 0.5$



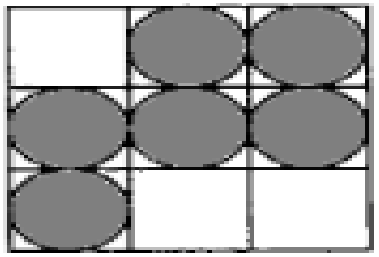
5

$0.5 \leq I < 0.6$



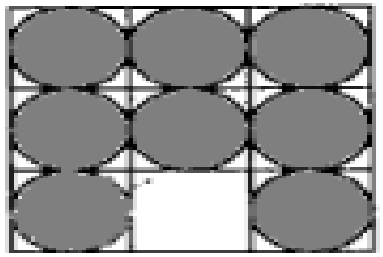
6

$0.6 \leq I < 0.7$



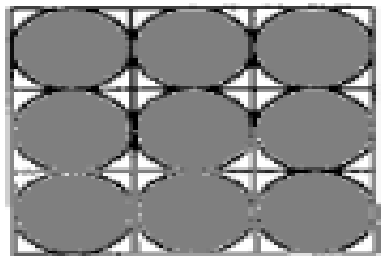
7

$0.7 \leq I < 0.8$



8

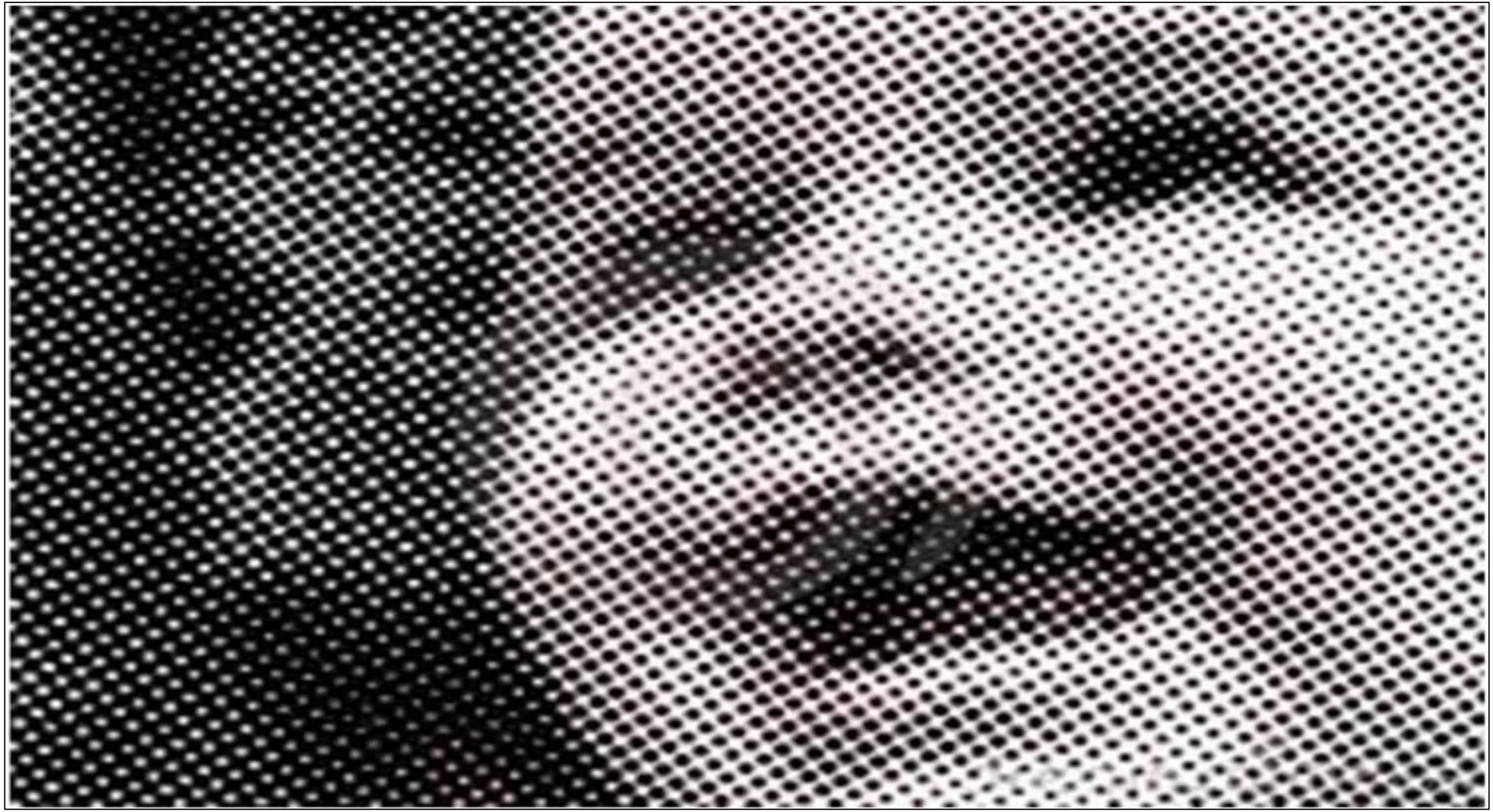
$0.8 \leq I < 0.9$



9

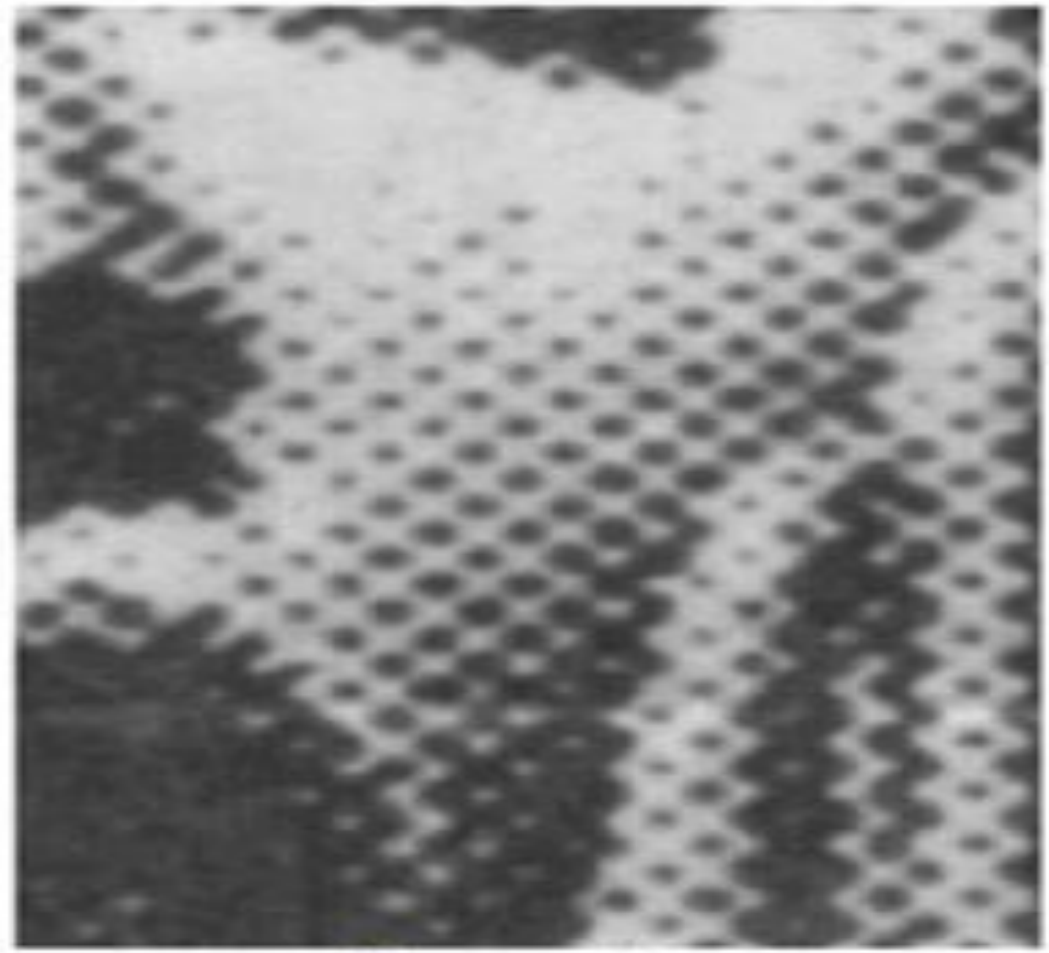
$0.9 \leq I < 1.0$

3*3 Pixel patterns for creating ten intensity levels



HALFTONE IN A NEWSPAPER

Original



Zoom-in

ORIGINAL

HALFTONE

COLOR HALFTONES

COLOR HALFTONES

Dithering

- Random values added to pixel intensities to break up contours are often referred to as **dither noise**.
- **Dithering** is used to create additional colors and shades from an existing palette by interspersing pixels of different colors.
- It is possible to improve the quality of a quantized image by distributing the quantized error.

Dithering

- Each pixel produces a *quantization* error
- The quality of the result may be improved by adjusting the threshold locally, so that adjacent pixels in small areas are quantized with different thresholds.
- This reduces the **average** local quantization error. Matrices of these threshold are called **dither** matrices.

Dithering Techniques

- Dithering refers to techniques for **approximating halftones without reducing resolution**, as pixel grid patterns do.
- The term dithering is also applied to halftone approximation methods using pixel grids, and sometimes it is used to refer to colour halftone approximations only.
- Dithering is the attempt by a computer program to approximate a color from a mixture of other colors when the required color is not available.
- For example, dithering occurs when a color is specified for a Web page that a browser on a particular operating system can't support.
- The browser will then attempt to replace the requested color with an approximation composed of two or more other colors it can produce.
- The result may or may not be acceptable to the graphic designer. It may also appear somewhat grainy since it's composed of different pixel intensities rather than a single intensity over the colored space.

Dithering Techniques

- Random values added to pixel intensities to break up contours are often referred as **dither noise**.
- Numbers of methods are used to generate intensity variations. Ordered dither methods generate intensity variations with a one-to-one mapping of points in a scene to the display pixels.
- To obtain n2 intensity levels, it is necessary to set up an n x n dither matrix whose elements are discrete positive integers in the range of 0 to n-1.
- The matrix elements for D2 and D3 are in the same order as the pixel mask for setting up 2 x 2 and 3 x 3 pixel grids respectively.
- For bi-level system we have to determine display intensity values by comparing input intensities to the matrix elements.

- Each input intensity is first scaled to the range $\mathbf{0 \leq I \leq n2}$. If the intensity I is to be applied to screen position (x, y), we have to calculate row and column numbers for the either matrix as

  $$\mathbf{i = (x \bmod n) + 1 \text{ and } j = (y \bmod n) + 1}$$

- If I > D1(i, j) the pixel at position (x, y) is turned on; otherwise the pixel is not turned on. Typically, the number of intensity levels is taken to be a multiple of 2.
- High order dither matrices can be obtained from lower order matrices with the recurrence relation.

Dithering Techniques

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

and it is possible to generate nine intensity levels with

$$D_3 = \begin{bmatrix} 7 & 2 & 6 \\ 4 & 0 & 1 \\ 3 & 8 & 5 \end{bmatrix}$$

$$i = (x \bmod n) + 1, \qquad j = (y \bmod n) + 1$$

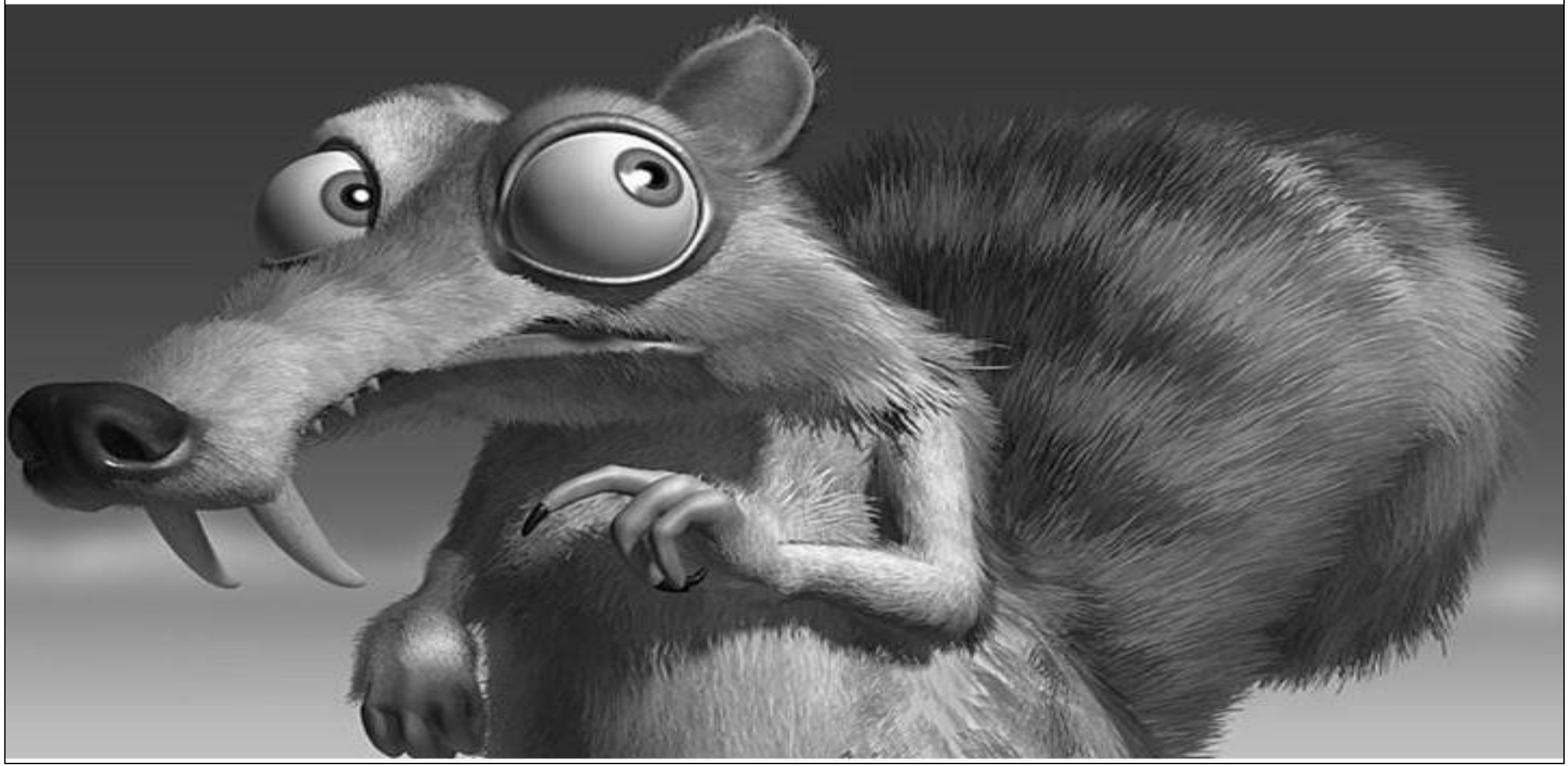
$$D_n = \begin{bmatrix} 4\,D_{n/2} + D_2(1,1)\,u_{n/2} & 4\,D_{n/2} + D_2(1,2)\,u_{n/2} \\ 4\,D_{n/2} + D_2(2,1)\,u_{n/2} & 4\,D_{n/2} + D_2(2,2)\,u_{n/2} \end{bmatrix}$$

assuming $n \geq 4$. Parameter $u_{n/2}$ is the unity matrix.

$$D_n = \begin{bmatrix} 4D_{n/2} + D_2(1,1)U_{n/2} & 4D_{n/2} + D_2(1,2)U_{n/2} \\ 4D_{n/2} + D_2(2,1)U_{n/2} & 4D_{n/2} + D_2(2,2)U_{n/2} \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 15 & 7 & 13 & 5 \\ 3 & 11 & 1 & 9 \\ 12 & 4 & 14 & 6 \\ 0 & 8 & 2 & 10 \end{bmatrix}$$

Original



Halftone

Dithering by applying Error Diffusion