APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS

CS 401 COMPUTER GRAPHICS

Module 5

Projections – Parallel And Perspective Projections – Vanishing Points.

Visible Surface Detection Methods– Back Face Removal- Z-Buffer Algorithm, A-Buffer Algorithm, Depth-Sorting Method, Scan Line Algorithm.

Three-Dimensional Viewing

In two-dimensional graphics applications, viewing operations transfer positions from the world-coordinate plane to pixel positions in the plane of the output device. Using the rectangular boundaries for the world-coordinate window and the device viewport, a two-dimensional package maps the world scene to device coordinates and clips the scene against the four boundaries of the viewport. For three-dimensional applications, the situation is a bit more involved, since we now have more choices as to how views are to be generated. First of all, we can view an object from any spatial position: from the front, from above, or from the back *or* we could generate a view of what we would see if we were standing in the middle of a group of objects or inside a single object, such as a building. Additionally, three-dimensional descriptions of objects must be projected onto the flat viewing surface of the output device. And the clipping boundaries now enclose a volume of space, whose shape depends on the type of projection we select.

3d-Viewing Pipeline

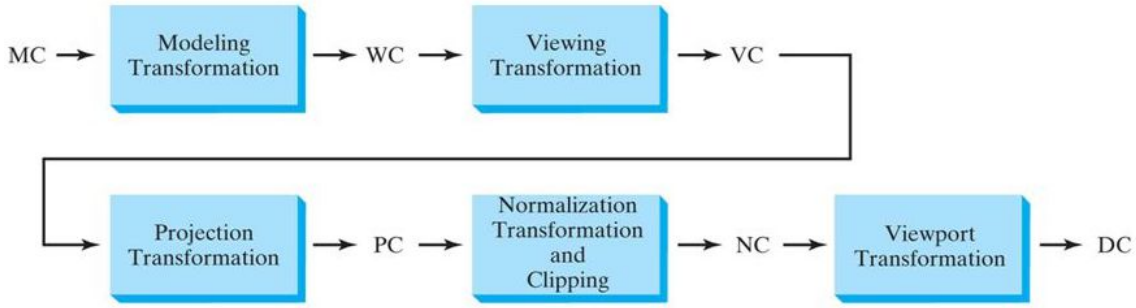
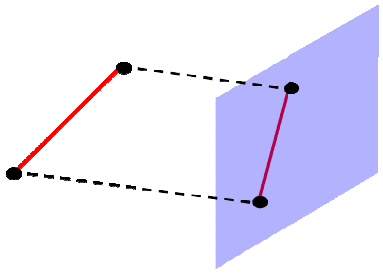


Figure shows the general processing steps for modeling and converting a world-coordinate description of a scene to device coordinates. Once the scene has been modeled, world-coordinate positions are converted to viewing coordinates. The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane, which we can think of in analogy with the camera film plane. Next, projection operations are performed to convert the viewing-coordinate description

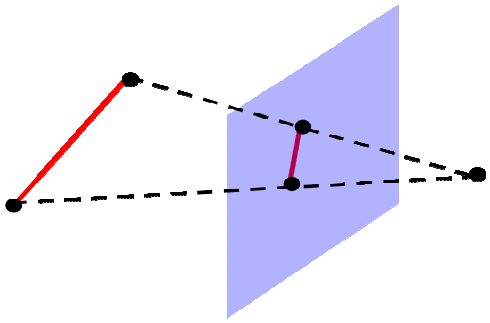
of the scene to coordinate positions on the projection plane, which will then be mapped to the output device. Objects outside the specified viewing limits are clipped from further consideration, and the remaining objects are processed through visible-surface identification and surface rendering procedures to produce the display within the device viewport.

Projections

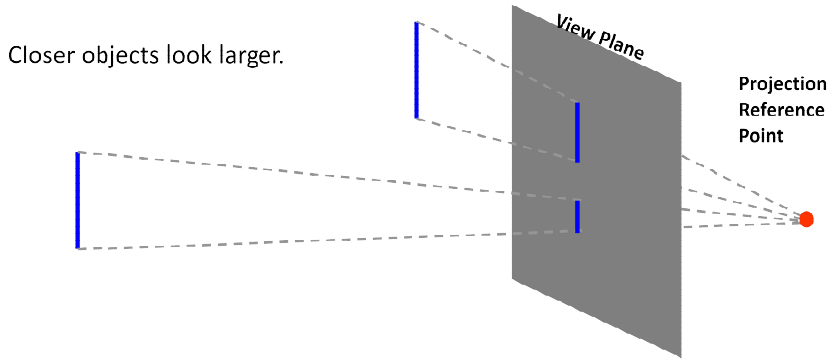
Once world-coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the three-dimensional objects onto the two dimensional view plane. There are two basic projection methods. In a parallel projection, coordinate positions are transformed to the vied plane along parallel lines.



For a perspective projection , object positions are transformed to the view plane along lines that converge to a point called the projection reference point (or center of projection).The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.

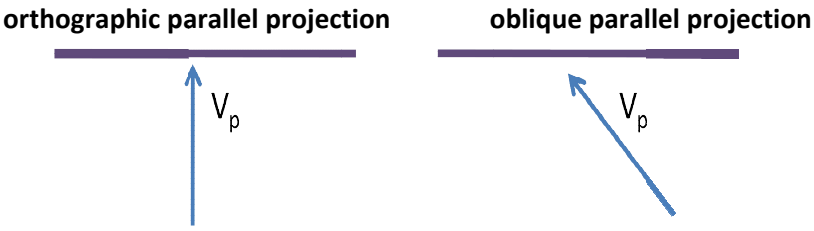


A parallel projection preserves relative proportions of objects, and this is the method used in drafting to produce scale drawings of three-dimensional objects. Accurate views of the various sides of an object are obtained with a parallel projection, but this does not give us a realistic representation of the appearance of a three-dimensional object. A perspective projection, on the other hand, produces realistic views but does not preserve relative proportions. Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.



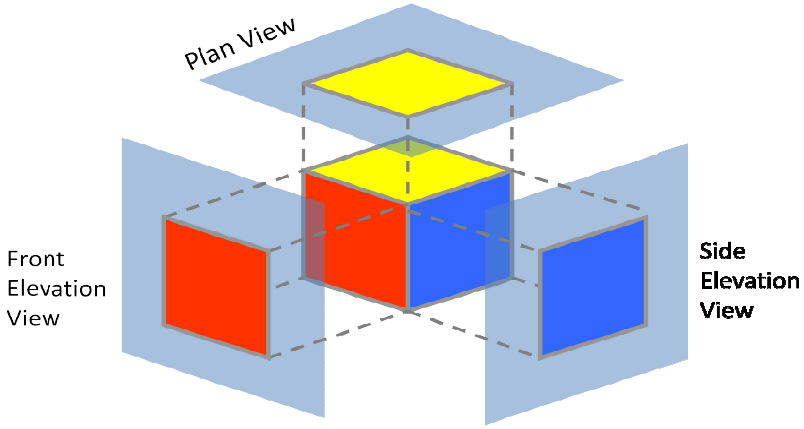
Parallel projections

We can specify a parallel projection with a projection vector, V_p that defines the dlrection for the projection lines. When the projection is perpendicular to the view plane, we have an orthographic parallel projection. Otherwise, we have an oblique parallel projection



Orthographic projections

Orthographic projections produce front, side and top views of an object.Front, side and rear projections known as elevations and top projection is known as plan view



Engineering and architectural drawings commonly employ these orthographic projections, because lengths and angles are accurately depicted and can be measured from the drawings. We can also form orthographic projections that dlsplay more than one face of an object. Such views are called axonometric orthographic projections. The most commonly used axonometric projection is the isometric projection. We generate an isometric projection by aligning the projection plane so that it intersects

each coordinate axis in which the object is defined (called principal axes***)*** at the same distance from the origin.

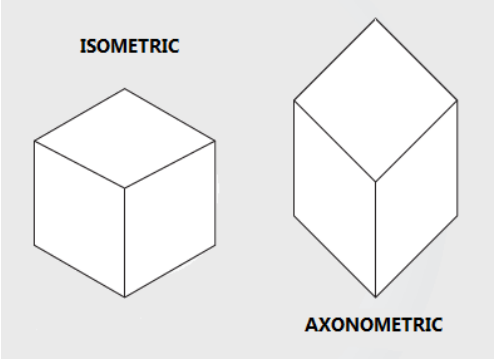


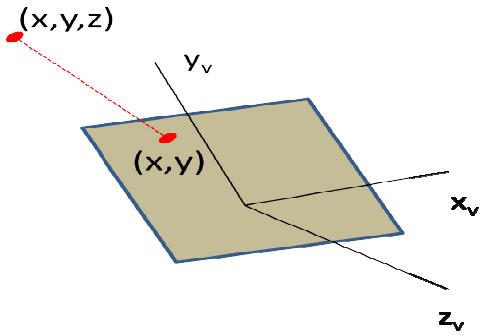
Figure shows axonometric and isometric projections of a cube.

Transformation Equation for orthographic parallel projection:

- Transformation equations for an orthographic parallel projection are straightforward
- If View plane is placed at position Z_{vp} along the Z_v axis ,then any point (x,y,z) in viewing coordinate is tranformed to projection coordinates as

$$x_p = x \text{ and } y_p = y$$

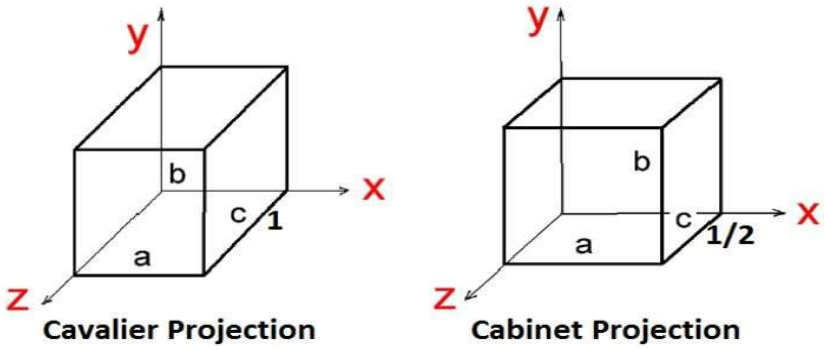
- z value is saved for finding the depth value (for using in visible surface detection algorithms



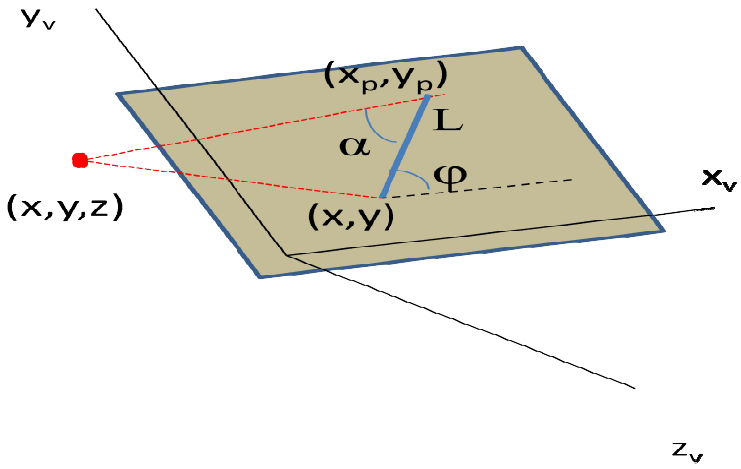
Oblique projections

There are two types of oblique projections – Cavalier and Cabinet. The Cavalier projection makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.

The Cabinet projection makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface are projected at ½ their actual length. Both the projections are shown in the following figure



Transformation Equation for oblique parallel projection:



- To project any point (x,y,z) in viewing coordinate to projection coordinates as (x_p, y_p)
- (x,y) is the orthographic projection point on the view plane
- On projection plane, a Line L of length L connects (x_p, y_p) and (x,y)
- Oblique projection line from (x,y,z) to (x_p, y_p) makes an angle α with L
- L is at an angle ϕ with the horizontal axis
- Thus we can represent projection points in terms of x,y,L and ϕ
- Thus by finding cos ϕ and sin ϕ ,we can write as

$$x_p = x + L \cos \phi$$

$$y_p = y + L \sin \phi$$

- L depends on angle α and z coordinate of the point to be projected

$$\tan \alpha = z / L \text{ ; So } L = z / \tan \alpha$$

- We can write as $L = z L_1$ where L_1 is inverse of tan α
- Thus we can rewrite Transformation equation as

$$x_p = x + z(L_1 \cos \phi)$$

$$y_p = y + z(L_1 \sin \phi)$$

- Transformation matrix for producing any parallel projection on to viewplane can be wriiten as:

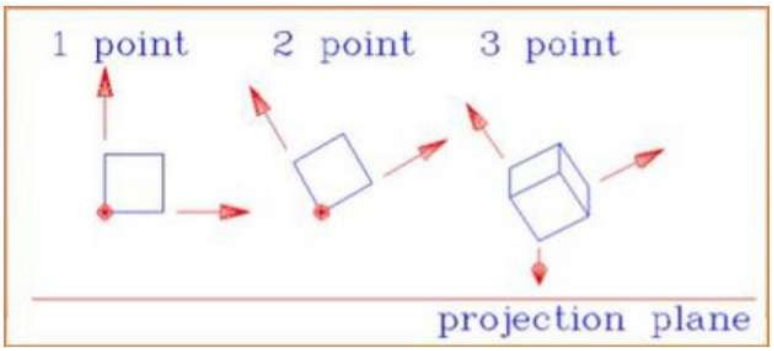
$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1 \cos \Phi & 0 \\ 0 & 1 & L_1 \sin \Phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

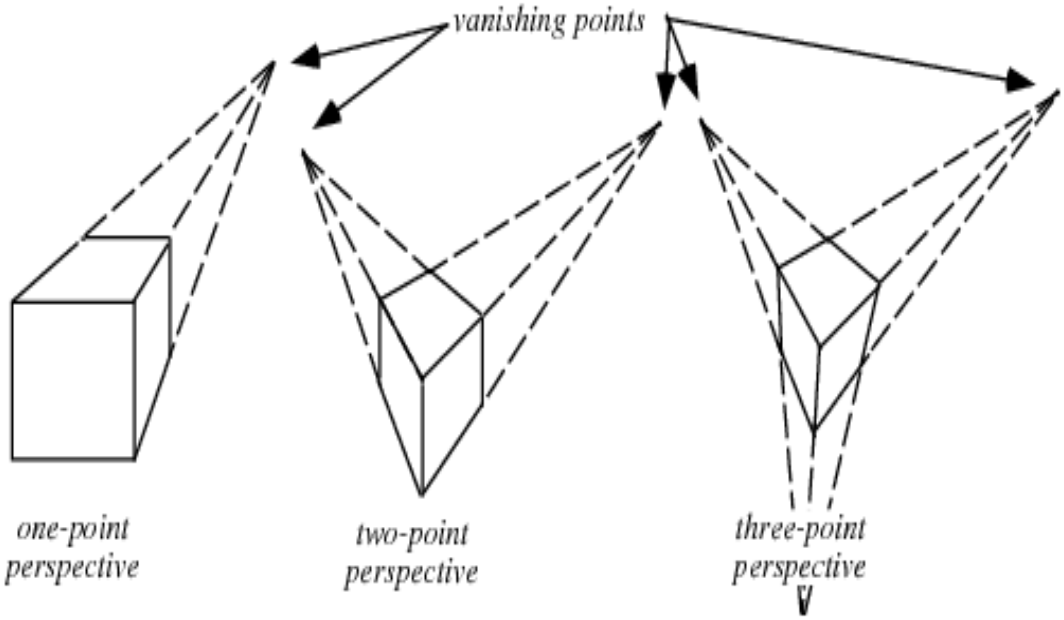
Perspective projection

In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic. The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called center of projection or projection reference point.

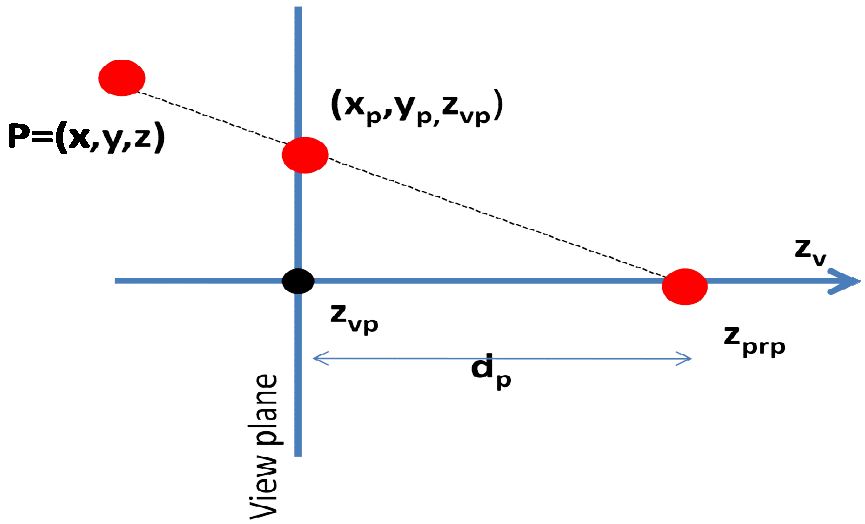
There are 3 types of perspective projections which are shown in the following chart.

- One point perspective projection is simple to draw.
- Two point perspective projection gives better impression of depth.
- Three point perspective projection is most difficult to draw.





Transformation Equation for perspective projection:



- Z_{prp} is the projection reference point along the z_v axis
- View plane is kept at point z_{vp}.
- (x',y',z') is any point that lies along the projection line
- (x_p,y_p,z_{vp}) is the point on the view plane along projection line
- In parametric form any point (x',y',z') can be represented as

  x'=x-x.u

  y'=y-y.u

  $z'=z-(z-z_{prp}).u$ where 0<=u<=1
- u=0 indicates point P=(x,y,z).
- u=1 indicates reference point coordinates $(0,0, z_{prp})$
- On the view plane. we know $z'=z_{vp}$
- By solving the equation for z', we get

  $u= (z_{vp} - z) / (z_{prp} - z)$
- Substituting u in equations for x' & y', we get

  $x_p= x.((z_{prp} - z_{vp}) / (z_{prp} - z))$

  $y_p= y.((z_{prp} - z_{vp}) / (z_{prp} - z))$
- distance of view plane from projection reference point is denoted by d_p
- If $d_p = z_{prp} - z_{vp}$,then

  $x_p= x.(d_p / (z_{prp} - z))$

  $y_p= y.(d_p / (z_{prp} - z))$

Perspective projection transformation matrix:

- In homogenous coordinate representation. Homogenous factor h is taken as $(z_{prp} - z)/ d_p$
- projection coordinates on view plane can be calculated as

$x_p = x_h / h$, then $x_h = x_p . h$

$y_p = y_h / h$, then $y_h = y_p . h$

- z value is preserved as z_{vp} and hence $z_h = z_{vp} . h$

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 2 special cases:

Tranformation equations:

$x_p = x.((z_{prp} - z_{vp}) / (z_{prp} - z))$

$y_p = y.((z_{prp} - z_{vp}) / (z_{prp} - z))$

- When $z_{vp} = 0$,equation changes like

$x_p = x.(z_{prp} / (z_{prp} - z))$

$y_p = y.(z_{prp} / (z_{prp} - z))$

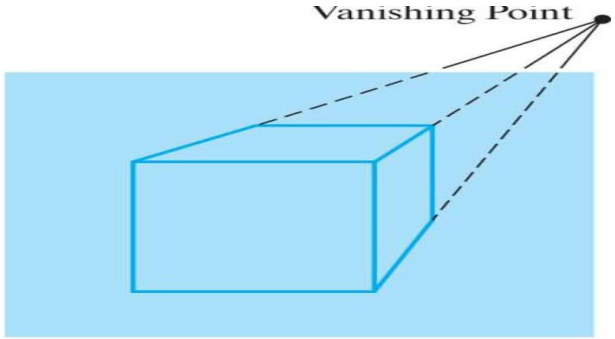
- When $z_{prp} = 0$,equation changes like

$x_p = x.(z_{vp} / z)$

$y_p = y.(z_{vp} / z)$

Vanishing point

When a three-dimensional object is projected onto a view plane using perspective transformation equations, any set of parallel lines in the object that are not parallel to the plane are projected into converging lines. The point at which a set of projected parallel lines appears to converge is called a vanishing point.Each such set of projected parallel lines will have a separate vanishing point; and in general, a scene can have any number of vanishing points, depending on how many sets of parallel lines there are in the scene. The vanishing point for any set of lines that are parallel to one of the principal axes of an object is referred to as a principal vanishing point



Visible Surface Detection Methods

A major consideration in the generation of realistic graphics displays is identifying those parts of a scene that are visible from a chosen viewing position. There are numerous for efficient identification of visible objects for different types of applications. Some methods require more memory, some involve more processing time, and some apply only to special types of objects. Decision upon a method for a particular application can depend on such factors as the complexity of the scene, type of objects to be displayed, available equipment, and whether static or animated displays are to be generated. The various algorithms are referred to as visible-surface detection methods. Sometimes these methods are also referred to as hidden-surface elimination methods.

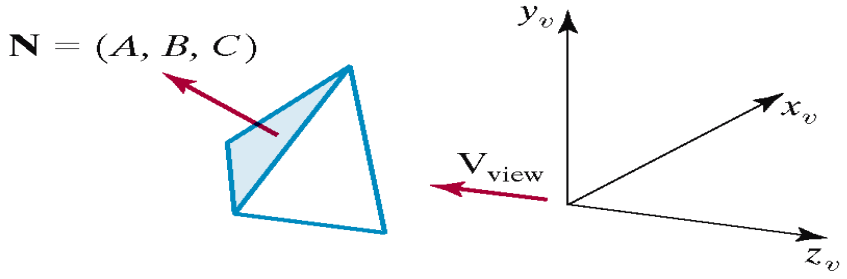
Visible-surface detection algorithms are broadly classified according to whether they deal with object definitions directly or with their projected images. These two approaches are called object-space methods and image-space methods, respectively. An object-space method compare objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible. In an image-space algorithm, visibility is decided point by point at each pixel position on the projection plane. Most visible-surface algorithms use image-space methods, although object space methods can be used effectively to locate visible surfaces in some cases. The Object-space method is implemented in physical coordinate system and image-space method is implemented in screen coordinate system.

Although there are major differences in the basic approach taken by the various visible-surface detection algorithms, most use sorting and coherence methods to improve performance. Sorting is used to facilitate depth comparisons by ordering the individual surfaces in a scene according to their distance from the view plane. Coherence methods are used to take advantage of regularities in a scene. An individual scan line can be expected to contain intervals of constant pixel intensities, and scan-line patterns often change little from one line to the next. Animation frames contain changes only in the vicinity of moving objects.

1.Back Face Detection

A fast and simple object-space method for identifying the back faces of a polyhedron is based on the "inside-outside" tests. A point *(x,* y,z) is "inside" a polygon surface with plane parameters A, B, C, and D if $Ax + By + Cz + D < 0$

When an inside point is along the line of sight to the surface, the polygon must be a back face (we are inside that face and cannot see the front of it from our viewing position).We can simplify this test by considering the normal vector N to a polygon surface, which has Cartesian components *(A, B, C).* In general, if V is a vector in the viewing direction from the eye (or "camera") position.



Then this polygon is a back face if V.N>0

Furthermore, if object descriptions have been converted to projection coordinates and our viewing direction is parallel to the viewing z axis, then V = (0, 0, *V)* and V.N= V_Z.C.

so that we only need to consider the sign of C, the Z component of the normal vector N .In a right-handed viewing system with viewing direction is along the negative z axis .Then the polygon is a back face if C < 0. AIso, we cannot see any face whose normal has z component C = 0, since our viewing direction is grazing that polygon. Thus, in general, we can label any polygon as a back face if its normal vector has a z component value C<=0.

2. Depth-Buffer Method

A commonly used image-space approach to detecting visible surfaces is the depth-buffer method, which compares surface depths at each pixel position on the projection plane. This procedure is also referred to as the z-buffer method, since object depth is usually measured from the view plane along the z axis of a viewing system. Each surface of a scene is processed separately, one point at a time across the surface. The method is usually applied to scenes containing only polygon surfaces, because depth values can be computed very quickly and the method is easy to implement. But the mcthod can be applied to nonplanar surfaces.

With object descriptions converted to projection coordinates, each *(x,* y, *z)* position on a polygon surface corresponds to the orthographic projection point *(x,* y) on the view plane. Therefore, for each pixel position *(x,* y) on the view plane, object depths can be compared by comparing z values.

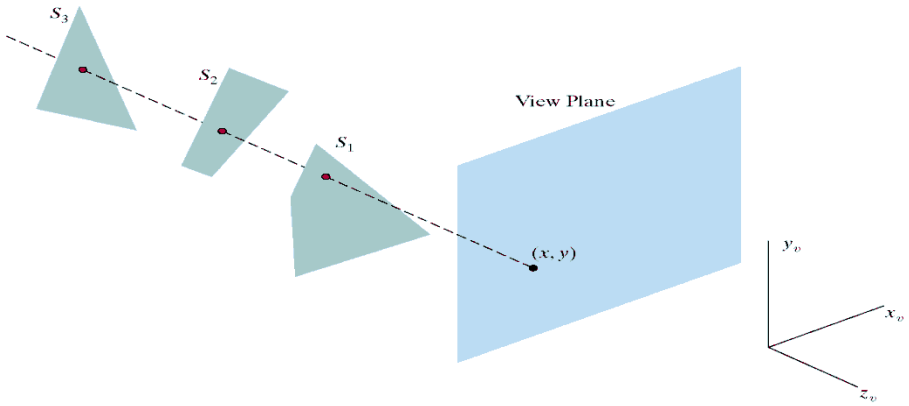


Figure shows three surfaces at varying distances along the orthographic projection line from **position (x,y) in a view plane .Surface** 1, is closest at this position, so its surface intensity value at **(x,** y) is saved. We can implement the depth-buffer algorithm in normalized coordinates,so that ***z*** values range from **0** at the back clipping plane to Z_{max} at the front clipping plane . The value of Z_{max} can be set either to 1 or to the largest value that can be stored on the system.

As implied by the name of this method, two buffer areas are required. A depth buffer is used to store depth values for each (x, y) position as surfaces are processed, and the refresh buffer stores the intensity values for each position. Initially,all positions in the depth buffer are set to 0 (minimum depth), and the refresh buffer is initialized to the background intensity. Each surface listed in the polygon tables is then processed, one scan line at a time, calculating the depth (z value) at each (x, y) pixel position. The calculated depth is compared to the value previously stored in the depth buffer at that position. If the

calculated depth is greater than the value stored in the depth buffer, the new depth value is stored, and the surface intensity at that position is determined and in the same *xy* location in the refresh buffer.

Algorithm

1. Initialize the depth buffer and refresh buffer so that for all buffer posi-tions (x, y),
   $$\text{depth}(x, y) = 0, \qquad \text{refresh}(x, y) = I_{\text{backgnd}}$$
2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
   - Calculate the depth z for each (x, y) position on the polygon.
   - If $z > \text{depth}(x, y)$, then set
     $$\text{depth}(x, y) = z, \qquad \text{refresh}(x, y) = I_{\text{surf}}(x,y)$$

Depth values for a surface position (x, y) are calculated from the plane equation for each surface:

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

For any scan line adjacent horizontal x positions or vertical y positions differ by 1 unit. The depth value of the next position (x+1,y) on the scan line can be obtained using

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

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

The ratio *-A/C* is constant for each surface, so succeeding depth values across a scan line are obtained from preceeding values with a single addition.

Advantages

- It is easy to implement.
- It reduces the speed problem if implemented in hardware.

Disadvantages

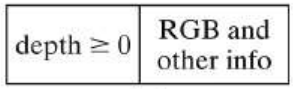
- It requires large memory.
- It is time consuming process.
- Can find only one visible surface at each pixel position.
- Deals with only opaque surfaces.
- Cannot accumulate intensity values for more than one surface.(for transparent surfaces).

3. A-Buffer Method

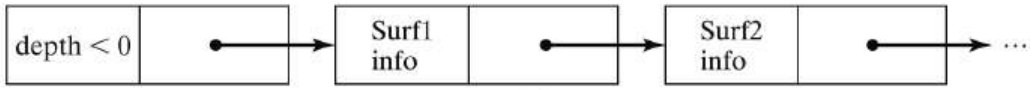
The A-buffer method is an extension of the depth-buffer method. The A-buffer expands on the depth buffer method to allow transparencies; so that each position in the buffer can reference a linked list of surfaces. Thus, more than one surface intensity can be taken into consideration at each pixel position. The key data structure in the A-buffer is the accumulation buffer.

Each position in the A-buffer has two fields −

- Depth field − It stores a positive or negative real number
- Intensity field − It stores surface-intensity information or a pointer value



(a)



(b)

If depth >= 0, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage.

If depth < 0, it indicates multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data. The surface buffer in the A-buffer includes −

- RGB intensity components
- Opacity Parameter
- Depth
- Percent of area coverage
- Surface identifier

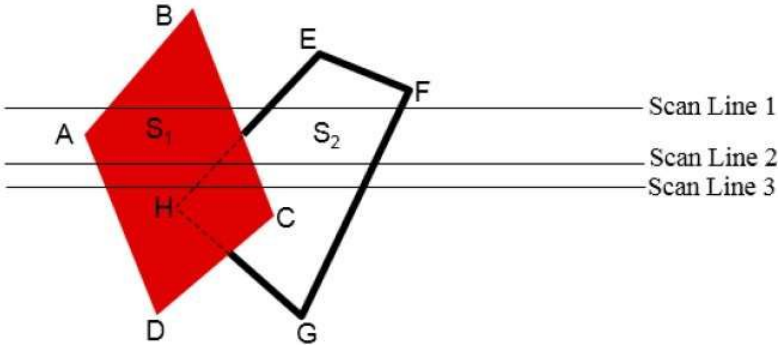
The algorithm proceeds just like the depth buffer algorithm. The depth and opacity values are used to determine the final color of a pixel.

4. Scanline Method

This image space method for removing hidden surface is an extension of the scan-line algorithm for polygon filling . As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible. Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane. When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.

Two important tables, edge table and polygon table, are maintained for this.

- The Edge Table − It contains coordinate endpoints of each line in the scene, the inverse slope of each line, and pointers into the polygon table to connect edges to surfaces.
- The Polygon Table − It contains the plane coefficients, surface material properties, other surface data, and may be pointers to the edge table.



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. We only need to perform depth calculations when multiple surfaces have their flags turned on at a certain scan-line position.

The active list for scan line 1 contains information from the edge table for edges AB, BC, EH, and FG. For positions along this scan line between edges *AB* and BC, only the flag for surface S1 is on. Therefore no depth calculations are necessary, and intensity information for surface S1 is entered from the polygon table into the refresh buffer. Similarly, between edges EH and FG, only the flag for surface S2 is on.

For scan lines 2 and 3 ,the active edge list contains edges *AD,EH, BC,* and *FG.* Along scan line 2 from edge *AD* to edge *EH,* only the flag for surface S1 is on. But between edges EH and *BC,* the flags for both surfaces are on.In this interval, depth calculations must be made using the plane coefficients for the two surfaces. For this example, the depth of surface S1 is assumed to be less than that of S2, so intensities for surface S1 are loaded into the refresh buffer until boundary BC is encountered. Then the flag for surface S1 goes off, and intensities for surface *S2* are stored until edge FG is passed.

We can take advantage of-coherence along the scan lines as we pass from one scan line to the next. The scan line 3 has the same active list of edges as scan line 2. Since no changes have occurred in line intersections, it is unnecessary again to make depth calculations between edges *EH* and *BC.* The two surfaces must be in the same orientation as determined on scan line 2, so the intensities for surface S1 can be entered without further calculations.

Advantage: Any number of overlapping polygon surfaces can be processed with this scan-line method.

5.Depth Sorting Method (Painter's Algprithm)

Using both image-space and object-space operations, the depth-sorting method performs the following basic functions:

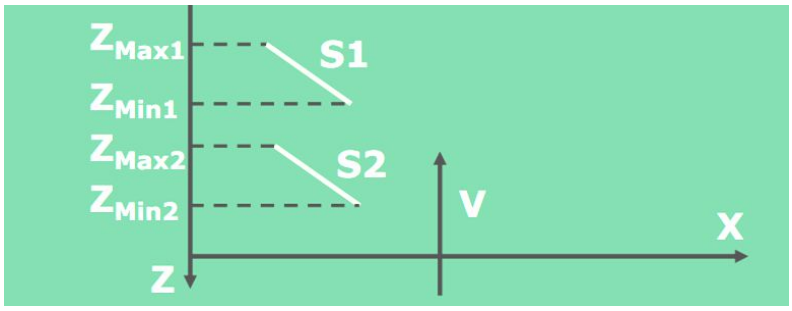
1. Surfaces are sorted in order of decreasing depth.
2. Surfaces are scan converted in order, starting with the surface of greatest depth.

Sorting operations are carried out in both image and object space, and 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. In creating an oil painting, an artist first paints the background colors. Next, the most distant objects are added, then the nearer objects, and so forth. At the final step, the foreground objects are painted on the canvas over the background and other objects that have been painted on the canvas. Each layer of paint covers up the previous layer. Using a similar technique, we first sort surfaces according to their distance from the view plane. The intensity values for the farthest surface are then entered into the refresh buffer. Taking each succeeding surface in turn (in decreasing depth order), we "paint" the surface intensities onto the frame buffer over the intensities of the previously processed surfaces. Hence the name Painters Algorithm.

Painting polygon surfaces onto the frame buffer according to depth is carried out in several steps. Assuming we are viewing along the-z direction,surfaces are ordered on the first pass according to the smallest ***z*** value on each surface. Surface S with the greatest depth is then compared to the other surfaces in the list to determine whether there are any overlaps in depth. If no depth overlaps occur, S is scan converted.

Figure shows two surfaces that overlap in the **xy** plane but have no depth overlap.

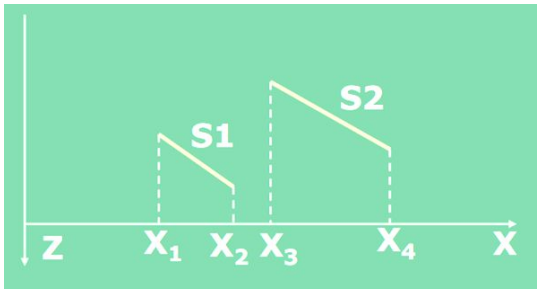


This process is then repeated for the next surface in the list. As long as no overlaps occur, each surface is processed in depth order until all have been scan converted. If a depth overlap is detected at any point in the list, we need to make some additional comparisons to determine whether any of the surfaces should be reordered.

We make the following tests for each surface that overlaps with S. If anyone of these tests is true, no reordering is necessary for that surface.

Note: In the following figures, we have considered s1 as the surface first scanconverted with greatest depth and s2 as overlapping surface identified.

1. The bounding rectangles in the **xy** plane for the two surfaces do not overlap.



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

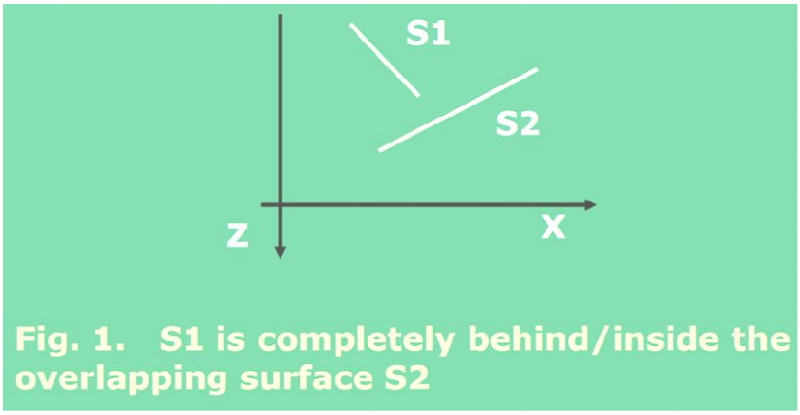


Fig. 1. S1 is completely behind/inside the overlapping surface S2

3. The overlapping surface is completely in front of S relative to the viewing position.

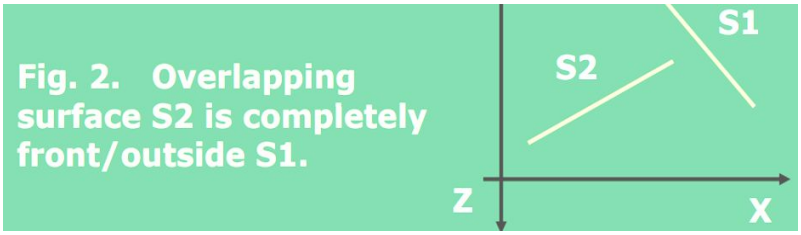
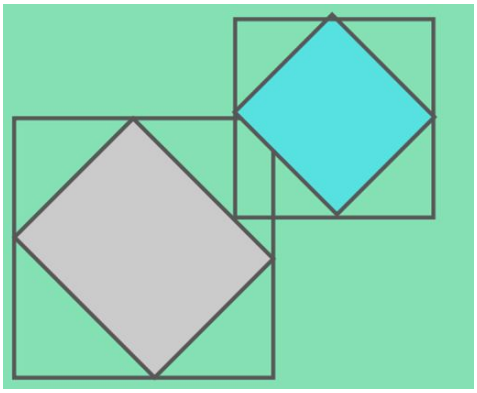


Fig. 2. Overlapping surface S2 is completely front/outside S1.

In the figure, S2 is in front of S1, but S1 is not completely inside S2.

4. The projections of the two surfaces onto the view plane do not overlap.



Method to test the 4 cases:

TEST #1: Check for overlap in x direction, then we check for overlap in y direction. If either of these directions show no overlap, 2 planes cannot obscure eachother.

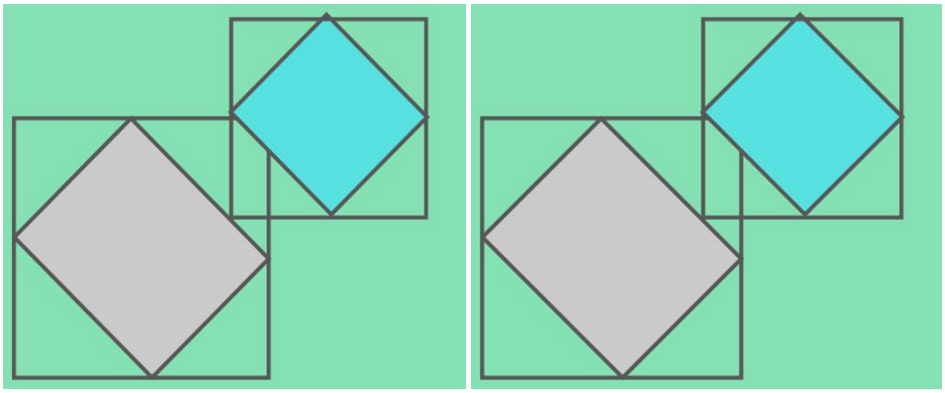
Test #2 & Test #3: We can perform tests 2 and 3 with an "inside-outside" polygon test. That is, we substitute the coordinates for all vertices of S into the plane equation for the overlapping surface and check the sign of the result.

Test #2: If all vertices of S1 are inside or behind S2 then S1 is behind S2. (Case 2: Fig. 1).

Test #3: If all vertices of S2 are infront of S1, but all vertices of surface S1 is not completely "inside or behind" S2 (i.e test 2 is not true).

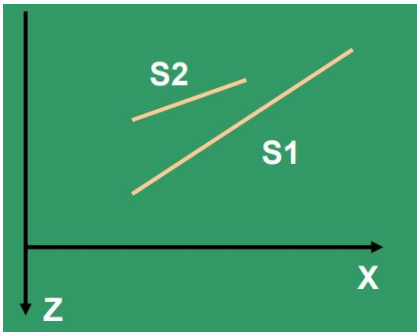
TEST #4: If tests 1 through 3 have all failed, we try test 4 by checking for intersections between the bounding edges of the two surfaces using line equations in the **xy** plane.

As demonstrated in Fig, two surfaces may or may not intersect .



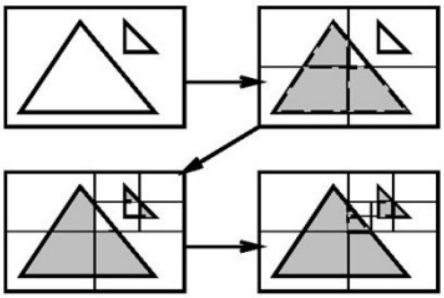
We perform these tests in the order listed and proceed to the next overlapping surface as soon as we find one of the tests is true. If all the overlapping surfaces pass at least one of these tests, none of them is behind S. No reordering is then necessary and *S* is scan converted.

If all four tests fail with a particular overlapping surface S', we interchange surfaces S and S' in the sorted list.In the figure, Surface S1 has greater depth but it obscures surface s2.



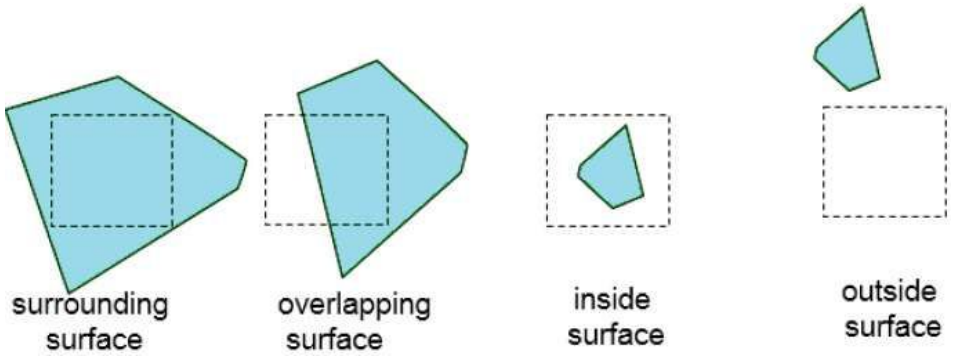
6.Area Subdivision Method

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.

- Surrounding surface − One that completely encloses the area.
- Overlapping surface − One that is partly inside and partly outside the area.
- Inside surface − One that is completely inside the area.
- Outside surface − One that is completely outside the area.



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.

Test 1: To check all surfaces are outside surfaces:

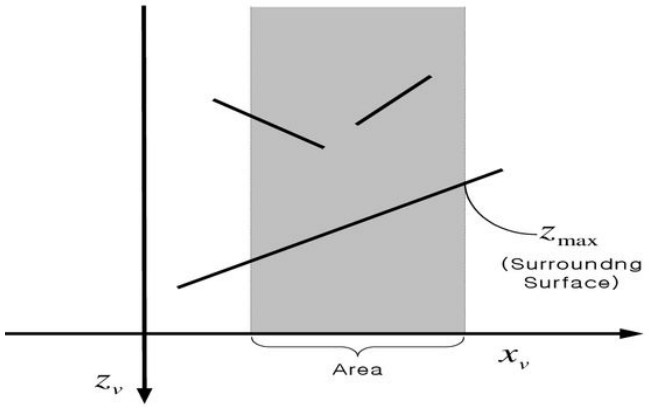
- It can be carried out by checking the bounding rectangles of all surfaces against the area boundaries.

Test 2: To check only one inside, overlapping or surrounding surface is in the area:

- It can also use the bounding rectangles in the xy plane to identify an inside surface.
- For other types of surfaces, the bounding rectangles can be used as an initial check. If a single bounding rectangle intersects the area in some way, additional checks are used to determine whether the surface is surrounding, overlapping, or outside.
- Once a single inside, overlapping or surrounding surface has been identified, its pixel intensities are transferred to the appropriate area within the frame buffer.

Test 3: To check surrounding surface obscures all other surfaces:

- Two methods: Minimum Depth computation & Using plane equation.
- Minimum Depth computation: Order surfaces according to their minimum depth from the view plane. For each surrounding surface, we then compute the maximum depth within the area under consideration. If the maximum depth of one of these surrounding surfaces is closer to the view plane than the minimum depth of all other surfaces within the area, test 3 is satisfied.



- Using plane equation: use plane equations to calculate depth values at the four vertices of the area for all surrounding, overlapping, and inside surfaces, If the calculated depths for one of the surrounding surfaces is less than the calculated depths for all other surfaces, test 3 is true.
- Then the area can be filled with the intensity values of the surrounding surface.

Important questions

1. Explain the concept of vanishing point.
2. Summarize the scan line algorithm for visible surface detection
3. Explain parallel projection.
4. How is A Buffer method used for back face detection
5. Explain Depth Buffer method used for back face detection
6. Explain perspective projection of a three dimensional object and transform points along projection lines that meet at the projection reference point.
7. Distinguish between 1 point perspective projection & 2 point perspective projection
8. Distinguish between object space method and image space methods for visible surface detection?
9. Explain Backface detection method.
10. Write note on octrees
11. Explain parallel and perspective projection
12. Write the steps in 3D viewing.
13. Derive the matrix for parallel and perspective projection.
14. Explain the different types of perspective projections
15. Analyse the advantage of A Buffer method over Z Buffer method
16. Explain Depth sorting method for visible surface detection. Why the method is known painters algorithm.
17. Explain area subdivision method for visible surface detection
18. Explain any two object space method for visible surface detection.

try it now

A KTU STUDENTS PLATFORM

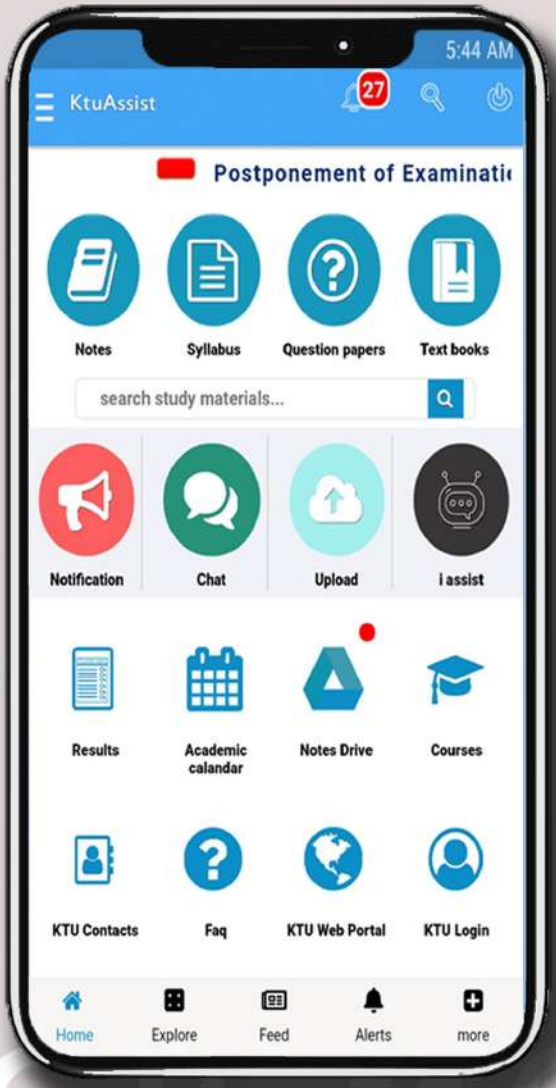
SYLLABUS

NOTES

TEXT BOOKS

QUESTION PAPERS

KTU NOTIFICATION



DOWNLOAD IT FROM GOOGLE PLAY

CHAT

FAQ

LOGIN

CALENDAR

MUCH MORE

DOWNLOAD APP

ktuassist.in

instagram.com/ktu_assist

facebook.com/ktuassist