Unit 3 Hidden Surface & Shading

Differentiate between Object space and Image space method

Object Space	Image Space
1. Image space is object based. It concentrates on geometrical relation among objects in the scene.	1. It is a pixel-based method. It is concerned with the final image, what is visible within each raster pixel.
2. Here surface visibility is determined.	2. Here line visibility or point visibility is determined.
3. It is performed at the precision with which each object is defined, No resolution is considered.	3. It is performed using the resolution of the display device.
4. Calculations are not based on the resolution of the display so change of object can be easily adjusted.	4. Calculations are resolution base, so the change is difficult to adjust.
5. These were developed for vector graphics system.	5. These are developed for raster devices.
6. Object-based algorithms operate on continuous object data.	6. These operate on object data.
7. Vector display used for object method has large address space.	7. Raster systems used for image space methods have limited address space.
8. Object precision is used for application where speed is required.	8. There are suitable for application where accuracy is required.
9. It requires a lot of calculations if the image is to enlarge.	9. Image can be enlarged without losing accuracy.

10. If the number of objects in the scene increases, computation time also increases.	10. In this method complexity increase with the complexity of visible parts.

Coherence

It is used to take advantage of the constant value of the surface of the scene. It is based on how much regularity exists in the scene. When we moved from one polygon of one object to another polygon of same object color and shearing will remain unchanged.

Types of Coherence

1. Edge coherence
2. Object coherence
3. Face coherence
4. Area coherence
5. Depth coherence
6. Scan line coherence
7. Frame coherence
8. Implied edge coherence

1. Edge coherence: The visibility of edge changes when it crosses another edge or it also penetrates a visible edge.
2. Object coherence: Each object is considered separate from others. In object, coherence comparison is done using an object instead of edge or vertex. If A object is farther from object B, then there is no need to compare edges and faces.
3. Face coherence: In this faces or polygons which are generally small compared with the size of the image.
4. Area coherence: It is used to group of pixels cover by same visible face.
5. Depth coherence: Location of various polygons has separated a basis of depth. Depth of surface at one point is calculated, the depth of points on rest of the surface can often be determined by a simple difference equation.
6. Scan line coherence: The object is scanned using one scan line then using the second scan line. The intercept of the first line.
7. Frame coherence: It is used for animated objects. It is used when there is little change in image from one frame to another.
8. Implied edge coherence: If a face penetrates in another, line of intersection can be determined from two points of intersection.

Algorithms used for hidden line surface detection

1. Back Face Removal Algorithm
2. Z-Buffer Algorithm
3. Painter Algorithm
4. Scan Line Algorithm
5. Subdivision Algorithm
6. Floating horizon Algorithm

Back Face Removal Algorithm

It is used to plot only surfaces which will face the camera. The objects on the back side are not visible. This method will remove 50% of polygons from the scene if the parallel projection is used. If the perspective projection is used then more than 50% of the invisible area will be removed. The object is nearer to the center of projection, number of polygons from the back will be removed.

It applies to individual objects. It does not consider the interaction between various objects. Many polygons are obscured by front faces, although they are closer to the viewer, so for removing such faces back face removal algorithm is used.

When the projection is taken, any projector ray from the center of projection through viewing screen to object pieces object at two points, one is visible front surfaces, and another is not visible back surface.

This algorithm acts a preprocessing step for another algorithm. The back face algorithm can be represented geometrically. Each polygon has several vertices. All vertices are numbered in clockwise. The normal M_1 is generated a cross product of any two successive edge vectors. M_1represent vector perpendicular to face and point outward from polyhedron surface

$$N_1=(v_2-v_1)(v_3-v_2)$$

If $N_1.P \geq 0$ visible

$N_1.P<0$ invisible

Advantage

1. It is a simple and straight forward method.
2. It reduces the size of databases, because no need of store all surfaces in the database, only the visible surface is stored.

Back Face Removed Algorithm

Repeat for all polygons in the scene.

1. Do numbering of all polygons in clockwise direction i.e.
   $v_1\ v_2\ v_3 \ldots . v_z$
2. Calculate normal vector i.e. N_1
   $N_1=(v_2-v_1)*(v_3-v_2)$
3. Consider projector P, it is projection from any vertex
   Calculate dot product
   Dot=N.P
4. Test and plot whether the surface is visible or not.
   If Dot ≥ 0 then
   surface is visible
   else
   Not visible

Z-Buffer Algorithm

It is also called a **Depth Buffer Algorithm**. Depth buffer algorithm is simplest image space algorithm. For each pixel on the display screen, we keep a record of the depth of an object within the pixel that lies closest to the observer. In addition to depth, we also record the intensity that should be displayed to show the object. Depth buffer is an extension of the frame buffer. Depth buffer algorithm requires 2 arrays, intensity and depth each of which is indexed by pixel coordinates (x, y).

Algorithm

For all pixels on the screen, set depth [x, y] to 1.0 and intensity [x, y] to a background value.

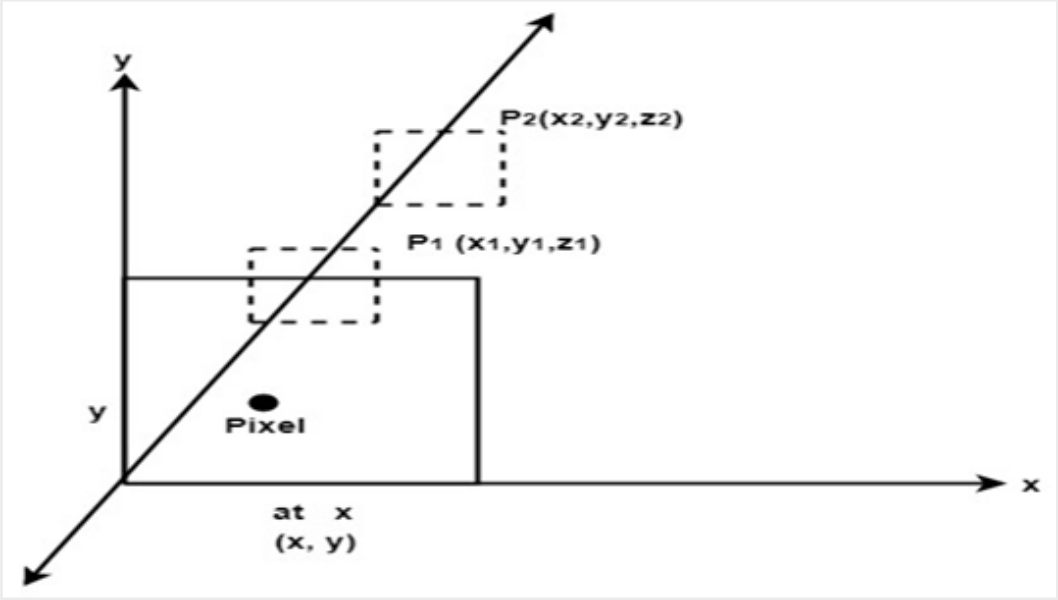
For each polygon in the scene, find all pixels (x, y) that lie within the boundaries of a polygon when projected onto the screen. For each of these pixels:

(a) Calculate the depth z of the polygon at (x, y)

(b) If z < depth [x, y], this polygon is closer to the observer than others already recorded for this pixel. In this case, set depth [x, y] to z and intensity [x, y] to a value corresponding to polygon's shading. If instead z > depth [x, y], the polygon already recorded at (x, y) lies closer to the observer than does this new polygon, and no action is taken.

3. After all, polygons have been processed; the intensity array will contain the solution.

4. The depth buffer algorithm illustrates several features common to all hidden surface algorithms.



5. First, it requires a representation of all opaque surface in scene polygon in this case.
6. These polygons may be faces of polyhedral recorded in the model of scene or may simply represent thin opaque 'sheets' in the scene.
7. The IInd important feature of the algorithm is its use of a screen coordinate system. Before step 1, all polygons in the scene are transformed into a screen coordinate system using matrix multiplication.

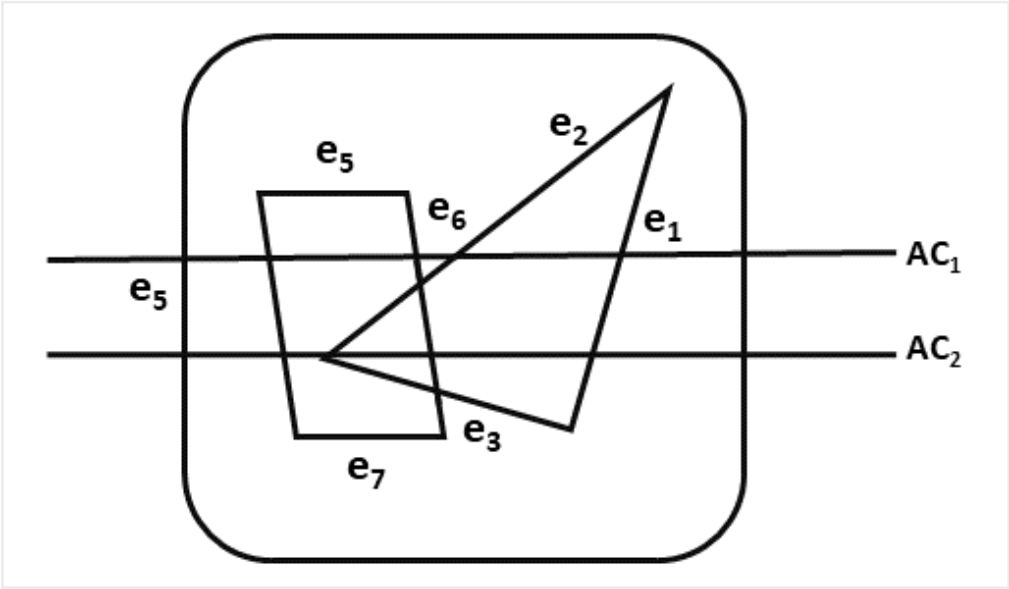
Limitations of Depth Buffer

1. The depth buffer Algorithm is not always practical because of the enormous size of depth and intensity arrays.
2. Generating an image with a raster of 500 x 500 pixels requires 2, 50,000 storage locations for each array.
3. Even though the frame buffer may provide memory for intensity array, the depth array remains large.
4. 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.
5. For example, the original 500 x 500 faster can be divided into 100 rasters each 50 x 50 pixels.
6. Processing each small raster requires array of only 2500 elements, but execution time grows because each polygon is processed many times.
7. 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.

Scan Line Algorithm

It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. So accurate bookkeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. Active Edge List (AEL) contain edges a given scan line intersects during its sweep. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.

Following figures shown edges and active edge list. The active edge list for scan line AC_1contain e_1,e_2,e_5,e_6 edges. The active edge list for scan line AC_2contain e_5,e_6,e_1.



Scan line can deal with multiple surfaces. As each scan line is processed, this line will intersect many surfaces. The intersecting line will determine which surface is visible. Depth calculation for each surface is done. The surface rear to view plane is defined. When the visibility of a surface is determined, then intensity value is entered into refresh buffer.

Algorithm

Step1: Start algorithm

Step2: Initialize the desired data structure

1. Create a polygon table having color, edge pointers, coefficients
2. Establish edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.
3. Create Active edge list. This will be sorted in increasing order of x.

4. Create a flag F. It will have two values either on or off.

Step3: Perform the following steps for all scan lines

1. Enter values in Active edge list (AEL) in sorted order using y as value
2. Scan until the flag, i.e. F is on using a background color
3. When one polygon flag is on, and this is for surface S_1enter color intensity as I_1into refresh buffer
4. When two or image surface flag are on, sort the surfaces according to depth and use intensity value S_n for the nth surface. This surface will have least z depth value
5. Use the concept of coherence for remaining planes.

Step4: Stop Algorithm

Shading And Illumination

Shading, Illumination Models, and Surface Rendering

Shading and illumination models are essential techniques used in computer graphics to simulate the interaction of light with surfaces, creating realistic and visually appealing images.

Basic Concepts

- **Shading:** The process of assigning colors to pixels on a screen to represent the intensity of light reflected from a surface.
- **Illumination Model:** A mathematical model that describes how light interacts with a surface, considering factors like the surface material, light source properties, and viewing angle.

Diffuse Reflection

- **Lambertian Reflection:** A model that assumes light is scattered equally in all directions from a surface.
- **Calculation:** The intensity of reflected light is proportional to the cosine of the angle between the surface normal and the light direction.
- **Visual Appearance:** Matte surfaces like chalk or unfinished wood exhibit diffuse reflection.

Curved Surfaces

To accurately render curved surfaces, we need to consider the variation of surface normals across the surface. Techniques like Gouraud shading and Phong shading are commonly used for this purpose.

Gouraud Shading

- **Interpolation of Colors:** Calculates the intensity at each vertex of a polygon and then linearly interpolates the color across the polygon's surface.
- **Advantages:** Efficient and produces smooth shading for most surfaces.
- **Limitations:** Can result in Mach bands, noticeable intensity discontinuities at polygon edges, especially for highly curved surfaces.

Phong Shading

- **Interpolation of Normals:** Interpolates the surface normal across the polygon's surface.
- **Per-Pixel Lighting Calculation:** Calculates the lighting equation at each pixel, using the interpolated normal to determine the intensity.
- **Advantages:** Produces more accurate and realistic shading, especially for specular highlights and curved surfaces.
- **Disadvantages:** More computationally expensive than Gouraud shading.

Additional Considerations

- **Ambient Light:** A constant level of light that illuminates all surfaces, simulating indirect lighting.
- **Specular Reflection:** The reflection of light in a concentrated direction, resulting in highlights on shiny surfaces.
- **Multiple Light Sources:** The effects of multiple light sources can be combined to create complex lighting scenarios.
- **Texture Mapping:** Applying images or patterns to surfaces to add detail and realism.
- **Bump Mapping and Normal Mapping:** Techniques to simulate fine-scale surface details without increasing the polygon count.

By understanding these concepts and techniques, you can create realistic and visually stunning 3D graphics, from simple objects to complex scenes.