8M Projection: There are two types
① Parallel projection ② Perspective proj

Once a world co-ordinate description of the objs in a scene are converted to viewing co-ord. We can project 3D objs onto the 2D view plane. There are 2 basic proj method
llel projection: co ord position are transformed to the view plane along parallel lines.

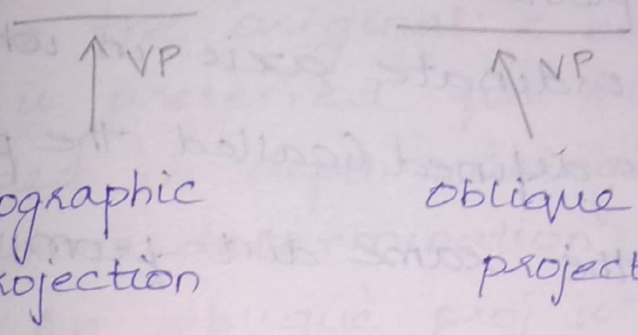
Perspective projection: Here, obj positions are transformed to the view plane alon lines that converge to a pt called projection reference pt (or centre of projection). The projected view of an obj is determined by calculating the intersection of projection lines with the view plane.

19/12 5M

Parallel projection 11d & persp 16M

We can specify a parallel projection with a projection vector that defines the direction for projection line

If the proj is perpendicular to view plane we have orthographic parallel proj otherwise we have oblique parallel projection.



orthographic projection

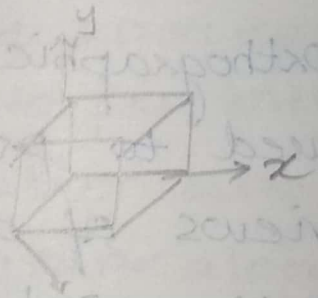
Oblique projection

Orthographic projections are most often used to produce front, side and top views of an obj. Front, side ~~and rear~~ orthographic proj of an obj are called elevation. Top orthographic proj is called plan view

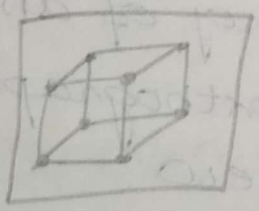
Engineering and architectural drawings commonly employ these orthographic proj bcz length and angles are accurately depicted and can be measured from the drawing.

We can also form orthographic projections that display more than one face of an obj. Such views are called axonometric orthographic projection. The most commonly used axonometric projection is isometric projection. We can generate an isometric projection by aligning the proj plane so that it intersects each coordinate axis in which the obj is defined (called the principle axis) at the same dist from the origin.

Isometric proj of cube

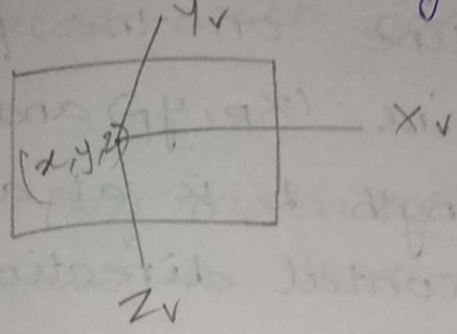


(Ref book)



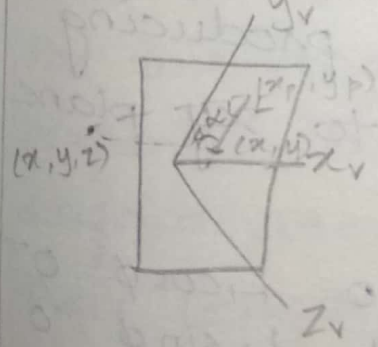
There are 8 pos one in each octant for obtaining an isometric view. All the 3 principle axis are equal in isometric proj whereas it is not equal in axonometric proj.

Trans eqn for orthographic parallel proj if the view plane is placed at position Z_{vp} along the Z_v axis



Then any pt (x, y, z) in viewing coordinates is transformed to projection coordinates, $x_p = x$; $y_p = y$ where the original z coordinate val is preserved for depth info needed in depth cueing and visible-surface determination procedure.

An oblique proj is obtained by projecting pts along parallel lines that are not perpendicular to the proj plane. An oblique proj vector is specified with 2 angles, α and ϕ. Pts (x, y, z) is projected to a position (x_p, y_p)



oblique proj of coord pos (x, y, z) to position (x_p, y_p) on the view plane

Orthographic proj coord on the plane are (x, y)

The oblique proj line from (x, y, z) to (x_p, y_p) makes an angle ϕ with the line on the proj plane that joins (x_p, y_p) and (x, y). This line of length L is at angle ϕ with the horizontal direction in the proj plane. We can express proj coord interms of x, y, L and ϕ as $x_p = x + L\cos\phi$; $y_p = y + L\sin\phi$ → (1)

Length L depends on angle α and z coord of pt to be projected,

$$\tan\alpha = \frac{z}{L} \Rightarrow L = \frac{z}{\tan\alpha} \Rightarrow zL_1$$

where $L_1 = \frac{1}{\tan\alpha} = \tan^{-1}\alpha$ which is also called the value of L when $z = 1$.

(1) becomes,

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

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

The transf matrix for producing any parallel proj on to xy plane can be written as

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow (2)$$

An orthographic proj is obtained when $L_1 = 0$ (which occurs at a proj angle of 90°). Oblique proj are generated with non-zero values for L_1.

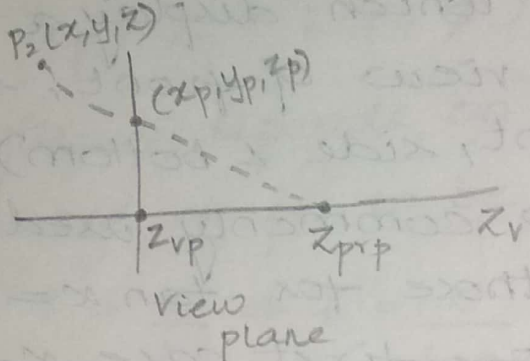
Note: Proj matrix (5) has a similar structure to that of z axis share matrix.

Common choices for angle ϕ are 30° and 45° which display a combination view of front, side & top (or front, side & bottom) of an object. Two commonly used values of α are those for $\tan\alpha = 1$ and $\tan\alpha = 2$. For first case $\alpha = 45°$ & the views obtained are cldcavalior projection. All lines perpendicular to proj plane and projected with no change in length.

When proj angle α is chosen, so that $\tan\alpha = 2$ the resultant view is called cabinet proj. For this angle (approx 63.4°), lines perpendicular to the viewing surface are projected at one-of their length. Cabinent proj appear more realistic than cavalior proj bcz of this reduction in length of

perpendicular — 8M

23/2 8M Perspective projection: To obtain a perspective projection of a 3D object, we transform points along proj lines that meet at the proj reference pt. Suppose we set the proj ref pt at position z_{prp} along the z_v & we place the view plane at z_{vp}



As shown in the abv diag, we can write eqns describing co ordinate pos along this perspective projection line in parametric form as,

$$\left.\begin{aligned} x' &= x - xu \\ y' &= y - yu \\ z' &= z - (z - z_{prp})\,u \end{aligned}\right\} \quad -(1)$$

Parameter 'u' takes values from 0 to 1 and co ordinate pos (x', y', z') represents any pt along the proj line. when $u = 0$, in (1) we are at pos $P = (x, y, z)$ when $u = 1$, in (1) ~~we are at pos P = (x~~ we have proj ref pt coordinates $(0, 0, z_{prp})$

On the view plane $z' = z_{vp}$. We can solve z' eqn for parameter u.

$$u = \frac{z_{vp} - z}{z_{prp} - z} \quad \rightarrow (2)$$

Substituting this value of u, in eqns for x' and y', we obtain the perspective transf eqn,

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z - z_{prp}}\right) = x\left(\frac{d_p}{z - z_{prp}}\right)$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z - z_{prp}}\right) = y\left(\frac{d_p}{z - z_{prp}}\right) \quad \rightarrow (3)$$

where $d_p = z_{prp} - z_{vp}$ is the dist of view plane from proj ref pt.

Using a 3D homogenous coordinate representation, we can write the perspective proj trans (3) in matrix form as,

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

In this representation, homogenous factor

$$h = \frac{z - z_{prp}}{d_p}, \quad x_p = \frac{x_h}{h} \; ; \; y_p = \frac{y_h}{h}$$

If the view plane is taken to be uv plane, then $z_{vp} = 0$, then the proj coordinates are, Put $z_{vp} = 0$ in (3)

$$x_p = x z_{prp} / z - z_{prp} \; ; \; y_p = y z_{prp} / z - z_{prp}$$

$x_p = \frac{1}{\frac{z}{z_{prp}} - 1}$ $y_p =$ (Ref Book)

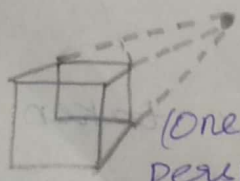
When a 3D obj is projected onto a view plane using persp eqn, any set of parallel lines in the obj that are not parallel to the plane are projected into converging lines. Parallel lines that are ||el to view plane will be projected as ||el lines.

Q.M Point at which a set of projected ||el lines appears to converge is called a vanishing point.

NOTE: In general a scene can any no of vanishing pts depending on how many set of ||el pts are there in scene.

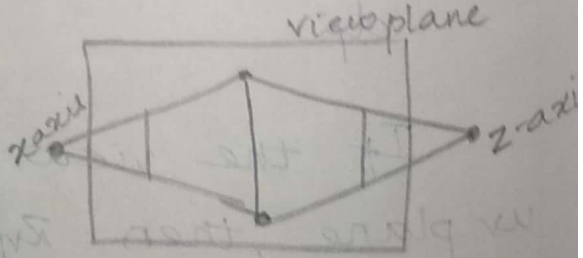
not needed for exam

Vanishing pt for any set of lines that are ||el to one of principle axis of an obj is referred to as principle vanishing.



(One-pt perspective projection)

(1)



(Two-pt perspective projection)

(2)

We control the no of principal vanishing pts (one, two or three) with the orientating of projection plane & perspective proj are acc classified as one-point, Two-point, Three-point proj. No of princ. vanishing pt in a proj is determined by the no of princ axis intersecting the view plane. In diag ① view plane is aligned ||el to xy object plane so that only z-axis is intersected. This orientation produces One-Point proj with z-axis vanishing pt.

In diag ②, the proj plane intersects both x and z axis but not y axis. The resulting Two-Point persp proj contains both x and z axis vanishing pts.

—SM

VISIBLE SURFACE REDUCTION METHOD :-

A major consideration in generat of graphic disp is identifying those parts of a scene that are visible from a chosen viewing position. There are many approaches we can take to solve this problem. Some methods req more memory. Some involve more processing time. Some apply only to spl types of objs. Deciding upon a method

for a particular application can depend on such factors as complexity of scene, types of objs to be displayed, available equipment, whether static or animated displays are to be generated. Various algo are referred to as Visible-Surface Reduction Methods. Sometimes these methods are ref to as Hidden-Surface Elimination method.

CLASSIFICATION OF VISIBLE SURFACE REDUCTION ALGORITHM: 2M

Visible surf red algo are broadly classified acc to whether they deal with obj defn directly or with projected imgs. Approaches are called Object-Space Method & Image-Space Method.

2M An obj-space method compares objs & parts of objs to each other within the scene defn to determine which surfs as a whole we should label as visible.

2M In img-space algo, visibility is decided pt by pt at each pixel position on the proj plane.

Most visible surf algo, use img-space method although obj-space method can be used effectively to locate visible surf. in some cases.

Diff btw: most visible surf red algo use sorting and coherence method to improve performance.

Sorting is used to facilitate depth comparison by ordering the individual surfaces acc to the dist from the view plane.

Coherence method are used to take adv of regularities in a scene.

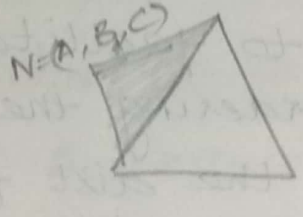
BACK-FACE REDUCTION 8M Fast and simple obj-space method for identifying the back faces of a polyhedron is based on inside-outside test. A pt (x, y, z) is inside a polygon surf with plain parameters A, B, C and D if ~~$ax+by+cz+d < 0$~~

$$Ax + By + Cz + D < 0$$

26/2

When an inside pt is along the line of side to the surf, 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 simply test by considering the normal vector N,



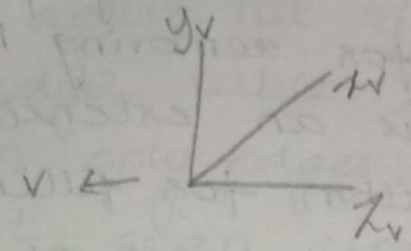
The vector N, to the polygon surf which has cartesian compone N (A, B, C)

In general, V is a vector in the viewing direction from the eye (or camera) pos as shown in above diag. Then this polygon is a back face $V \cdot N > 0$

If obj description have been converted to proj coordinates and our viewing direction is ||el to Z_v axis

Then $V = (0, 0, V_z)$ and $V \cdot N = V_z C$

In a right handed viewing system with the viewing dir along the negative z_v axis as in below diag.



The polygon is a back face if $C < 0$
Also we cannot see any face whose normal as z component $C = 0$

Thus in general, we can label any polygon as a back face if it is normal vector as a z component val $C \leq 0$.

Left handed system - clockwise dir
Right handed system - counter clockwise dir

By examining parameter C for diff planes defining an obj we can immediately identify all back faces. If C contains only non-overlapping polyhedra, then again all hidden surf are identified with back face method.

Back face removal can be expected to remove abt half of polygon surf in a scene ~~from~~ to rem further the visibility test.

Depth Buffer Method - not there in syllabus

SCAN-LINE METHOD :- This img space method is used for removing hidden surfaces which is an extension of scan line algorithm for filling polygon interiors. Instead of filling just one surf, we now deal with multiple surf 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 surf to determine which is nearest to view plane. When visible surf has been determined, the intensity val for that pos is entered into refresh buffer. We assume that tables are set up for various surfaces that include ~~edge~~ edge table and polygon table. ~~edge~~ edge table contains co ordinate end pts for each line in a scene, inverse slope of each line & pts into polygon table to identify surf bounded by each line.

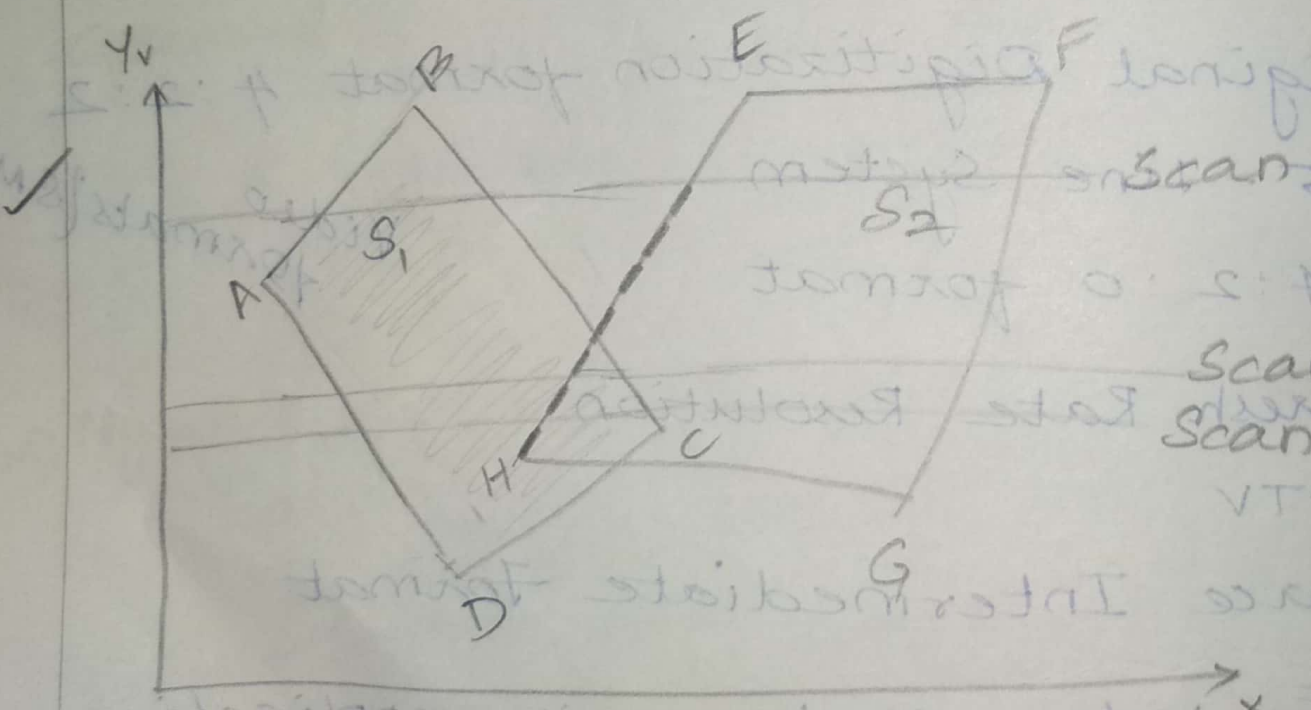
Polygon table contains co-eff of plane eqn for each surf, intensity info for the surfaces & possibly ptrs into the h table.

An active list of edges is set up in the edge table. Active list will contain only edges that cross the current scan line, sorted in order of increasing x. We define a flag for each surf that is set on or off to indicate whether a pos along a scan line is inside/outside a surface.

Scan lines are processed from left to right. At the left most bounda of a surf, the surf flag is turned on. At the right most boundary it is turned off.

Unit-3

5/3/18 SCAN-LINE METHOD:



Edges, surface, & explanation

Dashed line indicate boundary of hidden surfaces

Active list for scan line 1 contains info from the edge table for the edges AB, BC, EH, FG. For positions along the scan line btw two edges AB and BC only the flag for surface S_1 · ~~Therefore~~ S_1 is on. Therefore no depth calculations are necessary and intensity info for surf S_1 is entered in the polygon table. in the refresh buffer.

Btw the edges ~~only~~ EH and FG only flag for the surf S_2. No other pos along the scan line 1 intersect surfaces, so ~~other~~ the intensity values in other areas are set to background intensity.

For scan lines 2 and 3 in the abr diag, active edge list contains edges AD, EH, BC & FG. Along scanline 2 from edge AD to edge EH only the flag for the surface S_1 is on. Btw the edges EH and BC flags for both surfaces are on. We can take advantage of coherence along the scanlines as we pass from one scan line to next.

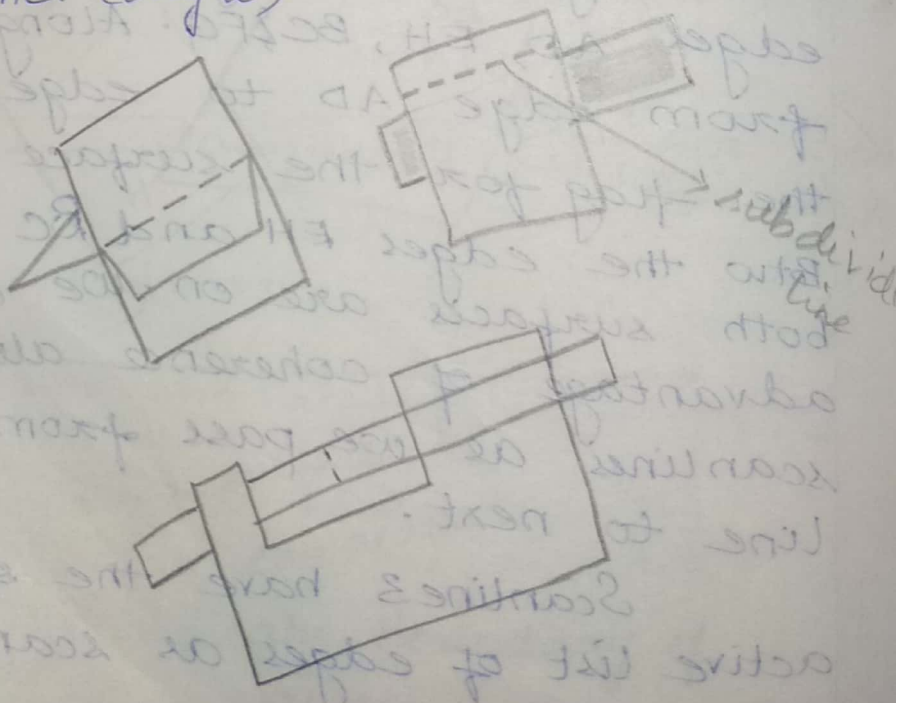
Scanline 3 have the same active list of edges as scan line 2.

Since no changes has occured in line intersection it is unnecessary again to make depth calculations btw edges BC and EH. Two surfaces must be in same orientation as determined on scanline a, so the intensity for surf 1 can be entered without further calc.

~~Dashed lines indicate the~~

Any no of overlapping polygon surfaces can be processed with this scan line method. Flags for the surfaces are said to indicate whether a position is inside or outside and depth calc are performed if surfaces are overlapped. When these coherence methods are used, we need to be careful to keep track which surface section is visible on each scan line.

This works only if surfaces dont cut through or otherwise cyclically overlap each other. (dig 2)

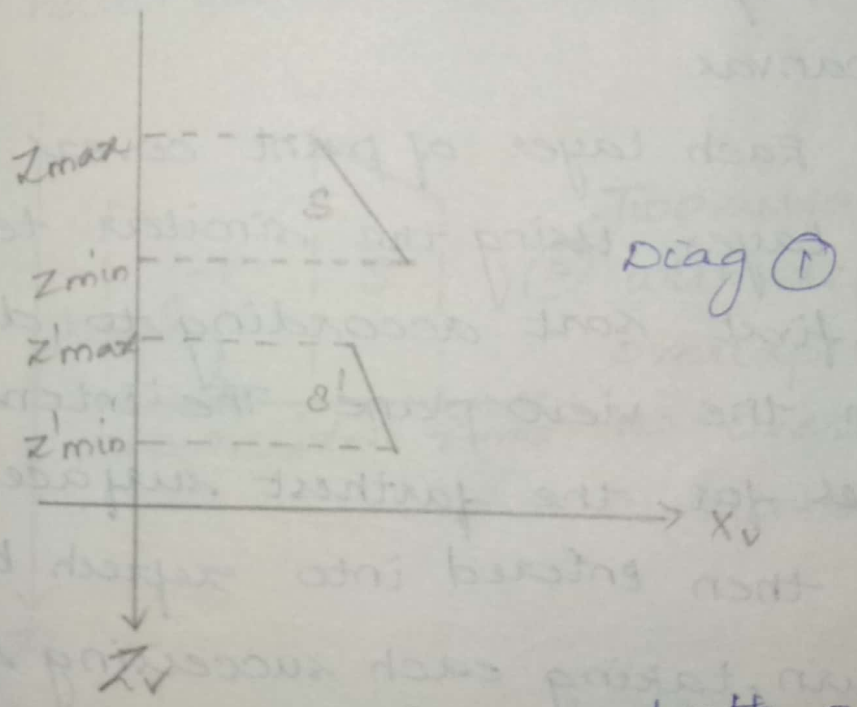


If any kind of cyclic overlap is present in a scene, we can divide the surfaces to eliminate overlaps. The dashed lines in this fig indicates where planes could be subdivided to form two distint surfaces so that cyclic overlaps are eliminated.

8M Depth Sorting Method (Painter's algo) :- Using both img space and obj space operations, depth sorting performs the following basic functs

1) Surface are sorted in order of decreasing depth.

2) Surfaces are scanned converted in order, starting with surface of greatest depth



Diag (1)

(Two surfaces with no depth overlap)

Sorting operations are carried out both in img & obj space and the scan conversion of the polygon surfaces is performed in img space. This method for solving the hidden surf prob is often referred to as Painter's Algorithm.

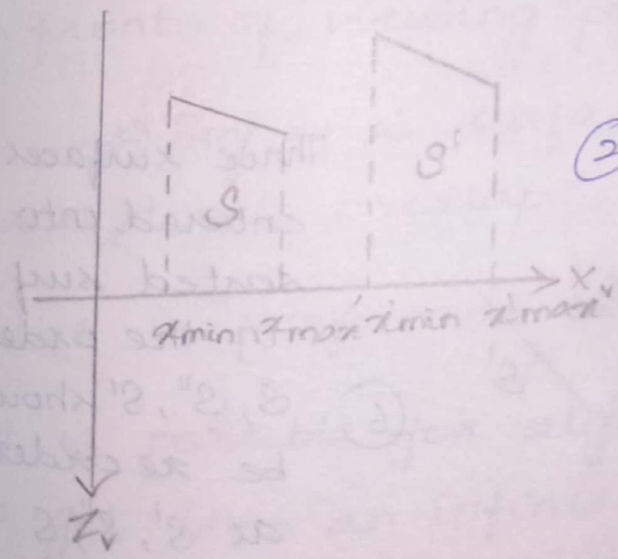
In creating an oil painting, an artist first paints the bg color then most distant objs are added and nearer objs and so forth. At the final step, the foreground objs are painted on the canvas over the bg & other objs that have been painted on the canvas.

Each layer of paint covers up the prev layer. Using the similar technique we first sort according to dist from the view plane. The intensity values for the farthest surface are then entered into refresh buffer. In turn, taking each succeeding surf (in decreasing ~~increasing~~ depth order), we paint the surf intensity on to the frame buffer over the intensities of

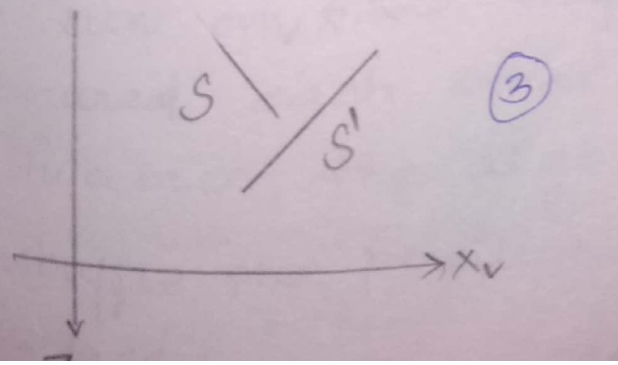
previously processed surfaces.

Assuming we are viewing along z direction, surfaces are ordered on the first path acc to largest z value on each surf. Surf S with greater depth is then compared to other surfaces in the list to determine whether there are any overlap in the depth. If no depth overlap occurs, S is scan converted.

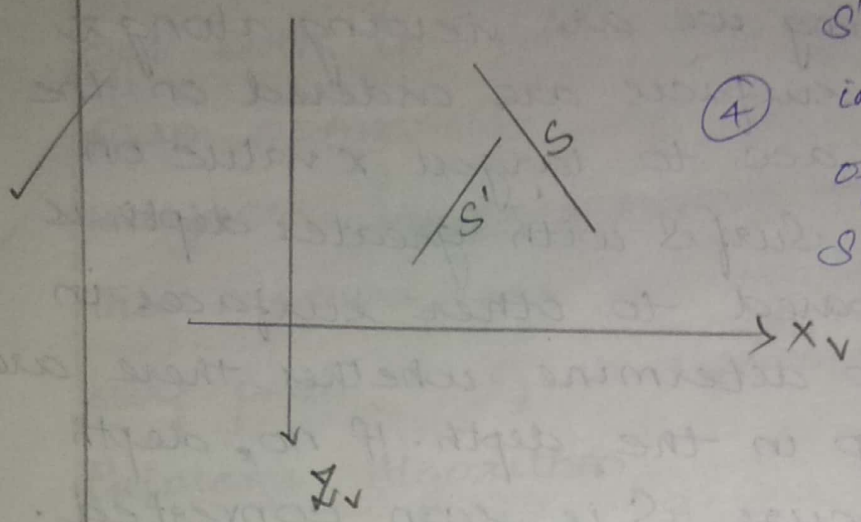
Diag ① shows two surfaces that overlap in xy-plane, but have no depth overlap. This process is then repeated for the next surf in the list as long as no overlaps occurs, each surf is processed in depth order until all have been scan converted.



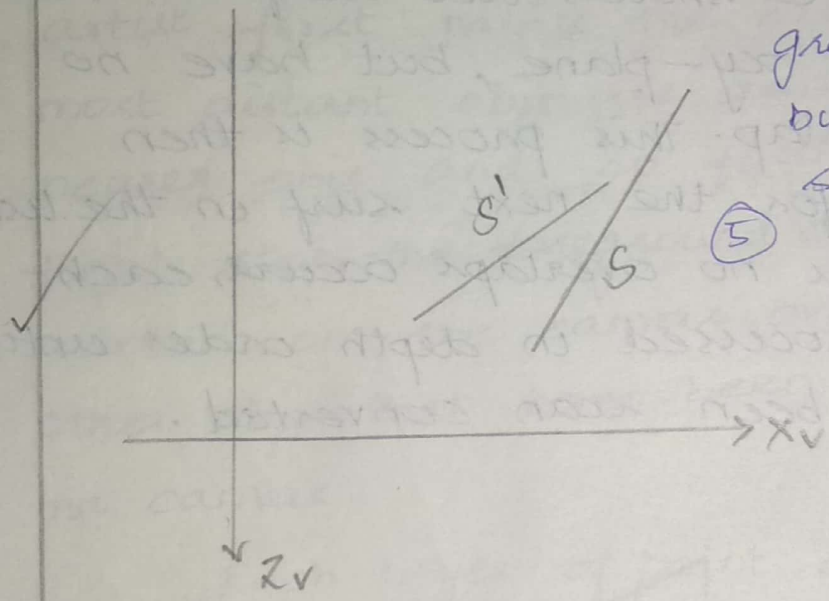
② Two surfaces with depth overlap with no overlap in x-direction.



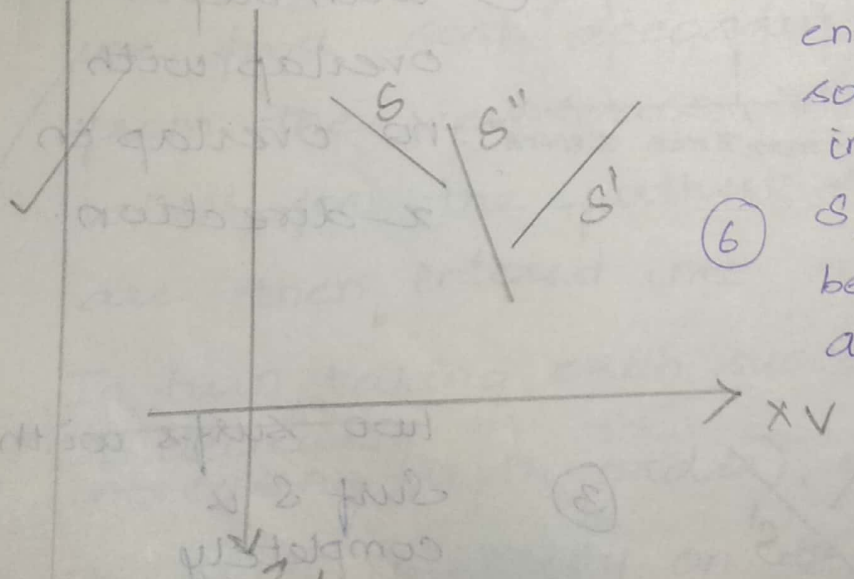
③ ~~Two surfs with~~ Surf S is completely beyond (inside) the overlapping surf S'



(4) Overlapping surf S' is completely in front (outside) of the surf S but S is not completely beyond S'.



(5) Surf S has greater depth but obscures surf S'.



(6) Three surfaces entered into sorted surf list in the order S, S'', S' should be reordered as S', S'', S

If a depth overlap is detected in any pt in the list, we need to make additional comparison to determine whether any of the surf must be reordered.

Foll test for each surf that overlap with S must be considered. If any one of these test is true, no reordering is needed for that surf. Tests are listed in order of inc difficulties

1) Bounding rect in xy plane for 2 surfaces donot overlap

2) Surf S is completely beyond ~~behind~~ the overlapping surf relative to viewing pos

3) Overlapping surf is completely in front of S, relative to viewing position

4) Proj of 2 surfaces on to the view plane donot overlap

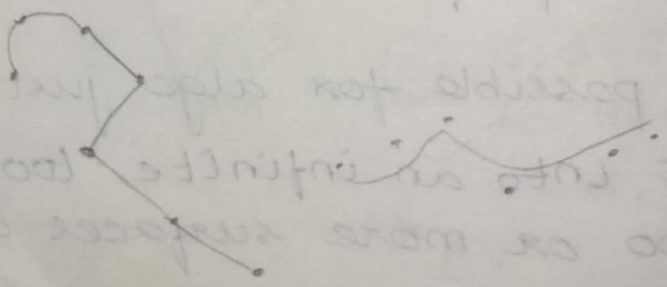
Test 1 is performed in two parts

11/3 It is possible for algo just outlined to get into an infinite loop if
2M ~~in~~ two or more surfaces alternatively obscured each other in such a situation, the algo will continually reshuffle the positions of the overlapping surface.

To avoid circuits, we can flag any surface that has been re ordered to a further depth position so that it cannot be moved again. If an attempt is made to switch surf is again timed we divide it into 2 parts to eliminate cyclic overlap. the original surf is then replaced by 2 new surfaces & we continue processing as before.

16M SPLINE REPRESENTATION:

2M A spline is a flexible strip used to produce a smooth curve through a designated set of pts. Several small blades are distributed along the length of strip to hold it in the pos on the drafting table as the curve is drawn



Mathematically, we can describe such a curve by piece wise polynomial func whose first and second derivatives

are continuous across the various curve sections.

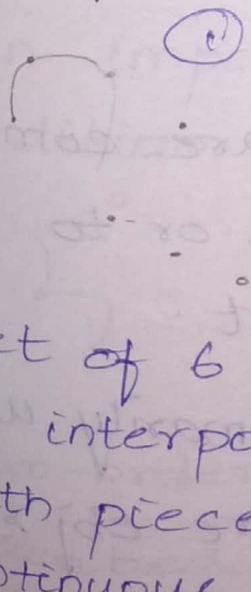
Use:

Splines are used in graphic applns to design curve and surf shapes, to digitize drawings for computer storage and to specify animation path for the objs or camera in a scene. Typical CAD applns for spline include the design of automobile bodies, aircraft & space craft surfaces and ship hulls.

Spline curve refers to any composite term curve formed with polynomial section satisfying specifying continuity conditions at the boundary of pieces.

Spline surface can be described with 2 sets of orthogonal spline curves

Interpolation & Approximation Spline:



Set of 6 ctrl pts interpolated with piece wise continuous polynomial section

Set of 6 ctrl pts approximated with piece wise cont. polynomial section

A spline curve is specified by giving a set of coord position called ctrl pts which indicate the general shape of the curve. These ctrl pts are then fitted with piece wise continuous parametric polynomial func in one of the 2 ways

i) Interpolate

ii) Approximate

When polynomial sections are fitted so that the curve passes thru each ctrl pt as shown in diag (1).
Resulting curve is said to interpolate the set of ctrl pts.

Polynomials are fitted to general ctrl pt path without necessarily passing thru any ctrl pt. Resulting curve is said to approximate the set of ctrl pts.

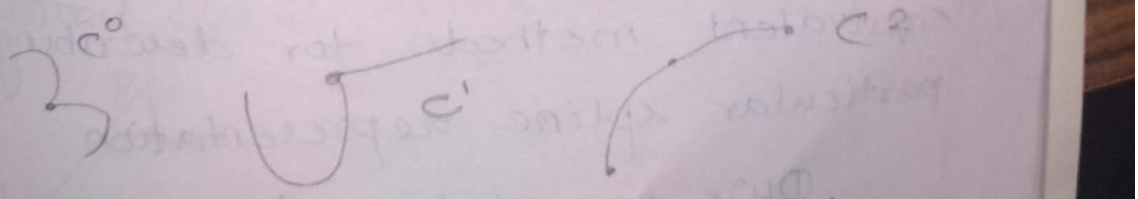
Interpolation curves are commonly used to digitize drawings or to specify animation ~~path~~ part.

Appr. curves are primarily used as design tools to structure obj surfa

A spline curve is defined, modified and manipulated with operations on the ctrl pts By interactively collecting spacial pos for ctrl pts, a designer can set up a initial curve. After a polyn fit is displayed for a given set of ctrl pts, the designer can then reposition some or all of the ctrl pts to restructure the shape of the curve. In addition, curve can be translated, rotated or scaled with transf. applied to the ctrl pts. CAD packages can also insert extra ctrl pts to help a designer in adjusting the curve shapes.

Parametric Continuity Conditions :

To ensure a smooth transition from one section of piece wise parametric curve to the next, we can impose various continuity condition at the connection pt.



Zero-order parametric continuity described as C^0 continuity means

simply that the curve meets

First order parametric continuity referred as C^1 continuity means the first order parametric continuity (tangent lines) of the co-ord func.

Second order parametric continuity or C^2 continuity means that both the first & second parametric derivatives of the 2 curve sections are same at the intersection.

Geometric Continuity Condition : An alternate method for joining 2 successive curve sections is to specify conditions for geometric continuity.

Zero order geometric Continuity described as G^0 is same as zero order parametric continuity

G^1 same as C^1 ; G^2 same as C^2

Spline specification : There are 3 equivalent methods for describing particular spline representation

① We can state set of bound conditions that are imposed on

the spline

(or)② we can state matrix that characterize the spline (or)② we can state the set of blending funcs (basis func) that determine how to specify geometric constraints on the curve combined to calculate the pos along the curve path

Cubic Spline Interpolation Method

16M This class of splines is most often used to set a path for obj motions (or) to provide a representation for an existing object

$$P_k = (x_k, y_k, z_k) \quad k = 0, 1, 2 \ldots n.$$

$$x(u) = a_x u^3 + b_x u^2 + c_x u$$

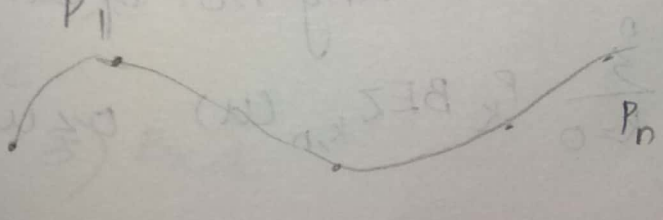
$$y(u) =$$

$$z(u) =$$

$$0 \le u \le 1$$

Types of cubic spline:

1) Natural Cubic spline:



2) Hermite Interpolation:

$P(0) = P_k$; $P(1) = P_{k+1}$; $P'(0) = DP_k$

$P'(1) = DP_{k+1}$

3) a Cardinal Spline : piece wise cubics

$P(0) = P_k$; $P(1) = P_{k+1}$;

$P'(0) = \frac{1}{2}(1-t)(P_{k+1} - P_{k-1})$

$P'(1) =$

4) Kochanek - Bartels Splines

$P(0) =$

$P(1) =$

$P'(0) =$

$P'(1) =$

Bezier

~~Later~~ Curves and Surfaces:

(approximation method)

Surf design, CAD systems ⇒ use

adv can be fitted to any no of ctrl pts

$$P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u) \quad 0 \le u \le 1$$

Bezier Blending func,

$$BEZ_{K,n}(u) = C(n,K)\,u^K\,(1-u)^{n-K}$$

$$C(n,K) = \frac{n!}{K!\,(n-K)!}$$

Types :

1. Cubic Bezier Curve

$n=3$; $BEZ_{n,3}(u) = (1-u)^3$

Bezier Surface :

$$P(u,v) = \sum_{j=0}^{m} \sum_{K=0}^{n} P_{j,K}\, BEZ_{j,m}(v)\, BEZ_{K,n}(u)$$

B-Spline (Bezier spline)

advantages
1. deg of B-spline polyn is ——
2. local

B-Spline Curves

$$P(u) = \sum_{K=0}^{n} P_K\, B_{K,d}$$ (ref book)

Properties of B-spline curve

1) deg is $d-1$; continuity is C^{d-2}

2) $(n+1)$ ctrl pts will have $(n+1)$ blending func.

3) $B_{K,d}$ ⇒ knot value

4) $n+d$ $n+d+1$

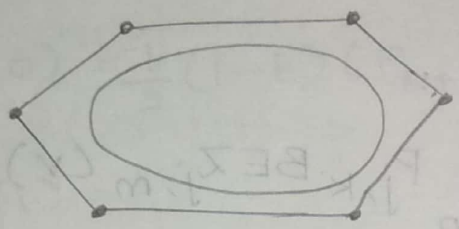
B-spline Types

① Uniform periodic B-splines

knot value, knot vector

② Uniform Quadratic B-spline

③ Cubic periodic B-spline → closed curve.



P(0) P(1) P'(0) P'(1)

④ Open, uniform B-splines

⑤ Non - Uniform B-spline

flexibility curve shape

B-spline surfaces

① Beta - splines (generalization of B-spline surf)

$$P_{j-1}(U_j) = P_j(U_j)$$

$B_1 = 1$; $B_1 >> 1$ →

Rational Splines : ratio of 2 spline

$$P(u) = \sum_{k=0}^{n} w_k P_k B_{k,d}(u)$$ Book

adv model the curves, shapes

Displaying Spline curves and Surfaces

3 methods

1. Horner's Rule : successive factoring

$$x(u) = a_x u^3 + b_x u^2 + c_x u + 1.$$

2. Forward Difference calculation (Δ)

$$x_{k+1} = x_k + \Delta x_k$$

forward difference $x_0, \Delta x_0, \Delta^2 x_0$

3. Sub division method

before subdivision

after subdivision

book