Geometric

2D. Graphic transformation : 8M or 13 M.

The basic geometric transformations are

* translation
* rotation
* scaling (change in shape).

And other transformations are reflection and shear

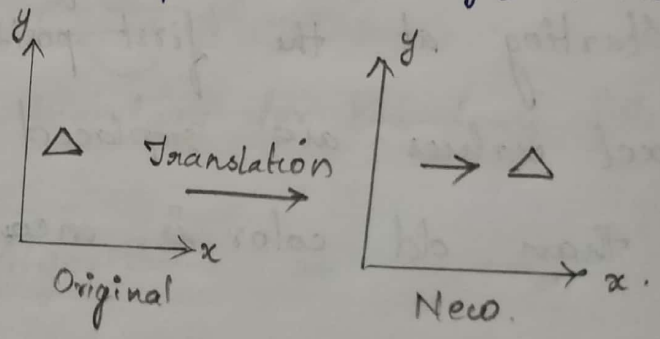
Translation :

Translation is applied to an object by repositioning it along a straight line path from one coordinate location to another.

Translation distance :

t_x, t_y

Original cordinate position (x, y), new position (x', y')



$$x' = x + t_x.$$

$$y' = y + t_y$$

The translation distance pair (t_x, t_y) is called translation vector or shift vector.

$$P = \begin{bmatrix} x, \end{bmatrix}$$

$$P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \qquad T = \begin{bmatrix} tx \\ ty \end{bmatrix} \qquad P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

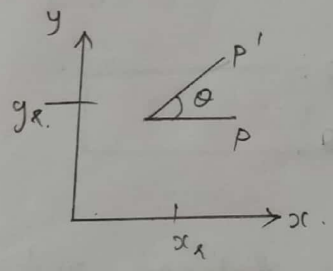
$$\boxed{P' = P + T.}$$ – translation equation.

2D translation equation in matrix form is $P' = P + T$.

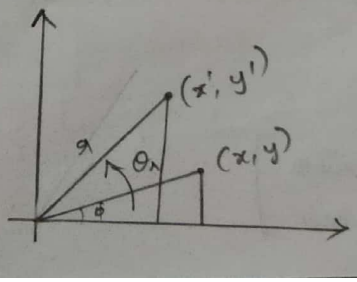
Translation is a rigid body transformation that moves object without deformation. (i.e) Every point on the object is translated by the same amount.

Rotation :

A 2D rotation is applied to an object by repositioning it along a circuit path in x, y plane. To generate a rotation we specify a rotation angle θ and position (x_r, y_r) of the rotation point or pivot point about which the object is to be rotated.



Positive values for the rotation angle defines the counter clockwise rotation about the pivot point. And negative values rotates the object in clockwise direction.



r - constant distance of the point from the origin

ϕ - Original Angular position of the point from the horizontal

θ - Rotational Angle.

$$x' = r\cos(\phi + \theta)$$

$$x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta \rightarrow (1)$$

$$y' = r\sin(\phi + \theta)$$

$$y' = r\cos\phi\sin\theta + r\sin\phi\cos\theta \rightarrow (2).$$

Original cordinates of the point in polar coordinates are

$$x = r\cos\phi \; ; \; y = r\sin\phi \rightarrow (3).$$

Using (3) in (1) & (2)

$$x' = x\cos\theta - y\sin\theta.$$
$$y' = x\sin\theta + y\cos\theta.$$

$$\boxed{P' = R.T}$$ R - Rotation.

where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$(P')^T = (R.P)^T$$
$$= P^T.R^T$$

where $$R^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

For rotation matrix R transpose is obtained by simply changing the sign of sine term.

Rotations are rigid body transformations that moves the object without deformation. Every point on an object is rotated through the same angle.

Scaling:

Scaling transformation alters the size of an object.

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

S_x, S_y - Scaling Factor.

Scaling factor S_x scales the objet in x direction and S_y scales the object in y-direction.

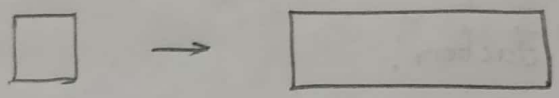
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\boxed{P' = S \cdot P}$$ where S is 2x2 scaling matrix.

Any positive numeric values can be assigned to scaling factors S_x, S_y. The values less than 1 reduces the size of object and values greater than 1 produce an enlargement. The value of 1 for both S_x and S_y leaves the size of object unchanged.

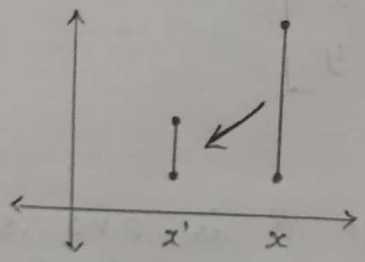
27:01:2018

When S_x and S_y are assigned the same value a uniform scaling is produced that maintains relative object proposition. Unequal values for S_x and S_y result in a differential scaling that is often used in design application where pictures are constructed from few basic shapes that can be adjusted by scaling and positioning transformation.



$S_x = 2$ $S_y = 1$ (differential scaling)

if $S_x = S_y$, it is called uniform scaling.



$S_x = S_y = 0.5$.

A line is scaled with $p = \frac{1}{2}$. p. using $S_x = S_y = 0.5$ is reduced in size and move closer to the coordinate origin.

We can control the location of scaled object by choosing a position called fixed point. (i.e) to

remain unchanged after the scaling transformation.

$$x' = x_f + (x - x_f) S_x$$

$$y' = y_f + (y - y_f) S_y.$$ // 2D transformation

// not in syllabus

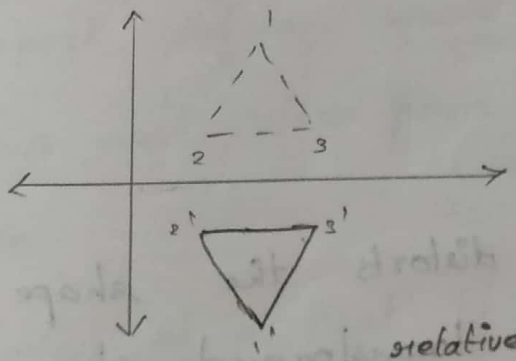
Composite transformation:

Forming product of transformation matrices is often referred to as concatenation or composition of matrices. /

OTHER TRANSFORMATION:

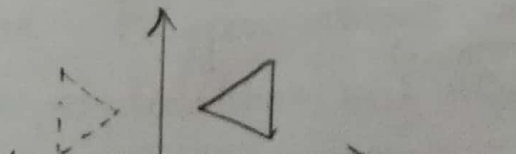
REFLECTION:

Reflection is a transformation that produces a mirror image of an object. The mirror image for 2D



relative

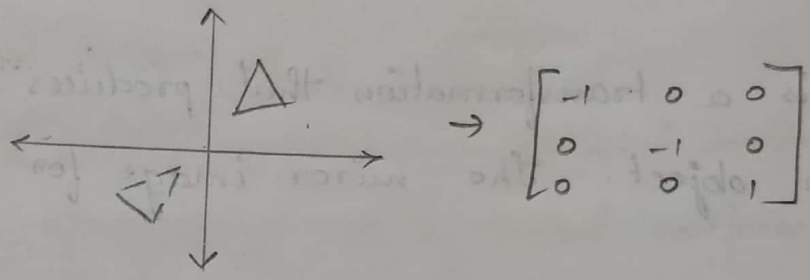
reflection is generated to an axis of generation. reflection by rotating the object 180° about the reflection axis. Reflection of an object about x-axis is shown in the above diagram.



Reflection about the line $y=0$, the x-axis is accomplished with the transformation matrix

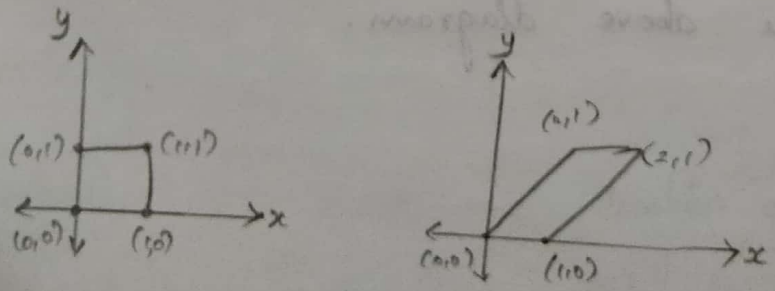
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ this transformation keeps the x-value the same but flips the y-value of the coordinate position.

A reflection about y-axis flips x-coordinate while keeping the y-coordinate the same. The matrix for this transformation is $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & +1 \end{bmatrix}$ (x-value, y-value, z-value).



SHEAR

A transformation that distorts the shape of an object such that the transformed shape appears as if the objects were composed of internal layers that had been caused to slide over each other is called shear.



A unit square in fig. 1 is converted to

parallelogram B using x-direction shear matrix.

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ with $sh_x = 2$.

An x-direction shear is related to produce a transformation matrix $\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ which transforms the coordinate position as $x' = x + sh_x \cdot y$

$y' = y$.

2DIMENSIONAL VIEWING: 2M.

A world coordinate area selected for display is called window. An area on a display is called device to which a window is mapped view port.

Window defines what is to be viewed.

Viewport defines where is to be displayed.

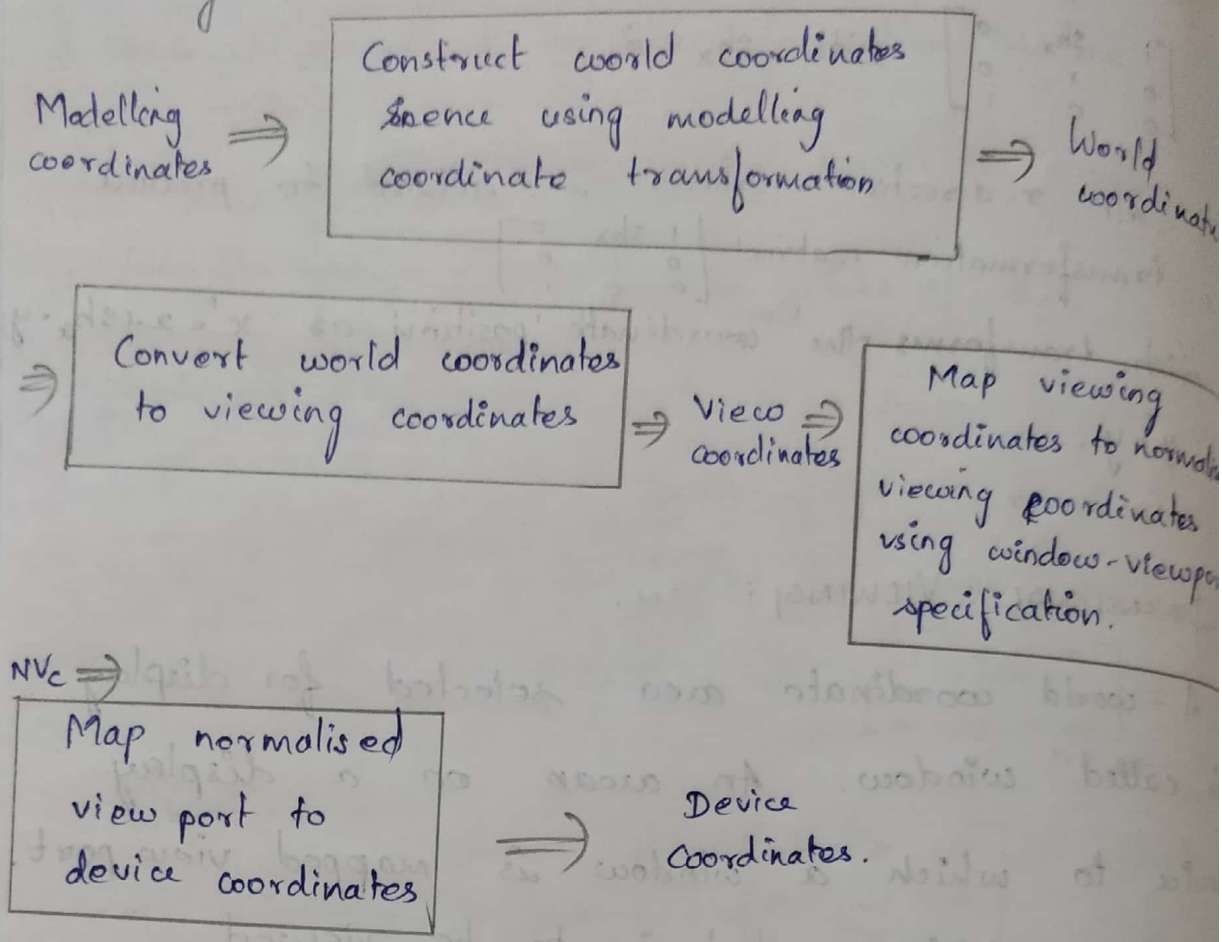
Windows and view ports are rectangles in standard position.

Mapping of a part of world cordinate scene to device cordinate is referred to as viewing transformation or view window - to- view port transformation. or windowing transformation

NOTE: Polygon shapes and circles are used in some application but these shapes takes longer to process.

2D VIEWING TRANSFORMATION PIPELINE : 2m.

Diagram.



View ports are typically defined within the unit square (normalised coordinates). This provides a means for seperating the viewing and other transformation from specific o/p device requirements, so that the graphic packages are largely device independant.

Different o/p devices can be used by providing the appropriate device drivers.

Clipping procedures are of fundamental importance in computer graphics. They are used not only in viewing transformation but also in

window - managing system. In painting and drawing packages to eliminate parts of a picture inside or outside of a designed screen area and in many other applications.

CLIPPING OPERATION: 2m. Alg - 13 m.

Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as clipping algorithm (or) clipping. The region against which an object is to be clipped is called a clipped window.

APPLICATIONS OF CLIPPING:

* Extracting part of a defined scence for viewing.
* Identifying visible surfaces in 3D views.
* Antialiasing the line segments or object boundaries.
* Creating objects using solid modeling procedures
* Displaying a multi window environment
* Drawing and painting operations. that allow part of a picture to be selected for copying, moving, erasing or duplicating.

For viewing transformation, we want to display only those picture parts that are within window area (assuming that the clipping flags have not been set to no clip) everything outside the

window is discarded.

Types of Clipping:

1. Point Clipping
2. Line Clipping (straight line segment).
3. Area Clipping (Polygon).
4. Curve Clipping
5. Text Clipping

Line and polygon clipping are standard components of graphics packages.

31:01:2018

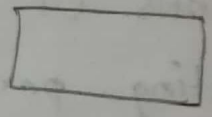
POINT CLIPPING:

Clipping window is rectangle shape

$P = (x, y)$

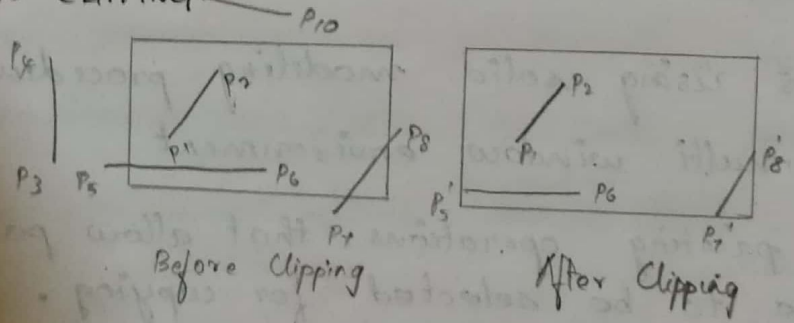
$x_{wmin} \leq x \leq x_{wmax}$

$y_{wmin} \leq y \leq y_{wmax}$



Used to depict explosion, sea foam, sea particles.

LINE CLIPPING:



Before Clipping

After Clipping

Parametric representation: (x_1, y_1) (x_2, y_2)

$$x = x_1 + u(x_2 - x_1)$$

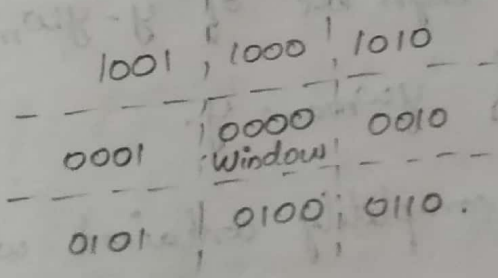
$$y = y_1 + v(y_2 - y_1)$$

$$0 \leq v \leq 1$$

1. Take a line whether the line is inside or outside
2. If it is neither inside nor outside then intersection are follow inside-outside test.
3. If it is outside the window, discard the line

COHEN SUTHERLAND LINE CLIPPING ALGORITHM: 13 marks diagram

This is one of the oldest most popular line clipping algorithm. Generally the method speeds up the processing of line segment by performing the initial test to reduce the no. of intersection that must be calculated. Every line endpoint in a picture is assigned. A four digit binary code called region code that identifies the location of the point relative to the boundaries of clipping rectangle.



Each bit position in the region code is used to indicate one of the four relative coordinate position of the point with respect to clip window left, right, top, bottom.

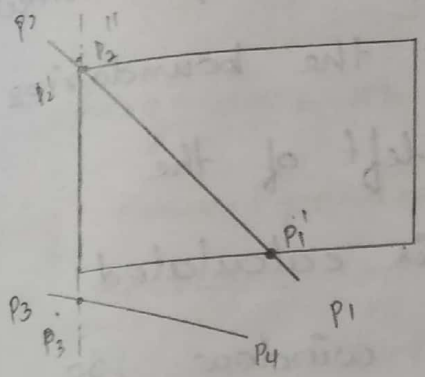
Bit-1 - LEFT
Bit-2 - RIGHT
Bit 3 - bottom
Bit 4 - TOP

The value of 1 in any bit position indicates that the point is in the relative position otherwise the point is set to 0. If the point is within the clipping rectangle then the region code is 0000. A point that is below and to the left of rectangle has a region code of 0101.

Bit values in the region code are determined by comparing endpoint coordinate value (x, y) to the clipping boundaries. Bit-one is set to 1 is less than xw_{min} and the other 3 bit values can be determined using similar comparison. Bit-1 is sign bit of $x - xw_{min}$, bit-2 is sign bit of $xw_{max} - x$, bit-3 is sign bit of $y - yw_{min}$ and bit-4 is sign bit of $yw_{max} - y$.

Any lines that are completely contained within the window boundary have a region code 0000 for both endpoints, we accept these lines. Any lines that have 1 in same bit position in the region code for each end point are completely outside the clipping rectangle, we reject these lines. We would discard the line that has a region code of 1001 for one endpoint and code of 0101 for other endpoint.

Both endpoints of this line are left of clipping rectangle, as indicated by 1 in bit-1 of each region code. Logical AND operation is performed with both the region codes to test the line. If the result is not 0000, then the line is completely outside the clipping region.



Lines that cannot be identified as completely inside or completely outside in the clip window are checked for intersection with the window boundary as shown in above figure. Such lines may or may not cross into window interior. Clipping process for a line is done by comparing a outside endpoint to a clipping boundary to determine how much of line can be discarded.

Cohen Sutherland algorithm is processed with the above diagram, starting with bottom endpoint P_1 to P_2, we check P_1 against left, right and bottom boundaries inturn find the this point is below the clipping rectangle.

We then find the intersection point P_1' with the bottom boundary and discard the line section. Discard the line section from P_1 to P_1'. The line now has been reduced to the section P_1' to P_2. Since P_2 is outside the clip window we check this endpoint against the boundaries and find that it is the left of the window. Intersection point P_2' is calculated but this point is above the window. So the final intersection calculation is P_2''. Line from P_1' to P_2'' is saved. This completes the processing for this line so we save this part and go on to next line. Point P_3 in the next line is left of the clipping rectangle so we determine the intersection P_3' and eliminate P_3 to P_3'. By checking region codes for the line section P_3' to P_4, we find that the remainder of the line is below the clip window so can be discarded also.

Algorithim : nuut.

LIANG- BARSKIN LINE CLIPPING ALGORITHM: 8M or 8M

Faster line clippers have been developed that are based on the analysis of parameteric equation of a line segment

$$x = x_1 + u\Delta x.$$
$$y = y_1 + u\Delta y \quad , \quad 0 \leq u \leq 1.$$

where $\Delta x = x_2 - x_1 \quad \Delta y = y_2 - y_1$

Using these parameteric equation, cyrus and beck developed an algorithm that is generally more efficient that cohen sutherland alggorithm. later Liang - Barskin independantly deviced a faster line clipping algorithm. We first write the point clipping conditions in parameteric form

$$xw_{min} \leq x_1 + u\Delta x \leq xw_{max},$$
$$yw_{min} \leq y_1 + \Delta y\, u \leq yw_{max}.$$

Each of these 4 inequalities can be expressed as $up_k \leq q_k$, $k = 1, 2, 3, 4$ where parameteres p and q are defined as $p_1 = -\Delta x$, $p_2 = \Delta x$, $p_3 = -\Delta y$, $p_4 = \Delta y$, $q_1 = x_1 - xw_{min}$,

$$q_2 = xw_{max} - x_1,$$
$$q_3 = y_1 - yw_{min},$$
$$q_4 = yw_{max} - y_1.$$

Any line that is parallel to one of the clipping boundary as $p_k = 0$ for the value of k

of boundary ($k = 1, 2, 3, 4$)

corresponding to left, right, bottom and top boundary respectively. For value of k we also find $q_k < 0$ then the line is completely outside the boundary and can be eliminated further consideration. If $q_k \geq 0$ the line is inside the parallel clipping boundary. If $q_k \geq 0$. $p_k < 0$, infinite extension of line proceeds from outside to inside of the infinite extension of this particular clipping boundary. If $p_k > 0$, the line proceeds from inside to outside. For a non-zero value of p_k, we can calculate the value of u that corresponds to the point where the infinitely extended line intersect the extension of the boundary k as $u = q_k / p_k$.

05:02:2018

For each line we can calculate values for parameters u_1 and u_2 that define a part of the line that lies within the clip rectangle. The value of u_1 is determined by

looking
line proc
these edges
Value
consisting
u_2 is dete
which li
Value of
boundaries
of the set
If u_1
the clip
otherwise,
calculated
line clippi
1
0.
cas
u
Algori
C

looking at the rectangle edges for which the line proceeds from outside to inside ($P < 0$). For these edges we calculate $R_k = q_k / P_k$.

Value of U_1 is taken as largest of the set consisting of 0 and various values of R. Value of U_2 is determined by examining the boundaries for which line proceeds from inside to outside ($P > 0$). Value of R_k is calculated for each of these boundaries and the value of U_2 is the minimum of the set consisting of 1 and calculated R values.

If $U_1 > U_2$ the line is completely outside the clip window and it can be rejected otherwise, the endpoints of the clipline is calculated from 2 values of the parameter U.

Line clipping Algo.

1 inequalities.

$Q_1, Q_2, P_1, P_2, P_3, P_4$.

case - (A).

$U = q_k / P_k$.

Algorithm:

Clip Test.

2M: Difference b/w cohen sutherland & Liang Barskin.

Liang - Barskin algorithm is more efficient than cohen sutherland algorithm since intersection calculations are reduced. Each update of parameters U_1 and U_2 requires only one division and window intersection of the lines are computed only once, when the final values of U_1 and U_2 have been computed.

In constrast cohen sutherland algorithm ~~Nicholl~~ repeatedly calculate the intersection along the line path. Eventhough the line maybe completely outside the clip window. And each intersection calculation requires both division and multiplication.

NOTE: Cohen Sutherland & Liang - Barskin algorithm can be extended to 3 dimensional clipping.

not for exam

Nicholl - Lee - Nicholl Line Clipping: (Among the 3, this is the best algorithm). If supports only 2D.

By creating more regions around the clip window, NLE avoids multiple clipping of an individual line segment.

Compared to both Cohen Sutherland and Liang Barskin algorithm, NLN performs fewer comparisons & division. NLN algorithm

can only be applied to 2D clipping whereas Liang-Barshien & Cohen Sutherland can be easily extended to 3D scenes. // not for exam.

05 : 02 : 2018

UNIT-III THREE DIMENSIONAL CONCEPTS

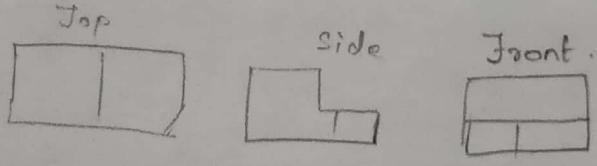
Viewing transformation in 3D is much more complicated because we have many more parameters to select when specifying how a three dimensional scence is to be mapped to a display device.

3D DISPLAY METHOD: Ex: Camera

PARALLEL PROJECTION

One method for generating a view of solid object is to project points on the object surface along parallel lines onto the display planes by selecting different viewing positions, we can project select visible points on the object onto the display plane to obtain different 2D views of the object.

Ex: Victory Stand. (Top, front, side view - 2D)



In a parallel projection, parallel lines in the world cordinates scence project into parallel lines on the 2D display frame. This technique is used in engineering & architectural

drawings with set of views to maintain relative proposition of the object. The appearance of the solid objects can then be reconstructed from the major views

07:02:2018

PRESPECTIVE PROJECTION:

Another method for generating a view of 3D scence is to project points to the display plain along converging part. This causes the objects further from the viewing position to be displayed smaller than the objects of the same size that are nearer to the viewing positions.

Ex: Airport.

In a perspective projection, the parallel lines in a scence that are not parallel to the display plane are projected into converging lines. Scences displayed using prespective projection appears more realistic since this is the way our eyes or camera lens forms images. Distant object appear smaller than objects closer to the viewing position.

DEPTH VIEWING: CUEING :

Depth information is important so that we

direction which is front or back of displayed objects. A simple method for indication depth with wire frame display is to vary the intensity of the object according to the distance from the viewing position. The lines closet to the viewing position are displayed with highest intensity and lines further away are displayed with decreasing intensity. Depth cueing is applied by choosing max and min intensity (or color) values and a range of distance over which intensities are to vary.

Application Of Depth Cueing:

Application of depth cueing is modelling the effect of atmosphere on the persued intensity of the object. More distant objects appear dimmer to us than the nearer objects due to light scatering by dust particles Haze and smoke. Some atmospheric effects can change the persued color of an object and we can model these effects with depth cueing.

12:02:2018

VISIBLE LINE & SURFACE IDENTIFICATION:

We can also clarify the depth relationship in a wireframe display by identifying visible lines in some way. the simplest method is to highlight the visible lines or to display them in a different color.

Another technique commonly used for engineering drawing is to display the non-visible lines as DASH lines. Another approach is to simply remove the non-visible lines. But removing the hidden lines also removes information about the shape of the back surfaces of an object. These visible line methods also identify the visible surfaces of the object.

When objects are to be displayed with color (or) shaded surfaces, we apply surface-rendering procedures to the visible surfaces so that the hidden surfaces are obscured.

Surface Rendering:

Added realism is attained in displays by setting the surface intensity of objects according to lightning conditions in the same and according to attained surface characteristics.

Lighting specification include intensity and position of light sources and general background illumination required for a scence.

Suface properties of object include degree of transpareny or how rough or how smooth the surface should be. Suface rendering methods are combined with perspective visible - suface identification to generate a degree of realism ~~to~~ in a display scene. scene.

Exploded and Cut away views:

Many graphics packages allow objects to be identified as hierarchical structures so that the internal details can be stored. Exploded and cut-away views of such objects can then be used to show the internal structure and relationship of object parts. Ex: Motor Engine.

An alternative to exploring the objects into its component part is the cut-away view which removes part of visible surfaces to show internal structures.

3D STEREOSCOPIC VIEWS:

Another method for adding a sense of reallism to a computer generated scene is to display

object using either 3D or stereoscopic views. 3D views can be obtained by reflecting a raster image from a vibrating flexible mirror. Vibration of mirror are synchronised with the display of scenes on the CRT. As the mirror vibrates the focal length varies so that each point in the scene is projected to a position corresponding to its depth.

Stereoscopic devices present two views of a scene, one for the left eye and the other for the right eye. The two views are generated by selecting viewing position that correspond to the two eye position of a single viewer. These two views then can be displayed on an alternate refresh cycles of a raster monitor and viewed through glasses that alternately draken first one lens and then the other in synchroni-zation with the monitor refresh cycles. // Any 4 methods 2 mark

3D OBJECT REPRESENTATION:

Representation schemes for solid objects are often divided into two board categories all though not all representations fall neatly into one or other of these two categories

Boundary Representation (B-REPS)

Space Partitioning Representation.

14:02:2018

Polygon surfaces: 8m

The most commonly used boundary representation for a 3D graphic object is a set of surface polygons that enclose the object interior. Many graphic system store all object descriptions as a set of surface polygon.

Reason for using polygon:

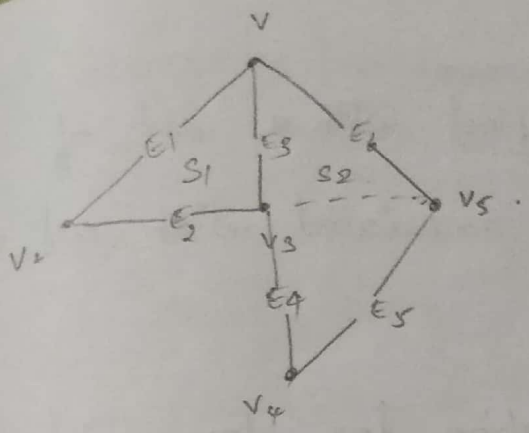
This surface polygon simplifies and speeds up the surface rendering and display of objects since all surfaces are described with linear equation, for this reason polygon descriptions are often referred to as "Standard Graphics Objects".

NOTE:

Polygonal representation is the only one available but many packages allow objects to be described with other schemes such as spline surfaces that are then converted to polygonal representations for processing.

POLYGON TABLE:

Polygon data tables can be organised into two groups (1) Geometric tables (2) Attribute tables.



(1)

VERTEX - TABLE
V_1 - x_1, y_1, z_1
V_2 - x_2, y_2, z_2
⋮
V_5 - x_5, y_5, z_5

(2)

EDGE TABLE
E_1 : V_1, V_2
E_2 : V_2, V_3
E_3 : V_3, V_1
E_4 : V_4, V_3
E_5 : V_4, V_5
E_6 : V_5, V_1

(3)

POLYGON SURFACE TABLE
S_1 : E_1, E_2, E_3
S_2 : E_3, E_4, E_5, E_6

(4) Edge table for surfaces of the above digram expanded to include pointers to the polygon table

E_1 : V_1, V_2, S_1
E_2 : V_2, V_3, S_1
E_3 : V_3, V_1, S_2
E_4 : V_4, V_3, S_2
E_5 : V_4, V_5, S_2
E_6 : V_5, V_1, S_2

Geometric data table representation with two adjacent surfaces, 6 edges and 5 vertices.

Geometric datal table contain vertex coordinates and parameters to identify spatial orientation of the polygon surfaces.

Attribute information of an object includes parameters specifying the degree of transparency of the object and its suface reflextivity and texture and characteristics.

NOTE:

Polygon suface is specified with a set of vertex coordinates and associated with set of parameters.

A convinient organisation for storing geometric data is to create 3 list.

(1) Vertex Table
(2) An edge Table
(3) Polygon Table.

Coordinate values for each vertex in the object are stored in vertex table. Edge table contains pointers back into vertex table to identify the vertices for each polygon edge. And polygon table contains pointers back into the edge table to identify edges for each polygon. This scheme is illustrated in the above diagram which uses two adjacent polygons on an object surface.

Listing the geometric data in three tables as in above diagram provides a convenient reference to the individual components (vertices, edges, polygons) of each object. Also the object can be displayed efficiently by using the data from the edge table to draw the component lines.

An alternative arrangement is to use, just two tables a vertex table and a polygon table. But this scheme is less convenient and some edges could get drawn twice.

Another posibility is to use only a polygon table but this duplicates coordinate information. since explicit coordinate values are listed for each vertex in each polygon. Also edge information has to be reconstructed for the vertex listing in the polygon table.

Since geometric data table may contain extensive listings of vertices and edges for complex objects. It is important for the data being checked for the consistency and completeness. More information included in data tables, by easier it is to check for errors. Therefore error checking is easier when three (3) data tables are used (Vertex, edge, polygon Table). Since this scheme provides the most information some of the test could be performed by a graphic package are

1. That every vertex is listed as an endpoint for atleast two edges.

(2) That every edge is part of atleast one polygon.

(3) That every polygon is closed

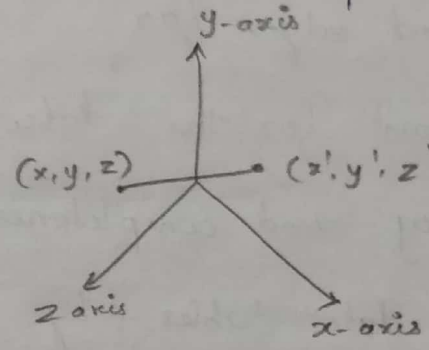
(4) Each polygon has atleast one shared edge.

(5) If edge table contains pointers to polygon, every edge is referenced by a polygon pointer. As a reciprocal pointer back to the polygon.

19:02:2018

3D GEOMETRIC AND MODELING TRANSFORMATION: 13m

Translation : In 3D homogenous coordinate representation, a point is translated from position



Translating a point with translation vector $T(t_x, t_y, t_z)$

$P = (x, y, z)$ to position $P' = (x', y', z')$ with matrix operation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow (1)$$

$$\boxed{P' = T \cdot P}$$

The parameters t_x, t_y, t_z specifying the translation distance for the coordinate direction are assigned any real values. In equation (1) matrix representation is equivalent to three equations

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

We obtain the inverse of the translation matrix in eq (1) by negation of the translation distance t_x, t_y, t_z. This produces a translation in the opposite direction. The product of the translation matrix and its inverse produces the identity matrix.

Rotation :

To generate a rotation transformation for an object we must designate an axis of rotation (about which the object is to be rotated) and the amount of angular rotation.

Coordinate Axis Rotation :

Easiest rotation axis to handle are those that are parallel to the coordinate axis.

2D z-axis rotation equations are easily extended to 3D.

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta \longrightarrow (1)$$
$$z' = z$$

The parameter θ specifies the rotation angle in homogenous coordinate form. The 3D z-axis rotation, the equations are expressed as

for x-axis rotation replace $x \to y$, $y \to z$, $z \to x$ in eq (1).

$$P' = R_z(\theta) \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformation equation for rotation about the other two coordinate axis can be obtained with a cyclic permutation of the coordinate parameters (x, y, z) in equation (1). We use the replacement

$x \to y \to z \to x$ - (2) substituting the permutation (2) in equation (1) we get the equation for x-axis rotation.

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_x(\theta) \cdot P$$

Rotation of object around y axis,

$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_y(\theta) \cdot P$$

An inverse rotation matrix is formed by replacing rotation angle θ by $(-\theta)$ negative values for rotation angle generate rotation in clockwise direction, so the identity matrix is produced when any rotation matrix is multiplied by its inverse.

Since only the sine function is affected by the change in the sign of the rotation angle, the inverse matrix can also be obtained by interchanging rows and columns

$$R^{-1} = R^T$$

Scaling :

The matrix expression for the scaling transformation $P = (x, y, z)$ relative to the coordinate origin can be written as $P' = S \cdot P$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow (1)$$

where the scaling parameters S_x, S_y, S_z are assigned any positive values.

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y \rightarrow (2)$$
$$z' = z \cdot S_z$$

Scaling an object with transformation with eq (1) changes the size of the object and repositions the object relative to the coordinate origin.

21:02:2018

Also, if the transformation parameter are not equal relative dimensions in the object are changed. We preserve the original shape of an object with a uniform scaling.

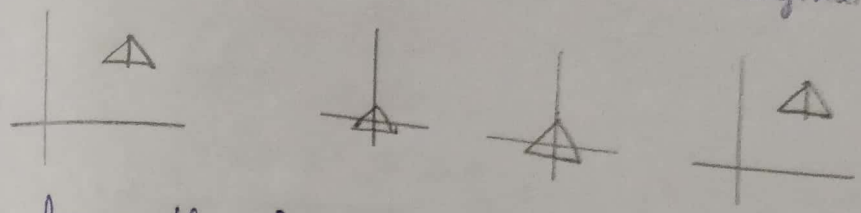
$$S_x = S_y = S_z$$

Scaling with respect to a selected fixed point position x_f, y_f, z_f can be represented with the following transformation segments.

(1) Translate the fixed point to the origin.

(2) Scale the object relative to the coordinate origin

3. Translate the fixed point back to its original position.



We form the inverse scaling matrix for equation(1) by replacing scaling parameters s_x, s_y and s_z with their reciprocals.

Inverse matrix generates an opposite scaling transformation so the concatination of any scaling matrix and its inverse produces the identity matrix.

Other transformations:

Reflection
Shear.

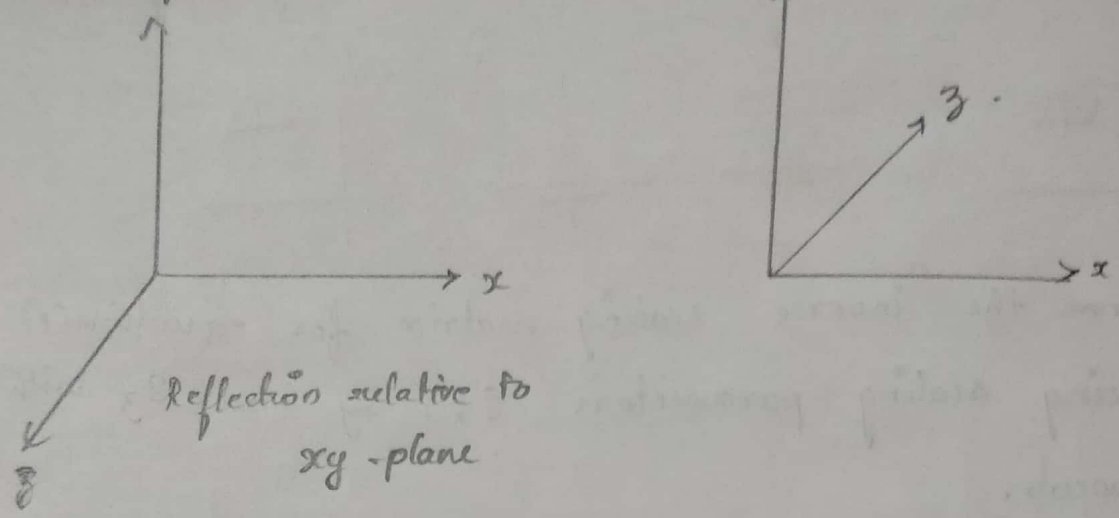
(1) Reflection:

A 3D reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane.

In general, three dimensional reflection matrices are set up similarly to those for two dimension.

Reflection relative to a given axis are equivalent to 180° rotation about that axis.

When the reflection plane is a coordinate plane (either (xy, yz or xz plane). We can think of transformation as a conversion between left handed and right handed system.



Reflection relative to xy-plane

An example of reflection that from a right handed system to left handed system or viceversa as showed in the above diagram.

This transformation changes the sign of the z-coordinate leaving x and y coordinate values unchanged.

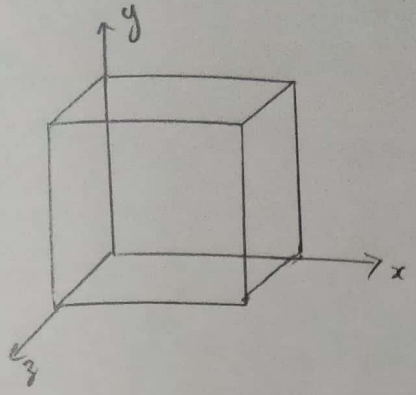
In matrix representation, of point relative to xy-plane is

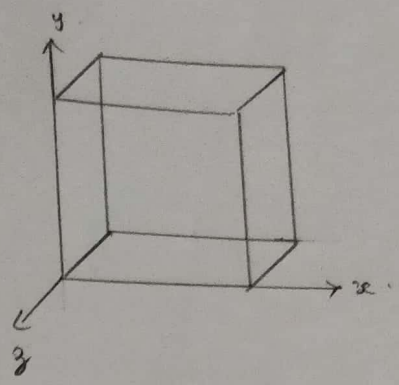
$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.) Shear:

Shearing transformations can be used to modify object shapes. They are also useful in three dimensional viewing for obtaining general projection transformation.

Examples of 3D shearing, the following transformation produces, z-axis shear.





$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parameters a and b can be assigned any real values, the effect of this transformation matrix is to alter x and y coordinate values by an amount that is proportional to the z-value while leaving the z-coordinate unchanged.

21:02:2018

PROJECTION:

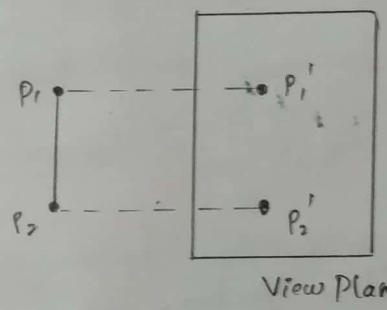
We can project three dimensional objects on to the two dimensional plane.

There are two basic projections methods

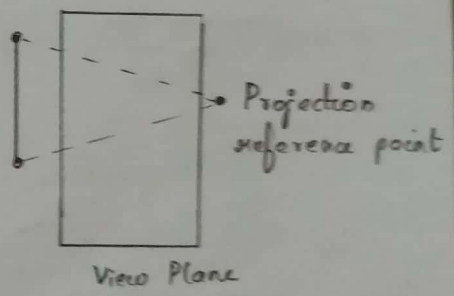
* Parallel Projection
* Perspective Projection

In a parallel projection, coordinate positions are transformed to the view plane along parallel lines.

In a perspective projection, object positions are transformed to the view plane along lines that converge to a point called projection reference point (centre for projection).



Parallel Projection



Perspective projection

Projected view of an object is determined by calculating the intersection of projection lines with the view plane.

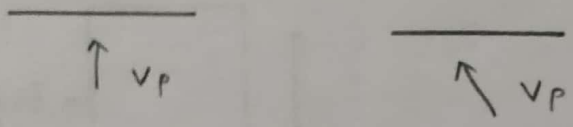
A parallel projection is used in drafting to produce scale drawings of three dimensional object.

Accurate views of various sides of an object are obtained with a parallel projection. But this does not give as a realistic representation of appearance of three dimensional object. But it preserves realistic proposition of object. Perspective projection produces realistic views relative propositions.

Parallel Projection:

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

When the projection is perpendicular to the view plane we have an orthographic parallel projection, otherwise we have oblique parallel projection.



Orthographic Oblique

V_P - view projection (or) Projection Vector.

Orthographic projections are most often used to produce front, side and top views of an object. Front, side and rear orthographic projections of an object are called elevation and top orthographic projection are called a plan view.

NOTE: Engineering and architectual drawings commonly employ these orthographic projections because length

and angles are accurately depicted can be measured from the drawings.

We can also form orthographic projections that display more than one face of an object. Such views are called axonometric orthographic projection.

The most commonly used axonometric orthographic projections is isometric projection. We can generate the isometric projection by aligning the projection plane. So that it intersect each coordinate axis in which the object is defined. (called principle axis) at the same distance from the origin.

Isometric projection is obtained by alligning the projection vector with the cube diagonal. There are eight positions, one in each octant for obtaining an isometric view.

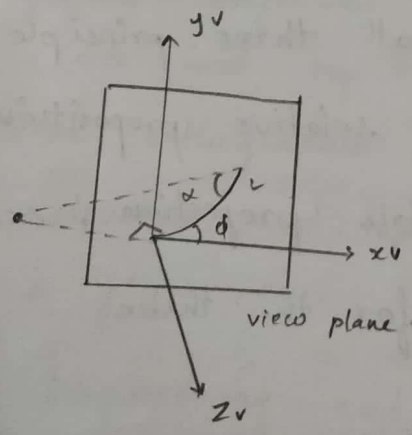
Difference between isometric projection and axonometric projection.

In isometric projection, all three principle axes are equally shortened so that relative propositions are maintained whereas in axonometric projection, scaling factors may be different for the three directions.

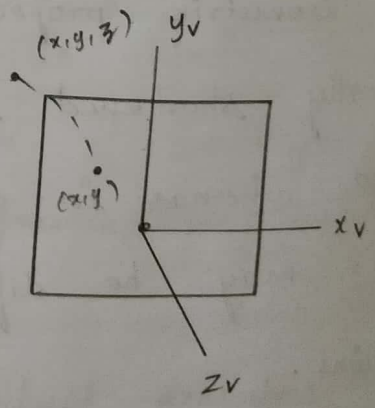
Transformation equation for an orthographic parallel projection if the viewplane is placed at position z_{vp} (vp - view point) along z_v axis. Then any point (x, y, z) in viewing coordinates is transformed to projection coordinates $x_v = x$; $y_v = y$ where the original z-coordinate value is preserved for depth information needed in depth cueing and visible surface determination procedures.

An oblique projection is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.

Two angles used in oblique projection vector is alpha (α) and (ϕ). The oblique projection (x, y, z) to (x_p, y_p) makes an angle α with the line on the projection plane that joins (x_p, y_p) and (x, y). This line of length L is at angle ϕ with horizontal direction in the projection plane.



Oblique projection of coordinate position (x, y, z) to (x_p, y_p) on the view plane



Orthographic projection of a point onto a viewing plane.

$$x_p = x + L\cos\phi$$
$$y_p = y + L\sin\phi \quad \rightarrow (1)$$

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

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

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

where $L_1 = \frac{1}{\tan\alpha}$ (i.e) inverse of $\tan\alpha$ which is also called the value of L when $z = 1$. Then equation (1) is written as $x_p = x + zL_1\cos\phi$

$$y_p = y + zL_1\sin\phi$$

∴ The transformation matrix for producing any parallel projection onto the x_v y_v 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 projection is obtained when $L_1 = 0$ (which occurs at projection angle α of 90°) oblique projection are generated with non-zero values of L_1.

NOTE : Projection Matrix to as a structure similar to that of a z-axis shear matrix.

Common choices for angle ϕ are 30° and 45° which display a combination view of front, side and top (or front, side and bottom) of an object.

Two commonly used values for α are those for which $\tan\alpha = 1$ or $\tan\alpha = 2$. For the first case $\alpha = 45^\circ$, views obtained are called cavalier projections. For $\tan\alpha = 2$ views obtained are called cabinet projections.

In cavalier projections, all lines perpendicular to projection plane are projected with no change in length.

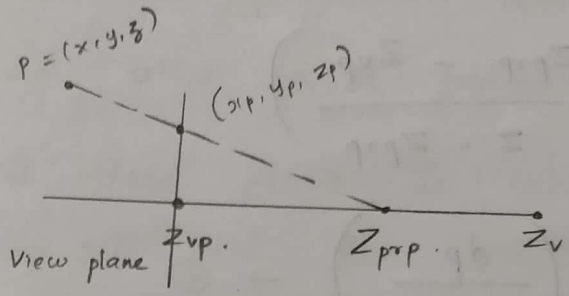
In cabinet projection, (approximately 63.4° angle) lines perpendicular to view surface are projected at ½ half of their length.

Cabinet projections appear more realistic than cavalier projections because of this reduction in the length of perpendiculars.

26 : 02 : 2018

PERSPECTIVE PROJECTION :

To obtain a perspective projection of 3D object we transform points along projection lines that meet up the projection reference point. Suppose we set the projection reference point at position z_{prp} along z_v axis (viewing direction is z axis). We place the view plane at z_{vp}



As shown in above diagram, z_{vp}, z_{prp}, z_v we can write equations describing coordinate position along this perspective projection line in parametric form as

$$x' = x - xu.$$
$$y' = y - yu.$$
$$z' = z - (z - z_{prp})u. \quad \rightarrow (1)$$

The parameter u takes values from 0 to 1 and coordinate position x', y', z' represents the any point along the projection line. When u=0, we are at position $P = (x, y, z)$.

When $u = 1$, we have projection reference point coordinate $(0, 0, z_{prp})$ on the view plane $z' = v_p$.

We can solve the z' equation for the parameter u at this position along the projection line.

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

Substituting this value of u into equations for x', y', we obtain perspective transformation equation

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

$$x_p = x \left(\frac{d_p}{z - z_{prp}} \right) \rightarrow (2)$$

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

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

where $d_p = z_{prp} - z_{vp}$ is the distance of view plane from the projection reference point. Perspective projection transformation equation (3) in matrix form is written as

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

Homogeneous factor $h = \frac{z - z_{prp}}{d_p}$ and the projection coordinates on the view plane are calculated from homogeneous coordinates.

$$x_p = x_h / h$$

$$y_p = y_h / h$$

where the original z-coordinate value would be retained in projection coordinates for visible surface and other depth processing.

If the view plane is taken to be uv-plane then $z_{vp} = 0$ and the projection coordinates are Substitute $z_{vp} = 0$ in eq (3).

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

$$y_p = y \left(\frac{z_{prp}}{z - z_{prp}} \right)$$

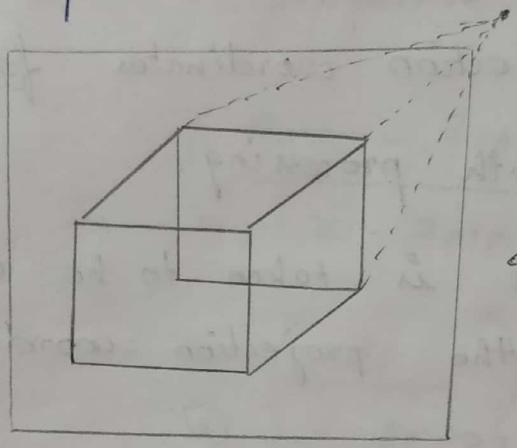
NOTE:

When a 3D object is projected onto a view plane using perspective transformation equation, any set of parallel lines in the object that are not parallel to the plane are projected into converging lines.

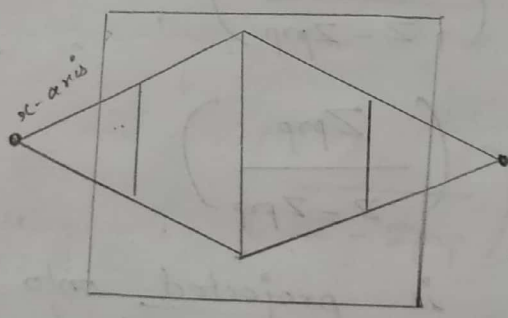
Parallel lines that are parallel to the view plane will be projected as parallel lines. The point at which set of projected parallel lines appears to converge ... point. 2M

A scene can have any no. of vanishing points, depending on how many sets of parallel lines in the scene.

Vanishing point for any set of lines that are parallel to one of the principle axis of the object is referred to as principle vanishing point.



one vanishing point
one point perspective projection



two point perspective projection
(2-vanishing point)
x-axis vanishing point
z-axis vanishing point

We control the no. of principle vanishing point (1, 2, or 3) with the orientation of the projection plane and perspective projection are classified as 1-point, 2-point, 3-point perspective projection.

Number of principle vanishing point in a projection is determined by the no. of principle axis intersecting the view plane.

28:02:2018

VISIBLE SURFACE REDUCTION METHOD:

Visible surface reduction algorithm is boardly classified into whether they deal with object definition directly or their projected images. These two approaches are called object-space methods and image-space methods.

NOTE:

Major consideration in generation of realistic graphic display is identifying those parts of a scene that are visible from a choosen viewing position.

2M // Object space method compares the objects and the parts of the object to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.//

2M. In a image space algorithm, the visibility is decided point by point in each pixel position on the projection plane.

Most visible surface algorithm use image-space method although object space method can be used to locate in some cases.

line display algorithm use object space method to identify visible lines in wire frame display but many image space visible surface algorithm can be adopted easily to visible line reduction.

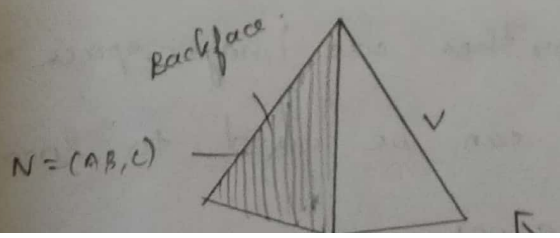
2M// There are major differences in the basic approach taken by various visible surface reduction algorithm Most used sorting and coherence methods to improve performance.

Sorting is used to facilitate depth comparisons by ordering the individual surfaces in a scene accordi to the distance from the view plane. Coherence methods are used to take advantages of regularit in a scene.

BACKFACE REDUCTION: 8M

A fast and simple object-space method for identifying the backfaces of polyhedron is based on the "inside-outside Jest". A point (x, y, z) is inside a polygon surface with a plain parameters A, B, C, D if

$$ax + by + cz + d < 0 \quad \rightarrow (1)$$



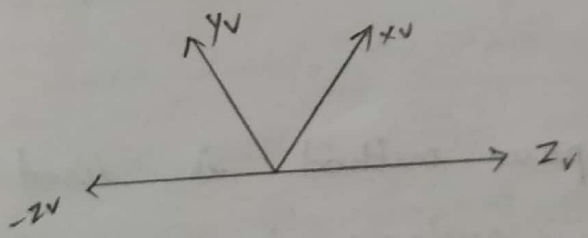
When an inside point is along the line of sight to the surface, the polygon must be a backface, we are inside that face and cannot see the front of it from our viewing position.

If V is a vector in the viewing direction from the eye (camera) position as shown in the above diagram, then this polygon is a backface if $V \cdot N > 0$ where N is normal vector.

If object description have been converted to projection coordinates and our viewing direction is parallel to z_v axis then $V = (0, 0, V_z)$ and

$V \cdot N = V_z C$.

In a right handed viewing system with viewing direction along negative z_v axis as shown in the diagram. The polygon is a backface if $C < 0$



we cannot see any face whose normal has z-component $C = 0$.

In general we can label any polygon as a backface if its normal vector as a z component value, $C \leq 0$

Left handed viewing system uses clockwise direction. Right handed viewing system uses counterclockwise direction.

By examining the parameter c for different planes defining an object. we can easily identify all the backfaces.

A general scene can be expected to contain overlapping objects along the line of sight. We then need to determine whether the obscured objects are partially or completely hidden by other objects. In general backface removal can be expected to eliminate about half of the polygon surfaces in a scene for further visibility test.

// no methods

Scan Line Method:

This image space method is used for removing image surfaces is an extension of scan line algorithm. For filling polygon interiors instead of filling just one surface we deal with multiple surface. As each scanline is processed all polygon surfaces intersecting that lines 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.

Edge table contains coordinate end points for each line in the scene, inverse slope of each line and pointers into the polygon table to identify the surfaces bounded by each line. Polygon table contains coefficients of the plane equation for each surface, intensity information for the surfaces and posibly pointers into the edge table.

To facilitate the search for surfaces processing, the given scan line, the active list of edges is introduced, this active list will contain only the edges that cross the current scan line, sorted in order of increasing x.

We define a flag for each surface, that is set on or off to indicate a position along a scan line whether inside or outside the surface. Scan lines are processed from left to right, at the left most boundary of the scan line the flag is turned on and on the right most boundary