UNIT – 3

2D TRANSFORMATIONS

Unit-03/Lecture-01

BASIC TRANSFORMATIONS

Changes in orientation, size, and shape are accomplished with geometric transformations that alter the coordinate descriptions of objects. The basic geometric transformations are translation, rotation, and scaling. Other transformations that are often applied to objects include reflection and shear.

TRANSLATION

A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another. We translate a two-dimensional point by adding translation distances, tx and ty, to the original coordinate position (x, y) to move the point to a new position (x', y').

$$x' = x + tx \quad , \quad y' = y + ty$$

The translation distance pair (tx, ty) is called a translation vector or shift vector.

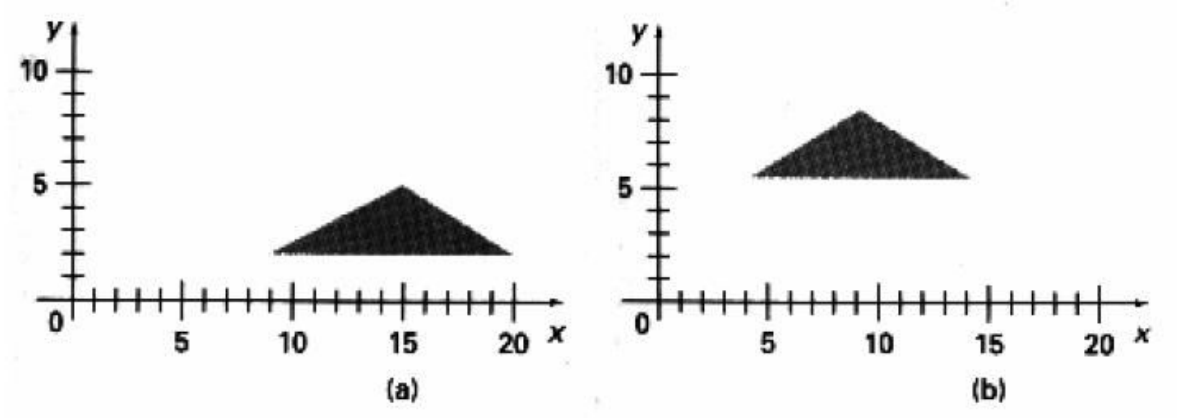
We can express the translation equations as a single matrix equation by using column vectors to represent coordinate positions and the translation vector.

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}, \qquad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

This allows us to write the two-dimensional translation equations in the matrix form:

$$P' = P + T$$

Sometimes matrix-transformation equations are expressed in terms of coordinate row vectors instead of column vectors. In this case, we would write the matrix representations as P = [x y] and T = [tx ty].



Translation is a rigid-body transformation that moves objects without deformation, i.e., every point on the object is translated by the same amount.

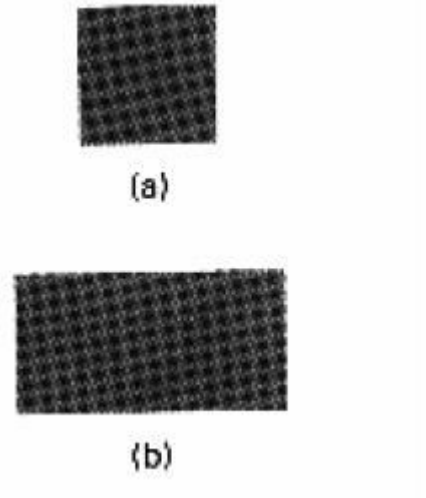
Polygons are translated by adding the translation vector to the coordinate position of each vertex and regenerating the polygon using the new set of vertex coordinates and the current attribute settings.

Similar methods are used to translate curved objects. To change the position of a circle or ellipse, we translate the center coordinates and redraw the figure in the new location. We translate other curves (splines) by displacing the coordinate positions defining the objects, and then we reconstruct the curve paths using the translated coordinate points.

SCALING

A scaling transformation alters the size of an object. This operation can be carried out for polygons by multiplying the coordinate values (x, y) of each vertex by scaling factors sx and sy to produce the transformed coordinates (x', y'):

$$x' = x \cdot s_x \quad , \qquad y' = y \cdot s_y$$



Scaling factor sx, scales objects in the x direction, while sy scales in the y direction. The transformation equations can be written in the matrix form as,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = S \cdot P$$

where S is the 2 by 2 scaling matrix.

- Specifying a value of 1 for both sx and sy leaves the size of objects unchanged.
- When sx and sy are assigned the same value, a uniform scaling is produced that maintains relative object proportions.
- Unequal values for sx and sy result in a differential scaling that are often used in design applications, when pictures are constructed from a few basic shapes that can be adjusted by scaling and positioning transformations.

We can control the location of a scaled object by choosing a position, called the fixed point that is to remain unchanged after the scaling transformation.

Coordinates for the fixed point (xf , yf) can be chosen as one of the vertices, the object centroid, or any other position. A polygon is then scaled relative to the fixed point by scaling the distance from each vertex to the fixed point. For a vertex with coordinates (x, y), the scaled coordinates (x', y') are calculated as,

$$x' = x_f + (x - x_f)\, s_x \quad , \qquad y' = y_f + (y - y_f)\, s_y$$

We can rewrite these scaling transformations to separate the multiplicative and additive terms:

$$x' = x \cdot s_x + x_f (1 - s_x)$$
$$y' = y \cdot s_y + y_f (1 - s_y)$$

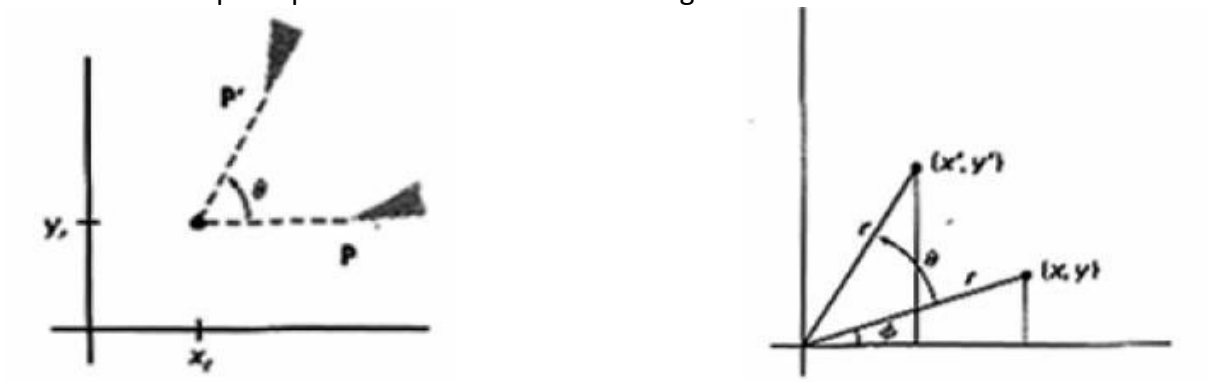
ROTATION

A two-dimensional rotation is applied to an object by repositioning it along a circular path in the xy plane. To generate a rotation, we specify a rotation angle 0 and the position (x, y) of the rotation point (or pivot point) about which the object is to be rotated.

- Positive values for the rotation angle define counterclockwise rotations.
- Negative values rotate objects in the clockwise direction.

This transformation can also be described as a rotation about a rotation axis that is perpendicular to the xy plane and passes through the pivot point.

We first determine the transformation equations for rotation of a point position P when the pivot point is at the coordinate origin. The angular and coordinate relationships of the original and transformed point positions are shown in the diagram.



In this figure, r is the constant distance of the point from the origin, angle is the original angular position of the point from the horizontal, and is the rotation angle.

Using standard trigonometric identities, we can express the transformed coordinates in terms of angles

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

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

The original coordinates of the point in polar coordinates are,

P'= R·P

where the rotation matrix is

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

S.NO	RGPV QUESTIONS	Year	Marks
1.	Derive a general 2D-Transformation matrix for rotation about the origin. perform a 45º rotation of a square having vertices- A(0,0),B(0,2),C(2,2),D(2,0) about the origin.	Dec, 2013	10
2.	Find the new coordinates of the triangle	Dec, 2011	10

	A(0,0),B(1,1) and C(5,2) after it has been • magnified to twice its size • Reduced to half its size.		

Unit-03/Lecture-02

Matrix Representation and Homogeneous Coordinates.

Matrix Representation and Homogeneous Coordinates.

Many graphics applications involve sequences of geometric transformations. An animation, for example, might require an object to be translated and rotated at each increment of the motion. In design and picture construction applications, we perform translations, rotations, and scalings to tit the picture components into their proper positions.

Each of the basic transformations can be expressed in the general matrix form

P' = M1 . P + M2

with coordinate positions P and P' represented as column vectors.

Matrix M1 is a 2 by 2 array containing multiplicative factors, and M2 is a two-element column matrix containing translational terms.

- For translation, M1 is the identity matrix.
- For rotation, M2 contains the translational terms associated with the pivot point. For scaling, M2 contains the translational terms associated with the fixed point.

To produce a sequence of transformations with these equations, such as scaling followed by rotation then translation, we must calculate the transformed coordinate one step at a time.

To express any two-dimensional transformation as a matrix multiplication, we represent each Cartesian coordinate position (x, y) with the homogeneous coordinate triple (xh, yh, h) where

$$x = \frac{x_h}{h} \ , \ y = \frac{y_h}{h}$$

Thus, a general homogeneous coordinate representation can also be written as (h.x, h.y, h). For two-dimensional geometric transformations, we can choose the homogeneous parameter h to be any nonzero value. A convenient choice is simply to set h = 1.

Each two-dimensional position is then represented with homogeneous coordinates (x, y, 1).

The term homogeneous coordinates is used in mathematics to refer to the effect of this representation on Cartesian equations.

When a Cartesian point (x, y) is converted to a homogeneous representation (xh, yh, h) equations containing x and y such as f (x, y) = 0, become homogeneous equations in the three parameters xh, yh and h.

Expressing positions in homogeneous coordinates allows us to represent all geometric transformation equations as matrix multiplications. Coordinates are represented with three-element column vectors, and transformation operations are written as 3 by 3 matrices.

For Translation,

we have

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

which we can write in the abbreviated form

P' = T (tx, ty) . P

with T (tx, ty) as the 3 by 3 translation matrix.

The inverse of the translation matrix is obtained by replacing the translation parameters tx and ty with their negatives – tx and – ty .

Similarly, Rotation Transformation equations about the coordinate origin are written as

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

$$\mathbf{P' = R\ (\theta)\ .\ P}$$

The rotation transformation operator R (□) is the 3 by 3 matrix with rotation parameter □. We get the inverse rotation matrix when □ is replaced with -□.
A Scaling Transformation relative to the coordinate origin is now expressed as the matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

P' = S (sx, sy) . P

where S (sx, sy) is the 3 by 3 matrix with parameters sx and sy.
Replacing these parameters with their multiplicative inverses (1/sx and 1/sy) yields the inverse scaling
matrix.
Matrix representations are standard methods for implementing transformations in graphics systems. Rotations and Scalings relative to other reference positions are then handled as a succession of transformation operations.
An alternate approach in a graphics package is to provide parameters in the transformation functions for the scaling fixed-point coordinates and the pivot-point coordinates.

COMPOSITE TRANSFORMATIONS

With the matrix representations, we can set up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transformations. Forming products of transformation matrices is often referred to as a concatenation, or composition, of matrices.
For column-matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order from right to left, i.e., each successive transformation matrix premultiplies the product of the preceding transformation matrices.

Translations:

If two successive translation vectors (tx1, ty1) and (tx2, ty2) are applied to a coordinate position P, the final transformed location P' is calculated as

P' = T (tx2, ty2) . {T (tx1, ty1) . P}

= {T (tx2, ty2) . T (tx1, ty1)} . P}

where P and P' are represented as homogeneous-coordinate column vectors.
Also, the composite transformation matrix for this sequence of translations is

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

T (tx2, ty2) . T (tx1, ty1) = T (tx1 + tx2 , ty1 + ty2)

which demonstrates that two successive translations are additive.

Rotations:

Two successive rotations applied to point P produce the transformed position

$$\mathbf{P}' = \mathbf{R}(\theta_2) \cdot \{\mathbf{R}(\theta_1) \cdot \mathbf{P}\}$$
$$= \{\mathbf{R}(\theta_2) \cdot \mathbf{R}(\theta_1)\} \cdot \mathbf{P}$$

so that the final rotated coordinates can be calculated with the composite rotation matrix as

$$\mathbf{P}' = \mathbf{R}(\theta_1 + \theta_2) \cdot \mathbf{P}$$

Scaling:

Concatenating transformation matrices for two successive scaling operations produces the following composite scaling matrix:

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1} \cdot s_{x2} & 0 & 0 \\ 0 & s_{y1} \cdot s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or

S (sx2, sy2) . S (sx1, s) = S (sx1 . sx2 , sy1 . sy2)

The resulting matrix in this case indicates that successive scaling operations are multiplicative.

General Pivot-Point Rotation:

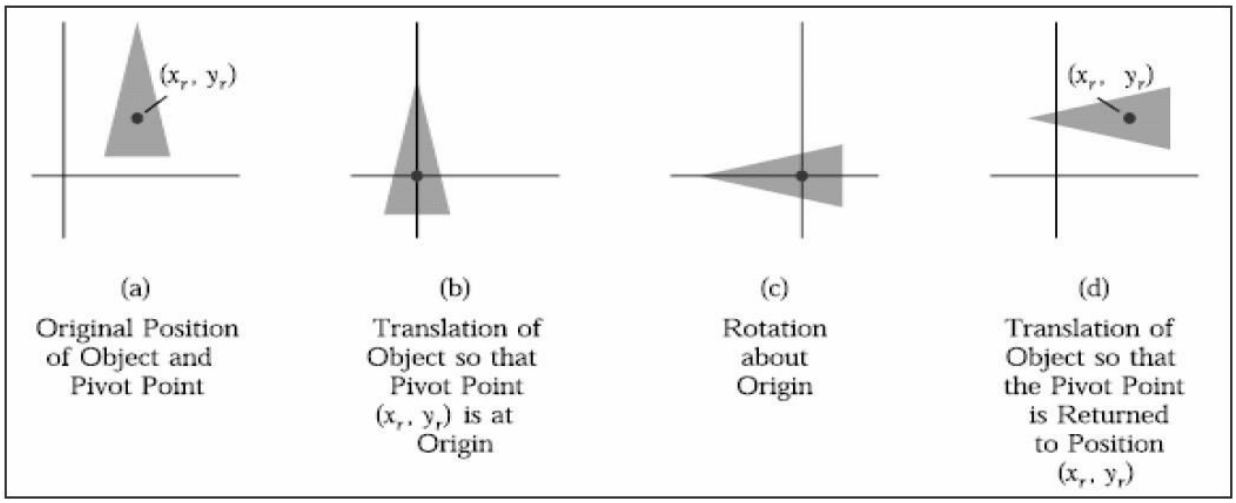
With a graphics package that only provides a rotate function for revolving objects about the coordinate origin, we can generate rotations about any selected pivot point (xr, yr) by performing the following sequence of translate-rotate-translate operations:

1. Translate the object so that the pivot-point position is moved to the coordinate origin.
2. Rotate the object about the coordinate origin.
3. Translate the object so that the pivot point is returned to its original position.

This transformation sequence is illustrated in the following diagram. The composite transformation matrix for this sequence is obtained with the concatenation.

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

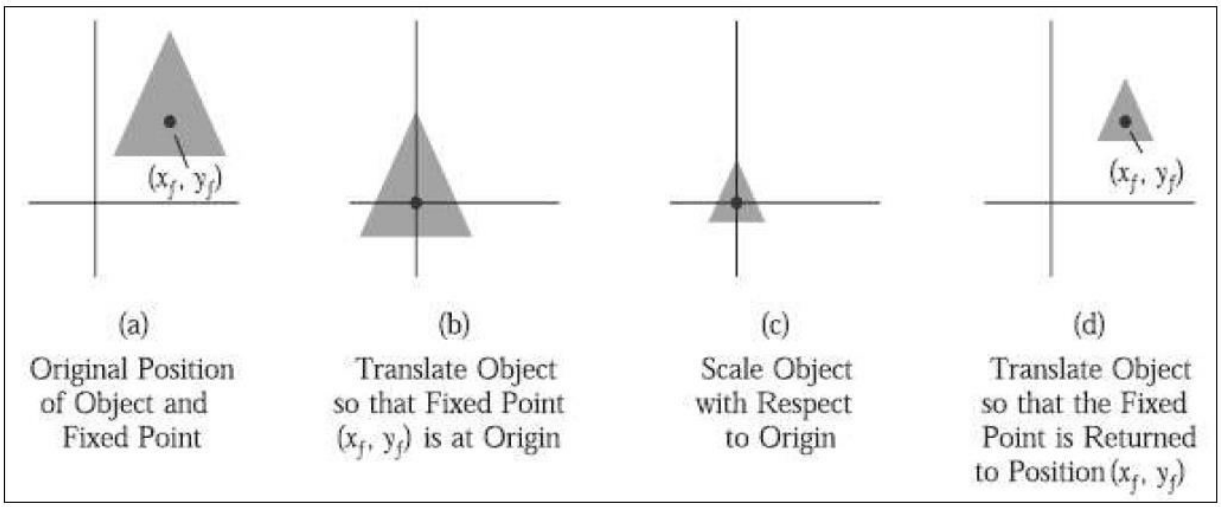
$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$



General Fixed-Point Scaling:

The following diagram illustrates a transformation sequence to produce scaling with respect to a selected fixed position (xf , yf) using a scaling function that can only scale relative to the coordinate origin.

1. Translate object so that the fixed point coincides with the coordinate origin.
2. Scale the object with respect to the coordinate origin.
3. Use the inverse translation of step 1 to return the object to its original position.



Concatenating the matrices for these three operations produces the required scaling matrix

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

or

$$\mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_f, -y_f) = \mathbf{S}(x_f, y_f, s_x, s_y)$$

S.NO	RGPV QUESTIONS	Year	Marks
1.	Obtain the instant transformation matrix for deriving fig B from Fig A. Fig A is described by the vertices A1(2,1), A2(4,1), A3(3,2) and fig B is described by vertices B1(-4,-2), B2(-2,-2) and B3(-3,-1).	June, 2012	10
2.	A triangle is defined by vertices P1(0,0), P2(2,0), and P3(3,2) Is enlarged twice in X-direction and thrice in Y-direction.The vertex P3 of the enlarged triangle is rotated counter-clockwise.Find the resultant points of the triangle.	June 2012	10

Unit-03/Lecture-03

Other Transformation

Reflection:

A reflection is a transformation that produces a mirror image of an object. The mirror image for a two-dimensional reflection is generated relative to an axis of reflection by rotating the object 180" about the reflection axis. Some common reflections are as follows:

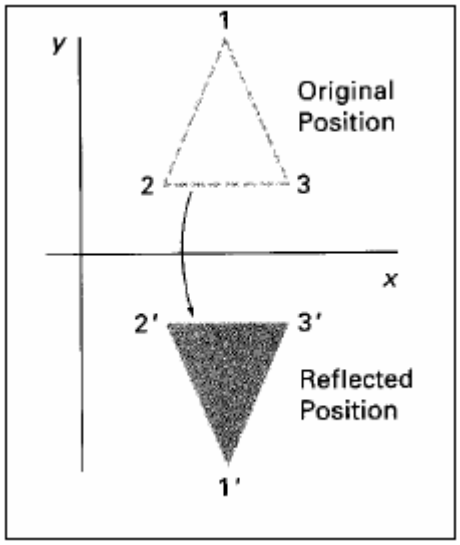
x-Reflection:

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 x values the same, but "flips" the y values of coordinate positions.

The resulting orientation of an object after it has been reflected about the x axis is shown in the diagram.



y-Reflection:

A reflection about the y axis flips x coordinates while keeping y coordinates the same. The matrix for this transformation is,

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

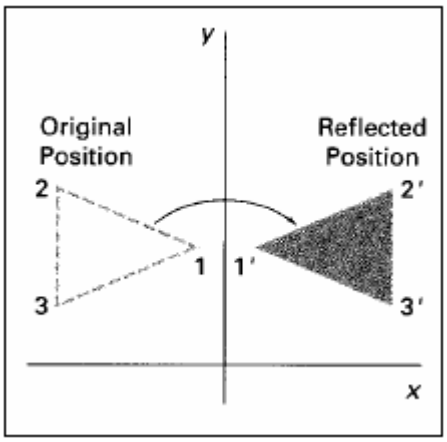
The diagram illustrates the change in position of an object that has been reflected about the line x = 0.

Origin-Reflection:

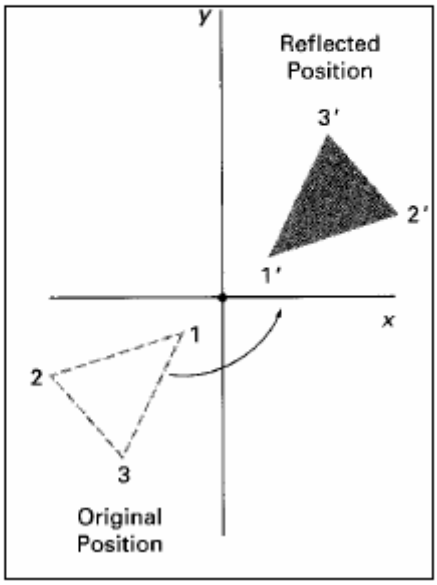
We flip both the x and y coordinates of a point by reflecting relative to an axis that is perpendicular to the xy plane and that passes through the coordinate origin.

This transformation, referred to as a reflection relative to the coordinate origin, has the matrix

representation:



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



If we chose the reflection axis as the diagonal line y = x, the reflection matrix is

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

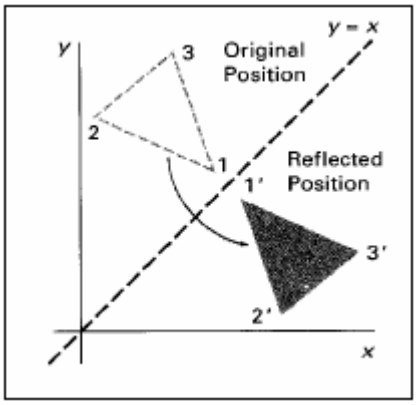
To obtain a transformation matrix for reflection about the diagonal y = - x, we could concatenate matrices for the transformation sequence:

(1) clockwise rotation by 45^0,
(2) reflection about the y axis, and
(3) counterclockwise rotation by 45^0

The resulting transformation matrix is

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

Reflections about any line y = mx + h in the xy plane can be accomplished with a combination of translate-rotate-reflect transformations.



Shear:

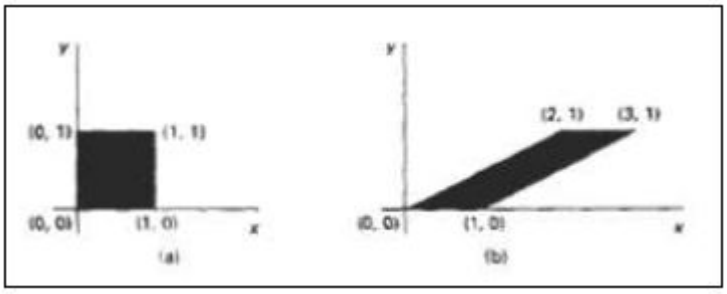
A transformation that distorts (deform or alter) the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called a shear.

Two common shearing transformations are those that shift coordinate x values and those that shift y values.

x-Shearing:

An x-direction shear relative to the x axis is produced with the transformation matrix

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



which transforms coordinate positions as

$$x' = x + sh\,x \cdot y\,,\ y' = y$$

In the following diagram, shx =2, changes the square into a parallelogram.

Negative values for shx shift coordinate positions to the left. We can generate x-direction shears relative to other reference lines with

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

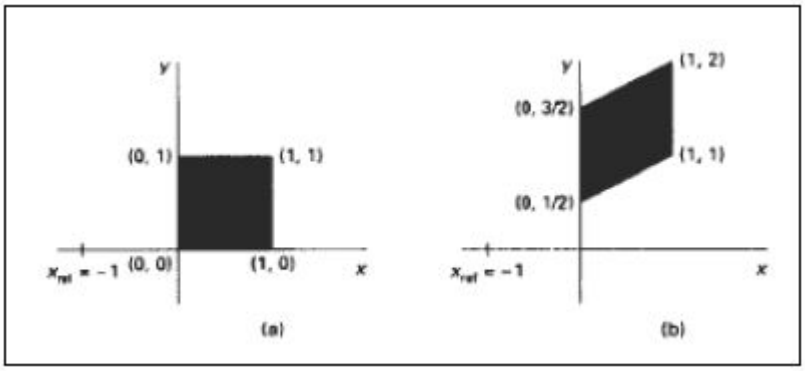
y-Shearing:

A y-direction shear relative to the line x = xref is generated with the transformation matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

which generates transformed coordinate positions

$$x' = x\,,\ y' = sh\,y\,(x - x\,ref) + y$$



S.NO	RGPV QUESTIONS	Year	Marks
1	Why are homogenous coordinate system used for transformation computation in computer graphics?	Dec,2012	5
2	Explain shearing and reflection with example.	June 2011	5
3	Magnify the triangle with vertices A(0,0),B(1,1) and C(5,2) to twice its size while keeping C(5,2) fixed.	June 2013	10
4	Find and show the transformation to reflect a polygon whose vertices are A(-1,0),B(0,-3) ,C(1,0) and D(0,3)about the line y=x+3.	June 2014	10

Unit-03/Lecture-04

2D VIEWING

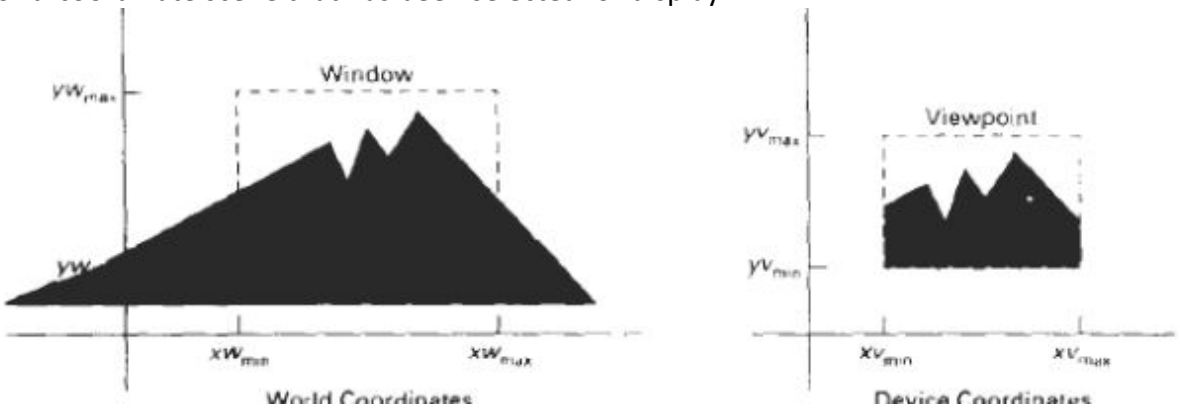
THE VIEWING PIPELINE:

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

- The window defines what is to be viewed.
- The viewport defines where it is to be displayed.

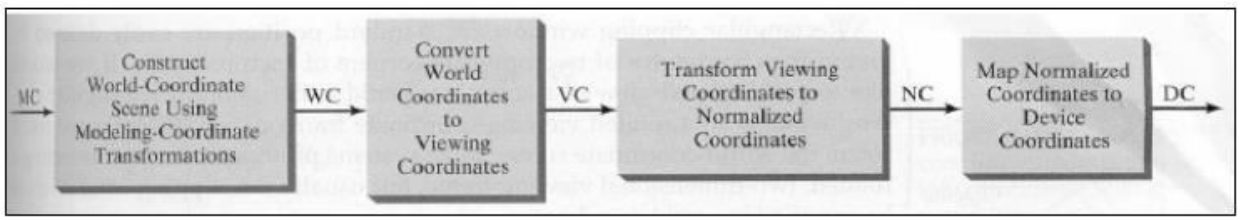
Often, windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes.In general, the mapping of a part of a world-coordinate scene to device coordinates is referred to as a viewing transformation.

Sometimes the two-dimensional viewing transformation is simply referred to as the window-to-viewport transformation or the windowing transformation. The term window to refer to an area of a world-coordinate scene that has been selected for display.



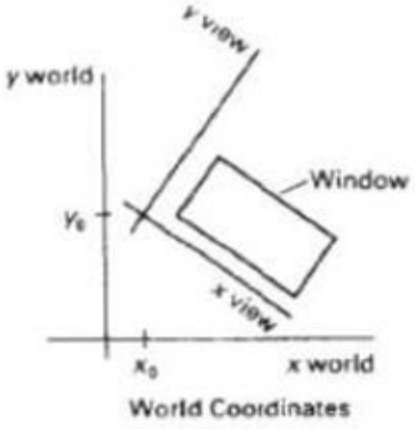
We carry out the viewing transformation in several steps, as indicated below.

1. First, we construct the scene in world coordinates using the output primitives and attributes.
2. Next, to obtain a particular orientation for the window, we can set up a two-dimensional viewing-coordinate system in the world-coordinate plane, and define a window in the viewing-coordinate system.
3. The viewing coordinate reference frame is used to provide a method for setting up arbitrary orientations for rectangular windows. Once the viewing reference frame is established, we can transform descriptions in world coordinates to viewing coordinates.
4. We then define a viewport in normalized coordinates (in the range from 0 to 1) and map the viewing-coordinate description of the scene to normalized coordinates.
5. At the final step, all parts of the picture that lie outside the viewport are clipped, and the contents of the viewport are transferred to device coordinates.

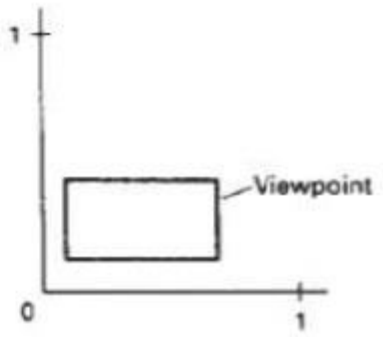


By changing the position of the viewport, we can view objects at different positions on the display area of an output device. Also, by varying the size of viewports, we can change the size and proportions of displayed objects. We achieve zooming effects by successively mapping different-sized windows on a fixed-size viewport.

Panning effects are produced by moving a fixed-size window across the various objects in a scene. When all coordinate transformations are completed, viewport clipping can be performed in normalized coordinates or in device coordinates. This allows us to reduce computations by concatenating the various transformation matrices.



World Coordinates



Normalized
Device Coordinates

VIEWING COORDINATE REFERENCE FRAME:

This coordinate system provides the reference frame for specifying the world coordinate window. First, a viewing-coordinate origin is selected at some world position: Po = (x0, y0). Then we need to establish the orientation, or rotation, of this reference frame. One way to do this is to specify a world vector V that defines the viewing y0, direction. Vector V is called the view up vector.

Given V, we can calculate the components of unit vectors v = (vx, vy) and u = (ux, uy) for the viewing yv and xv axes, respectively. These unit vectors are used to form the first and second rows of the rotation matrix R that aligns the viewing xvy v axes with the world xwy w axes.

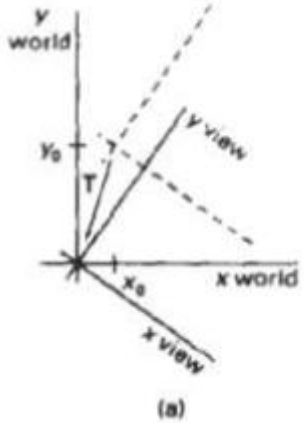
We obtain the matrix for converting world coordinate positions to viewing coordinates as a two-step composite transformation:

- First, we translate the viewing origin to the world origin,
- Then we rotate to align the two coordinate reference frames.

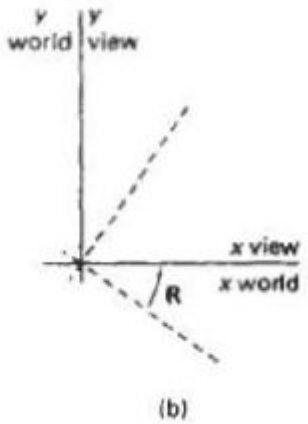
The composite 2D transformation to convert world coordinates to viewing coordinate is

$$\mathbf{M}_{WC,VC} = \mathbf{R} \cdot \mathbf{T}$$

where T is the translation matrix that takes the viewing origin point Po to the world origin, and R is the rotation matrix that aligns the axes of the two reference frames.



(a)



(b)

A viewing-coordinate frame is moved into coincidence with the world frame in two steps:

(a) translate the viewing origin to the world origin, then

(b) rotate to align the axes of the two systems.

WINDOW-TO-VIEWPORT COORDINATE TRANSFORMATION

Once object descriptions have been transferred to the viewing reference frame, we choose the window extents in viewing coordinates and select the viewport limits in normalized coordinates.

Object descriptions are then transferred to normalized device coordinates. We do this using a transformation that maintains the same relative placement of objects in normalized space as they had in viewing coordinates. If a coordinate position is at the center of the viewing window, for instance, it will be displayed at the center of the viewport.

The above diagram illustrates the window-to-viewport mapping. A point at position (xw, yw) in the window is mapped into position (xv, yv) in the associated viewport. To maintain the same relative placement in the viewport as in the window, we require that,

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

Solving these expressions for the viewport position *(xv, yv)*, we have

$$xv = xv_{min} + (xw - xw_{min})sx$$

$$yv = yv_{min} + (yw - yw_{min})sy$$

where the scaling factors are

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

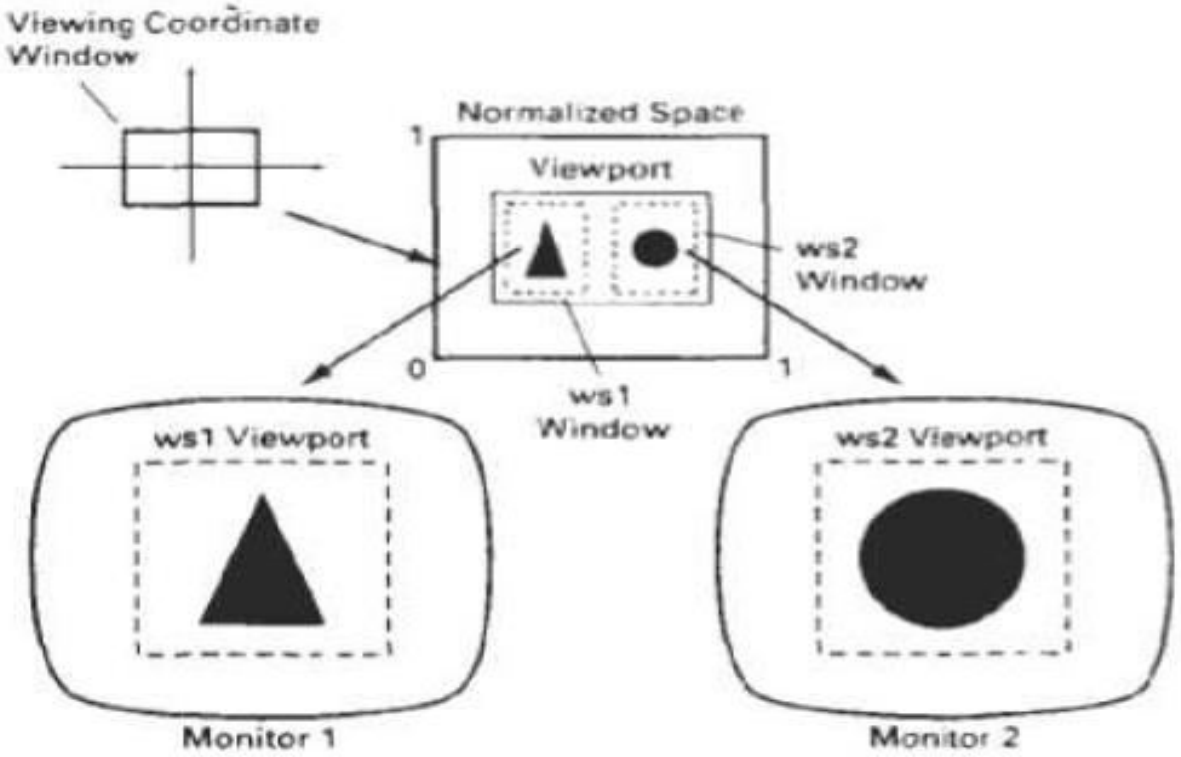
The Equations can also be derived with a set of transformations that converts the window area into the viewport area.

This conversion is performed with the following sequence of transformations:

1. Perform a scaling transformation using a fixed-point position of (xwmin, ywmin) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport.

Relative proportions of objects are maintained if the scaling factors are the same (sx = sy). Otherwise, world objects will be stretched or contracted in either the x or y direction when displayed on the output device.

The mapping, called the workstation transformation, is accomplished by selecting a window area in normalized space and a viewport area in the coordinates of the display device.



S.NO	RGPV QUESTIONS	Year	Marks
1	Write a short note on Normalised Device Co-ordinates	Dec,2012	5
2	Find the normalization transformation of a window whose lower left Conrner at (0,0) and upper right corner is (4,3) onto normalized device screen so that aspect ratio are preserved.	Dec,2010	10

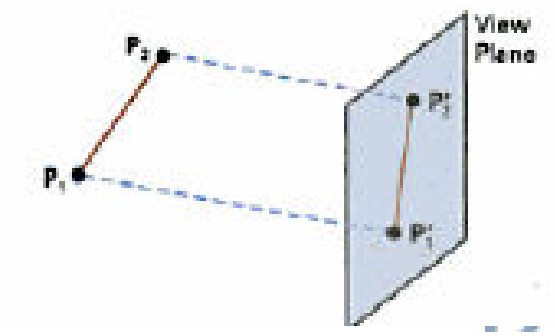
Unit-03/Lecture-05

Parallel and Perspective Projection

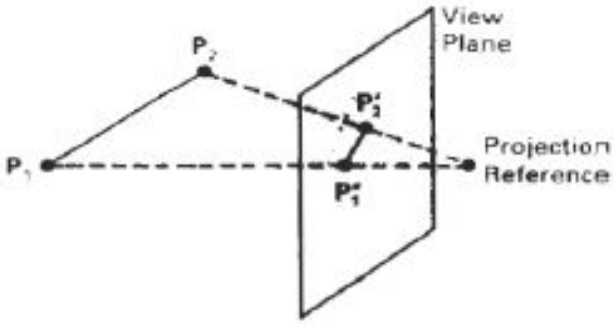
Projections

Once world coordinate descriptions of the objects are converted to viewing coordinates, we can project the 3 dimensional objects onto the two dimensional view planes.
There are two basic types of projection.

1. Parallel Projection - Here the coordinate positions are transformed to the view plane along parallel lines. Parallel projection of an object to the view plane



2. Perspective Projection – Here, object positions are transformed to the view plane along lines that converge to a point called the projection reference point.



Parallel Projections

Parallel projections are specified with a projection vector that defines the direction for the projection lines.When the projection in perpendicular to the view plane, it is said to be an Orthographic parallel projection, otherwise it said to be an Oblique. Orientation of the projection vector Vp to produce an orthographic projection.
Parallel projections have lines of projection that are parallel both in reality and in the projection plane. Parallel projection corresponds to a perspective projection with an infinite focal length (the distance from the image plane to the projection point), or "zoom". Within parallel projection there is an ancillary category known as "pictorials". Pictorials show an image of an object as viewed from a skew direction in order to reveal all three directions (axes) of space in one picture. Because pictorial projections innately contain this distortion, in the rote, drawing instrument for pictorials, some liberties may be taken for economy of effort and best effect.

Parallel vs Perspective Projection

Drawing is a visual art that has been used by man for self-expression throughout history. It uses pencils, pens, colored pencils, charcoal, pastels, markers, and ink brushes to mark different

types of medium such as canvas, wood, plastic, and paper.
It involves the portrayal of objects on a flat surface such as the case in drawing on a piece of paper or a canvas and involves several methods and materials. It is the most common and easiest way of recreating objects and scenes on a two-dimensional medium.
To create a realistic reproduction of scenes and objects, drawing uses two types of projection: parallel projection and perspective projection. What humans usually see is perspective projection. We see a horizon wherein everything looks small, and we see bigger things when they are nearer to us.
Perspective projection is seeing things larger when they're up close and smaller at a distance. It is a three-dimensional projection of objects on a two-dimensional medium such as paper. It allows an artist to produce a visual reproduction of an object which resembles the real one.
The center of projection in a perspective projection is a point which is at a distance from the viewer or artist. Objects located at this point appear smaller and will appear bigger when they are drawn closer to the viewer. Perspective projection produces a more realistic and detailed representation of an object allowing artists to create scenes that closely resemble the real thing. The other type of projection which is also used aside from perspective projection is parallel projection.
Parallel projection, on the other hand, resembles seeing objects which are located far from the viewer through a telescope. It works by making light rays entering the eyes parallel, thus, doing away with the effect of depth in the drawing. Objects produced using parallel projection do not appear larger when they are near or smaller when they are far. It is very useful in architecture. However, when measurements are involved, perspective projection is best.
It provides an easier way of reproducing objects on any medium while having no definite center of projection. When it is not possible to create perspective projection, especially in cases where its use can cause flaws or distortions, parallel projection is used.

Several types of parallel projection are the following:

- Orthographic projection
- Oblique projection
- Cavalier projection
- Cabinet projection

Axonometric projection
Axonometric projection is a type of orthographic projection where the plane or axis of the object depicted is not parallel to the projection plane such that multiple sides of an object are visible in the same image. It is further subdivided into three groups: isometric, dimetric and trimetric projection, depending on the exact angle at which the view deviates from the orthogonal. A typical characteristic of axonometric pictorials is that one axis of space is usually displayed as vertical.

Comparison of several types of graphical projection.

Isometric projection
In isometric pictorials (for protocols see isometric projection), the most common form of axonometric projection, the direction of viewing is such that the three axes of space appear equally foreshortened. There are two commonly used standards for creating scaled isometric drawings. An accurate drawing of a three-dimensional object projected isometrically would have its axis-parallel edges foreshortened by a factor of approx 81.65\%, but for convenience this is usually approximated as 3⁄4. That is, the length of an edge on a drawing of this type would be 3⁄4 of its length on a three-dimensional object. Alternatively, "full-size" isometric

drawings may be made in which no foreshortening is shown: the length of an edge on a drawing is the same as its three-dimensional length.

Dimetric projection

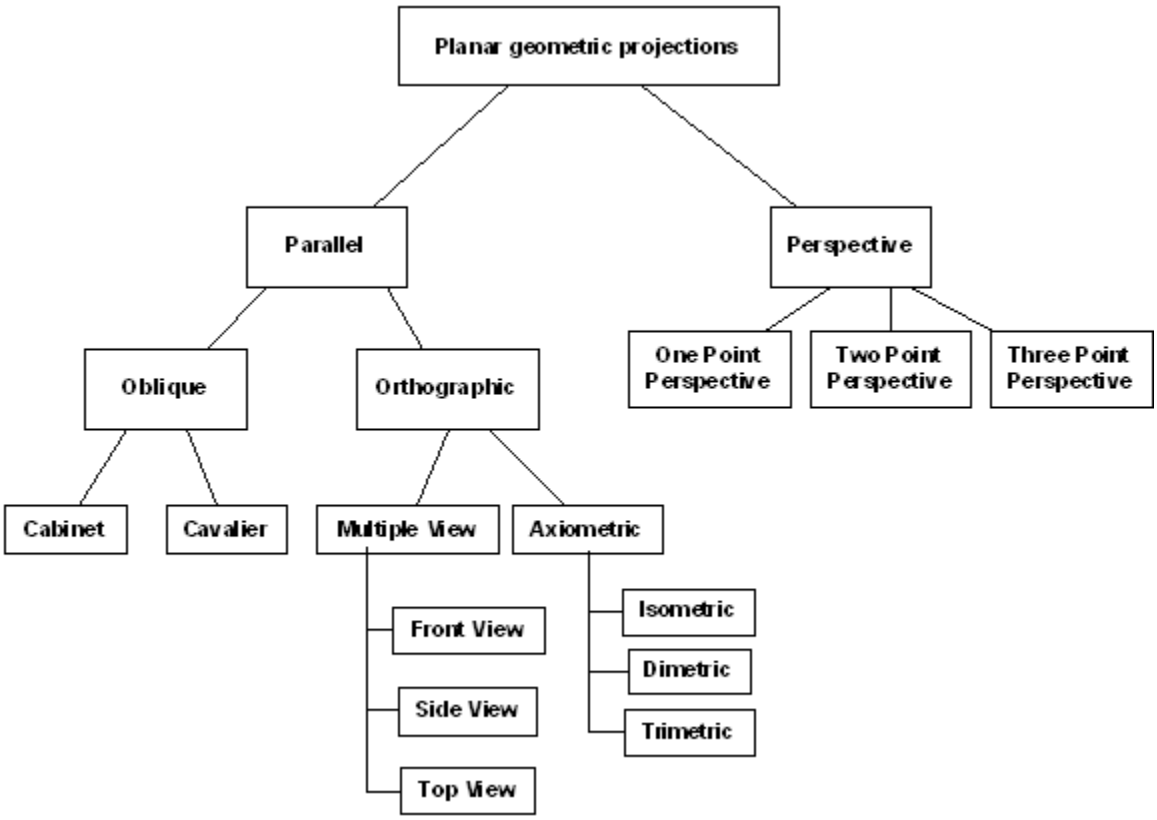
In dimetric pictorials (for protocols see dimetric projection), the direction of viewing is such that two of the three axes of space appear equally foreshortened, of which the attendant scale and angles of presentation are determined according to the angle of viewing; the scale of the third direction (vertical) is determined separately. Approximations are common in dimetric drawings.

Trimetric projection

In trimetric pictorials (for protocols see trimetric projection), the direction of viewing is such that all of the three axes of space appear unequally foreshortened. The scale along each of the three axes and the angles among them are determined separately as dictated by the angle of viewing. Approximations in trimetric drawings are common, and trimetric perspective is seldom used.

Oblique projection

In oblique projections the parallel projection rays are not perpendicular to the viewing plane as with orthographic projection, but strike the projection plane at an angle other than ninety degrees. In both orthographic and oblique projection, parallel lines in space appear parallel on the projected image. Because of its simplicity, oblique projection is used exclusively for pictorial purposes rather than for formal, working drawings. In an oblique pictorial drawing, the displayed angles among the axes as well as the foreshortening factors (scale) are arbitrary. The distortion created thereby is usually attenuated by aligning one plane of the imaged object to be parallel with the plane of projection thereby creating a true shape, full-size image of the chosen plane. Special types of oblique projections are cavalier projection and cabinet projection.

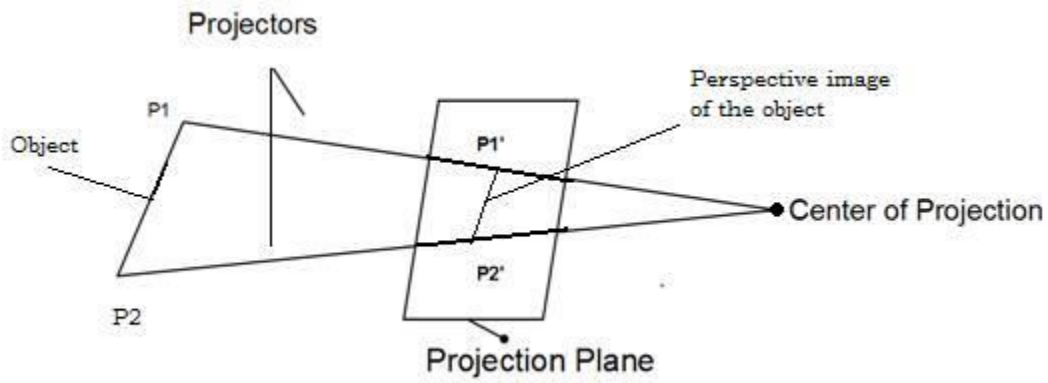


PERSPECTIVE PROJECTION

Perspective projection is a projection in which Center of projection is at a finite distance from a projection plane. The technique of perspective projection is used in preparing perspective

drawings of three dimensional objects and scenes. Perspective Projection is characterized by perspective foreshorting and vanishing points.

Perspective foreshortening is the illusion that objects and length appear smaller as their distance from the center of projection increases. The illusion that certain sets of parallel lines appear to meet at a point is another feature of perspective drawings. These points are Vanishing Points.



S. NO	RGPV QUESTIONS	Year	Marks
1.	Explain different types of Parallel Projections.	Dec,2009	10
2.	Write the composite matrix representation for rotating a 3-D object about an axis that is parallel to one of the co-ordinate axis.	June 2010	10
3.	It is desired to rotate object clockwise through 30 –degree about an axis passing through the origin and the point p (10, 10,0).What are the sequence of transformation matrices that must be used to carry out the desired rotation?	Dec 2011	10
4.	Find a matrix for parallel projection onto the plane 3x+y+4z+1=0 when (a)An orthographic projection is used (b)An oblique projection is used.	June 2012	10
5.	Differentiate between parallel and perspective projection.	Dec,2012	10

UNIT 3/LECTURE 6

Clipping

To clip a line , we need to consider only its endpoints. Three cases can be considered:

- If both endpoints of a line lie inside a clip rectangle, the entire line lies inside the clip rectangle and can be accepted.
- If one endpoint lies inside and one outside ,the line intersects the clip rectangle and we must compute the intersection pint.
- If both endpoints are outside the clip rectangle, the lines may intersect with the clip rectangle and we need to perform further calculations to determine whether they are intersections and if they are, where they occur.

Cohen-Sutherland Line Clipping Algorithm

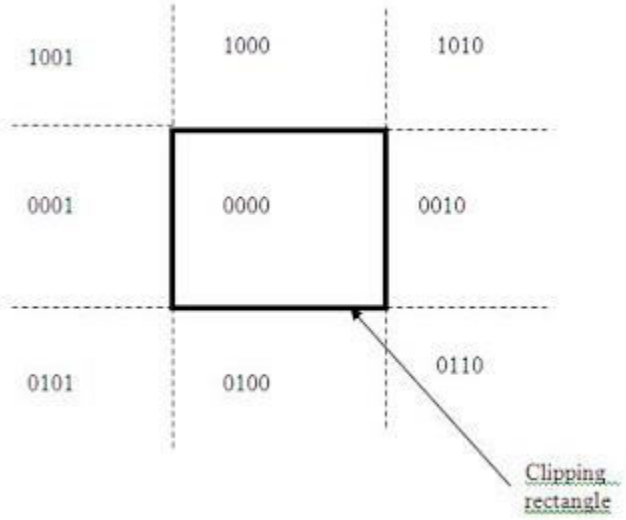
The Cohen-Sutherland algorithm performs initial tests on a line to determine whether intersection calculations can be avoided. First , endpoint pairs are checked for trivial acceptance. If the line cannot be accepted, then the region checks are done.

If the line segment can be neither accepted nor rejected, it is divided into two segments at a clip edge, so that one segment can be rejected. Thus a segment is iteratively clipped by testing for trivial acceptance or rejection and is then subdivided if neither test is successful. The algorithm is particularly efficient for two common cases. In the first case of a large clip rectangle enclosing all or most of the display area, most primitives can be accepted. In the second case of a small clip rectangle, almost all primitives can be rejected.

To perform accept or reject tests, we extend the edges of the clip rectangle to divide the plane of the clip rectangle into nine regions. Each region is assigned a 4-bit code, determined by where the region lies with respect to the outside halfplanes of the clip-rectangle edges. Each bit in the outcode is set to either 1 (true) or 0 (false), the 4 bits in the code correspond to the following conditions:

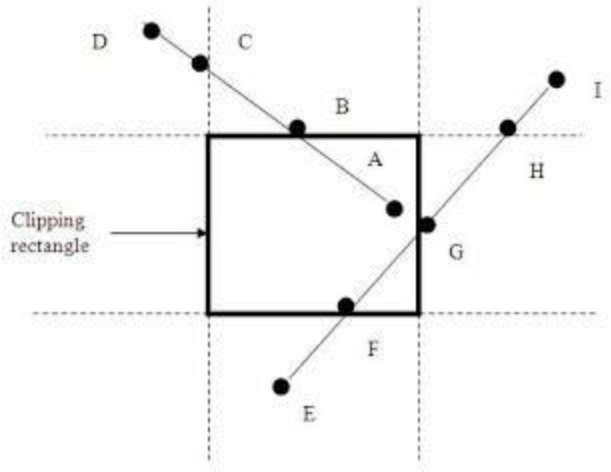
1. first bit outside halfplane of top edge, above top edge $y>y_{max}$
   2. second bit outside halfplane of bottom edge, below bottom edge $y<y_{min}$
   3. third bit outside halfplane of right edge, to the right of right edge $x>x_{max}$
   4. fourth bit outside halfplane of left edge, to the left of left edge $x<x_{min}$

2. Since the region lying above and to the left of the clip rectangle, it is assigned a code of 1001. Each endpoint of the line is assigned the code of the region in which it lies. We can now use these endpoint codes to determine whether the line segment lies completely inside the clip rectangle or in the outside halfplane of the edge. If both 4-bit codes of the endpoints are 0, then the line lies completely inside the clip rectangle.



If a line cannot be accepted or rejected, we must divide it into two segments such that one or both segments can be discarded. We accomplish this subdivision by using an edge that the line crosses to cut the line into two segments. We can choose any order to test the edges but we must use the same order each time in the algorithm. The algorithm works as follows: we compute the outcodes of both endpoints and check for acceptance or rejection. If neither test is successful, we find an endpoint that lies outside and then test the outcode to find the edge that is crossed and to determine the corresponding intersection point. We can then clip off the line segment from the outside endpoint to the intersection point by replacing the outside endpoint with the intersection point and compute the outcode of this new endpoint to prepare for the next iteration.

Example



Consider the line segment AD in the figure. Point A has outcode 0000 and point D has outcode 1001. The line can neither be accepted nor rejected. Therefore the algorithm chooses D as an outside point , whose outcode shows that the line crosses the top edge and left edge. By our testing order, we first use the top edge to clip AD to AB and we compute B's outcode as 0000. In the next iteration we apply the acceptance/rejection test to AB and it is accepted and displayed.

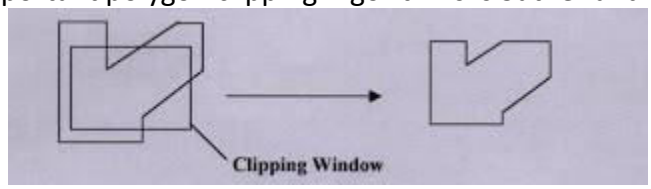
Line EI requires multiple iterations. The first endpoint E, has an outcode of 0100, so the algorithm chooses it as the outside point and tests the outcode to find that the first edge against which the line is cut is the bottom edge, where AI is clipped to FI. In the second

iteration FI cannot be accepted or rejected. The outcode of the first point is 0000, so the algorithm chooses the outside point I that has outcode 1010. The first edge clipped against is therefore the top edge, yielding FH. H's outcode is determined to be 0010, so the third iteration results in a clip against the right edge to FG. This is accepted in fourth and final iteration and displayed. A different sequence of clips would have resulted if we had picked I as the initial point.

On the basis of its outcode we would have clipped against the top edge first, then the right edge and finally the bottom edge.

Area Clipping or Polygon Clipping

In polygon clipping we clip a polygon by processing polygon boundary as a whole against each edge of the clipping window .Each edge of the polygon is tested against each edge of the clipping window, usually a rectangle. As a result, new edges may be added, and existing edges may be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. One of the important polygon clipping Algorithms is Sutherland Hodgeman Algorithm.

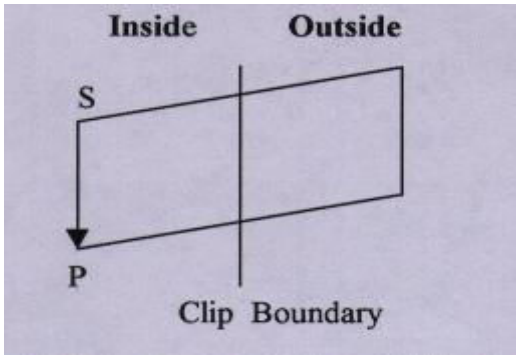


Sutherland Hodgeman Algorithm

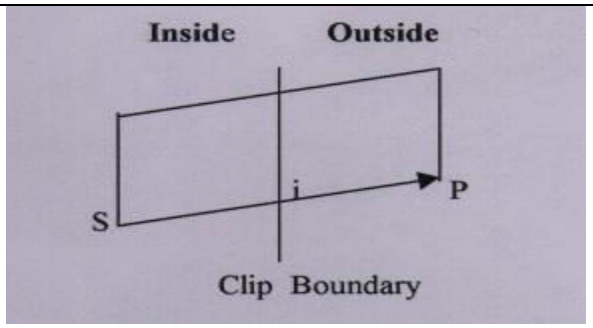
This algorithm is based on a divide-and-conquer strategy that solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clipping edge. This process outputs the series of vertices that define the clipped polygon. Four clipping edges, each defining one boundary of the clipping window, are used to successively to fully clip the polygon. The

There are four cases that we need to analyze when clipping a polygon to a single edge of the clipping window. Let us consider the polygon edge from vertex s to vertex p.

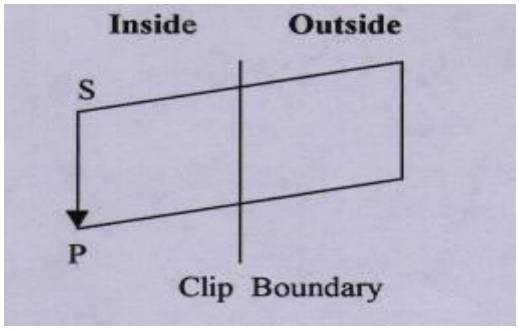
Case 1: If both endpoints (s and p) of the edge are inside the clipping edge, vertex p is added to the output list.

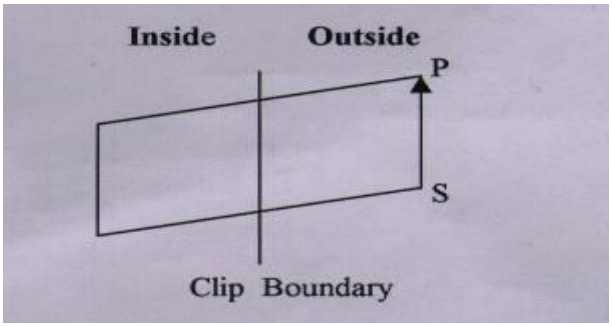


Case 2: If vertex s is inside the clipping edge and vertex p is outside of it, the intersection point, i, is output.

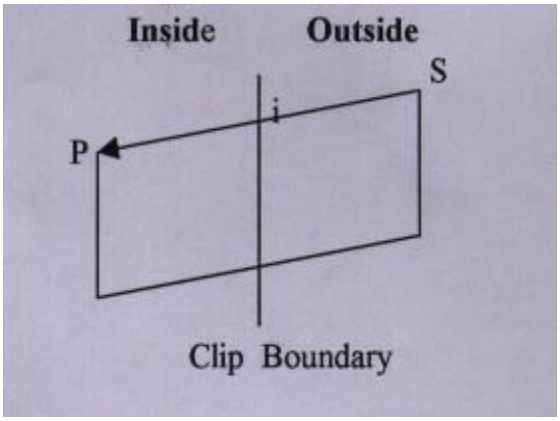


Case 3: If both endpoints (s and p) of the edge are outside of the clipping edge, there is no output.





Case 4: If vertex p is inside of the clipping edge and vertex s is outside of it, the intersection point, i, and vertex p are both added to the output list.



S.NO	RGPV QUESTION	YEAR	MARKS
1.	Explain the midpoint subdivision Line clipping algorithm?	June 2012	10
2	Explain why Sutherland Hodgman polygon clipping algorithm works for only convex clipping region?	June 2013	10

UNIT 3/LECTURE 7

Hidden Surface elimination

Hidden-Surface Removal

The elimination of parts of solid objects that are obscured by others is called hidden-surface removal. Hidden-line removal, which does the same job for objects represented as wireframe skeletons, is a bit trickier.

Methods can be categorized as:

- **Object Space Methods** These methods examine objects, faces, edges etc. to determine which are visible. The complexity depends upon the number of faces, edges etc. in all the objects.
- **Image Space Methods**These methods examine each pixel in the image to determine which face of which object should be displayed at that pixel. The complexity depends upon the number of faces and the number of pixels to be considered.

Z Buffer

The easiest way to achieve hidden-surface removal is to use the depth buffer (sometimes called a z-buffer). A depth buffer works by associating a depth, or distance from the viewpoint, with each pixel on the window. Initially, the depth values for all pixels are set to the largest possible distance, and then the objects in the scene are drawn in any order.

Graphical calculations in hardware or software convert each surface that's drawn to a set of pixels on the window where the surface will appear if it isn't obscured by something else. In addition, the distance from the eye is computed. With depth buffering enabled, before each pixel is drawn, a comparison is done with the depth value already stored at the pixel.

If the new pixel is closer to the eye than what's there, the new pixel's colour and depth values replace those that are currently written into the pixel. If the new pixel's depth is greater than what's currently there, the new pixel would be obscured, and the colour and depth information for the incoming pixel is discarded.

Since information is discarded rather than used for drawing, hidden-surface removal can increase your performance.

To use depth buffering in OpenGL, you need to enable depth buffering. This has to be done only once. Each time you draw the scene, before drawing you need to clear the depth buffer and then draw the objects in the scene in any order.

Scan-Line Algorithm

The scan-line algorithm is another image-space algorithm. It processes the image one scan-line at a time rather than one pixel at a time. By using area coherence of the polygon, the processing efficiency is improved over the pixel oriented method.

Using an active edge table, the scan-line algorithm keeps track of where the projection beam is at any given time during the scan-line sweep. When it enters the projection of a polygon, an IN flag goes on, and the beam switches from the background colour to the colour of the polygon. After the beam leaves the polygon's edge, the colour switches back to background colour. To this point, no depth information need be calculated at all. However, when the scan-line beam finds itself in two or more polygons, it becomes necessary to perform a z-depth sort and select the colour of the nearest polygon as the painting colour.

Accurate bookkeeping is very important for the scan-line algorithm. We assume the scene is defined by at least a polygon table containing the (A, B, C, D) coefficients of the plane of each polygon, intensity/colour information, and pointers to an edge table specifying the bounding lines of the polygon. The edge table contains the coordinates of the two end points, pointers to

the polygon table to indicate which polygons the edge bounds, and the inverse slope of the x-y projection of the line for use with scan-line algorithms. In addition to these two standard data structures, the scan-line algorithm requires an active edge list that keeps track of which edges a given scan line intersects during its sweep. The active edge list should be sorted in order of increasing x at the point of intersection with the scan line. The active edge list is dynamic, growing and shrinking as the scan line progresses down the screen.

Painter's algorithm

The idea behind the Painter's algorithm is to draw polygons far away from the eye first, followed by drawing those that are close to the eye. Hidden surfaces will be written over in the image as the surfaces that obscure them are drawn.

The concept is to map the objects of our scene from the world model to the screen somewhat like an artist creating an oil painting. First she paints the entire canvas with a background colour. Next, she adds the more distant objects such as mountains, fields, and trees. Finally, she creates the foreground with "near" objects to complete the painting. Our approach will be identical. First we sort the polygons according to their z-depth and then paint them to the screen, starting with the far faces and finishing with the near faces.

The algorithm initially sorts the faces in the object into back to front order. The faces are then scan converted in this order onto the screen. Thus a face near the front will obscure a face at the back by overwriting it at any points where their projections overlap. This accomplishes hidden-surface removal without any complex intersection calculations between the two projected faces.

The algorithm is a hybrid algorithm in that it sorts in object space and does the final rendering in image space.

The basic algorithm :

- Sort all polygons in ascending order of maximum z-values.
- Resolve any ambiguities in this ordering.
- Scan converts each polygon in the order generated by steps (1) and (2).

S.NO	RGPV QUESTION	YEAR	MARKS
1.	Explain the Z buffer and painters algorithm for hidden surface elimination?	Dec 2011	10

UNIT 3/LECTURE 8

Basic illumination model

Simple Illumination Model

- Deficiencies
  - point light source
  - no interaction between objects
  - ad hoc, not based on model of light propagation
- Benefits
  - fast
  - acceptable results
  - hardware support

An Illumination model also called Lightening model and sometimes also called Shading modelis used to calculate the intensity of light that at a given point on a surface of an object.
When an object is exposed to light many phenomenon takes place. Light ray is reflected and the intensity with which it goes into our eye is all which is responsible for viewing an object.

Local vs. Global Illumination:
Local illumination only takes into account the relationship between light sources and a single object, and does not consider the effects that result from the presence of multiple objects. For instance, light can be contributed not by a light source, but by a reflection of light from some other object, but local illumination does not visually show it.
In real life there are often multiple sources of light and multiple reflecting objects that interact with each other in many ways.

Global illumination takes into account the light that is reflected from other surfaces to the current surface in addition to the light source. A global illumination model is more comprehensive, more physically correct, and produces more realistic images.

Types of Light Sources

- Point Light
- Spotlight
- Projection Light
- Area Light
- Distant Light
- Infinite Area Light

Major Components for Calculating Light Intensities:

Diffuse Reflection
Diffuse reflection is uniform reflection of light with no directional dependence for the viewer, e.g., a matte surface such as cardboard. Diffuse reflection originates from a combination of internal scattering of light, i.e. the light is absorbed and then re-emitted, and external scattering from the rough surface of the object. An illumination model must handle both direct diffuse reflection, i.e., light coming directly from a source to a surface and then reflected to the viewer, and indirect diffuse reflection (or diffuse interreflections), that is light coming from a source,

being reflected to a surface, then reflected to another surface, etc, and finally to the viewer.

Some local illumination models, such as those that are usually used in many scan-line rendering and ray tracing systems, handle the diffuse inter reflection contribution by simply assuming it has uniform intensity in all directions. This constant term is called the "ambient" contribution and it is considered a non-directional background light. Radiosity explicitly computes the diffuse inter reflection term.

Diffuse reflection of ambient light

If we assume a uniform intensity of ambient light, Ia, then the intensity of the diffuse reflection at any point on the surface: I = ka * Ia, where ka = the ambient diffuse coefficient of reflection with: 0.0 <= ka <= 1.0. For colored surfaces there are three ka values, one for red, green, and blue. We are approximating the ambient diffuse part of the reflection intensity function R(j,q,l) by the constant ka. For a dull, non-reflecting surface ka is close to 0.0 and for a highly reflective surface, such as a mirror, ka is close to 1.0.

Then Lambert's law states that the reflected light is proportional to cos q (with q <= 90 else the light is behind the surface). Let Ip="intensity" of point light source, then the brightness is f(1/d2) where d="distance" of light source from the surface. Some models use a d distance dependence rather than d2, and since most light sources are not really point sources, this works. Some models define a "distant" light source, such as the sun, which have no distance dependence. Here is more information on lights in computer graphics.

Then I = [(kd * Ip) / (D)] * (**N.L**) with D = the distance function (< = 1 for a distant light source, d or d2 for other models and kd = the diffuse coefficient of reflection with: 0.0 <= kd <= 1.0. For colored surfaces there are three kd values, one for red, green, and blue. Some models also add a constant term do to prevent d the denominator from going to zero. Frequently ka is set to be the same as kd. Note that the vectors **N** and **L** are always normalized.

Then including ambient light we get a simple two term illumination model:

I = ka * Ia + [(kd * Ip) / (d)] * (N.L)

Specular Highlights – (Phong Reflection Model)

Specular reflection is the mirror-like reflection of light (or of other kinds of wave) from a surface, in which light from a single incoming direction (aray) is reflected into a single outgoing direction. Such behavior is described by the law of reflection, which states that the direction of incoming light (the incident ray), and the direction of outgoing light reflected (the reflected ray) make the same angle with respect to the surface normal, thus the angle of incidence equals the angle of reflection ($\theta_i = \theta_r$ in the figure), and that the incident, normal, and reflected directions are coplanar.

- Regions of significant brightness, exhibited as spots or bands, characterize objects that specularly reflect light.
- Specular highlights originate from smooth, sometimes mirror like surfaces
- Fresnel equation is used to simulate this effect.
- The Fresnel equation states that for a perfectly reflecting surface the angle of incidence equals the angle of reflection.

Total ray reflected by a point surface ,Ir = Ira + Ird + Irs

Ir = Ia*Ka + (Kd *Ip*(L.N))/d + Ip*Ks*(V.R)^ns

But we are using multiple light sources:

Ir = Ia*Ka + ∑i=0 to n(Kd *Ipi*(Li.N))/di + Ipi*Ks*(V.Ri)^ns

S.NO	RGPV QUESTION	YEAR	MARKS
1.	Explain Phong illumination model with their expression?	Dec,2013	10
2	Compare diffuse reflection and specular reflection?	June 2011	10

UNIT 3/LECTURE 9

Shading Algorithm

Phong Shading !

Phong Shading overcomes some of the disadvantages of Gouraud Shading and specular reflection can be successfully incorporated in the scheme. The first stage in the process is the same as for the Gouraud Shading - for any polygon we evaluate the vertex normals. For each scan line in the polygon we evaluate by linear intrepolation the normal vectors at the end of each line. These two vectors Na and Nb are then used to interpolate Ns. we thus derive a normal vector for each point or pixel on the polygon that is an approximation to the real normal on the curved surface approximated by the polygon. Ns , the interpolated normal vector, is then used in the intensity calculation. The vector interpolation tends to restore the curvature of the original surface that has been approximated by a polygon mesh.

The algorithm is as follows:

1).Compute a normal N for each vertex of the polygon.

2).From bi-linear interpolation compute a normal, Ni for each pixel.

3).From Ni compute an intensity Ii for each pixel of the polygon.

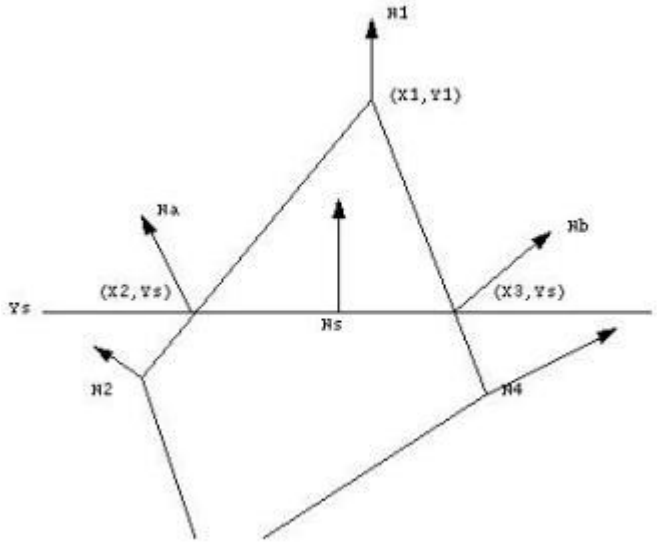
4).Paint pixel to shade corresponding to Ii.

We have:

$$N_a = \frac{1}{y_1 - y_2}[N_1(y_s - y_2) + N_2(y_1 - y_s)]$$

$$N_b = \frac{1}{y_1 - y_4}[N_1(y_s - y_4) + N_4(y_1 - y_s)]$$

$$N_s = \frac{1}{x_b - x_a}[N_a(x_b - x_s) + N_b(x_s - x_a)]$$



These are vector equations that would each be implemented as a set of three equations, one for each of the components of the vectors in world space. This makes the Phong Shading interpolation phase three times as expensive as Gouraud Shading. In addition there is an application of the Phong model intensity equation at every pixel.

Gouraud Shading !

Gouraud shading, computes an intensity for each vertex and then interpolates the computed intensities across the polygons. Gouraud shading performs a bi-linear interpolation of the intensities down and then across scan lines. It thus eliminates the sharp changes at polygon boundaries.

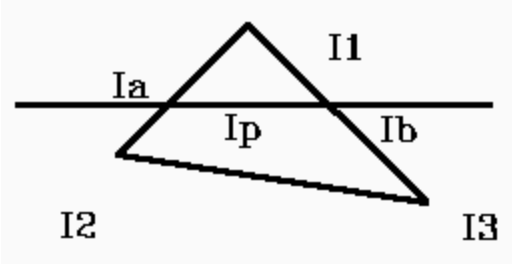
The algorithm is as follows:

1).Compute a normal N for each vertex of the polygon.

2).From N compute an intensity I for each vertex of the polygon.

3).From bi-linear interpolation compute an intensity Ii for each pixel.

4).Paint pixel to shade corresponding to Ii.



S.NO	RGPV QUESTION	YEAR	MARKS
1	Explain Phong and Gouraud shading algorithm?	June 2013	10
2	Differentiate between Phong and Gouraud shading algorithm?	June 2012	10

UNIT 3/LECTURE 10

Color models

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. When this model is associated with a precise description of how the components are to be interpreted (viewing conditions, etc.), the resulting set of colors is called color space. This section describes ways in which human color vision can be modeled.

CIE Chromaticity Diagram

One of the first mathematically defined color spaces is the CIE XYZ color space (also known as CIE 1931 color space), created by the International Commission on Illumination in 1931. These data were measured for human observers and a 2-degree field of view. In 1964, supplemental data for a 10-degree field of view were published.

Note that the tabulated sensitivity curves have a certain amount of arbitrariness in them. The shapes of the individual X, Y and Z sensitivity curves can be measured with a reasonable accuracy. However, the overall luminosity function (which in fact is a weighted sum of these three curves) is subjective, since it involves asking a test person whether two light sources have the same brightness, even if they are in completely different colors. Along the same lines, the relative magnitudes of the X, Y, and Z curves are arbitrary. One could as well define a valid color space with an X sensitivity curve that has twice the amplitude. This new color space would have a different shape. The sensitivity curves in the CIE 1931 and 1964 xyz color space are scaled to have equal areas under the curves.

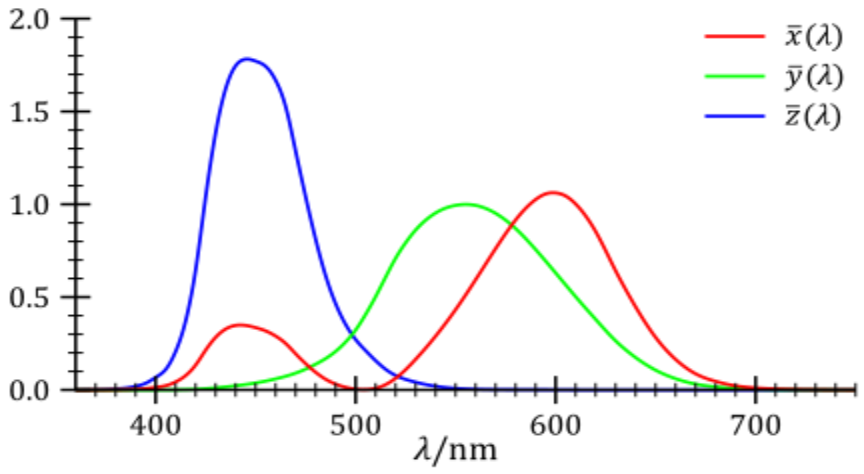
Sometimes XYZ colors are represented by the luminance, Y, and chromaticity coordinates x and y, defined by:

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

Mathematically, x and y are projective coordinates and the colors of the chromaticity diagram occupy a region of the real projective plane. Because the CIE sensitivity curves have equal areas under the curves, light with a flat energy spectrum corresponds to the point (x,y) = (0.333,0.333).

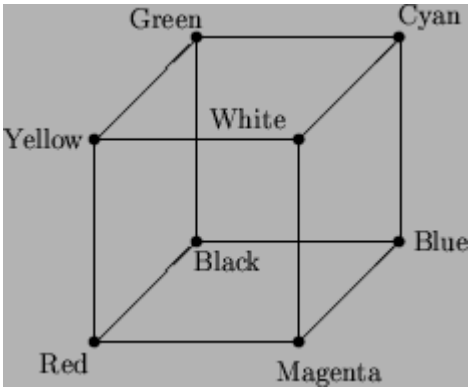
The values for X, Y, and Z are obtained by integrating the product of the spectrum of a light beam and the published color-matching functions.



RGB COLOR MODEL

Media that transmit light (such as television) use additive color mixing with primary colors of red, green, and blue, each of which stimulates one of the three types of the eye's color receptors with as little stimulation as possible of the other two. This is called "RGB" color space. Mixtures of light of these primary colors cover a large part of the human color space and thus produce a large part of human color experiences.

This is whycolor television sets or color computer monitors need only produce mixtures of red, green and blue light.Other primary colors could in principle be used, but with red, green and blue the largest portion of the human color space can be captured. Unfortunately there is no exact consensus as to what loci in the chromaticity diagram the red, green, and blue colors should have, so the same RGB values can give rise to slightly different colors on different screens.



Conversion from (R, G, B) to (X, Y, Z) is given via the chromaticities (X_r, Y_r, Z_r), (X_g, Y_g, Z_g) and (X_b, Y_b, Z_b) of the CRTs phosphors by matrix multiplication via:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} Xr & X_g & X_b \\ Yr & Y_g & Y_b \\ Zr & Z_g & Z_b \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Le

$$C_r := X_r + Y_r + Z_r$$

n $X_r = x_r \cdot C_r \quad Y_r = y_r \cdot C_r$,

$$Z_r = z_r \cdot C_r = (1 - x_r - y_r) \cdot C_r$$

.

This can be calculated from

$X = \frac{x}{y}Y$, $Y = Y$, $Z = \frac{1-x-y}{y}Y$.

The CMY Color Model

This stands for cyan-magenta-yellow and is used for hardcopy devices. In contrast to color on the monitor, the color in printing acts subtractive and not additive. A printed color that looks red absorbs the other two components G and B and reflects R. Thus its (internal) color is G+B=CYAN. Similarly R+B=MAGENTA and R+G=YELLOW. Thus the C-M-Y coordinates are just the complements of the R-G-B coordinates:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

If we want to print a red looking color (i.e. with R-G-B coordinates (1,0,0)) we have to use C-M-Y values of (0,1,1). Note that M absorbs G, similarly Y absorbs B and hence $M+Y$ absorbs all but R.

Black ($(R,G,B) = (0,0,0)$)

corresponds to $(C,M,Y) = (1,1,1)$

which should in principle absorb R, G and B. But in practice this will appear as some dark gray. So in order to be able to produce better contrast printers often use black as 4^{th} color. This is the CMYK-model. Its coordinates are obtained from that of the CMY-model by

$K := \max(C,M,Y)$, $C := C - K$, $M := M - K$, $Y := Y - K$.

The YIQ Color Model

This is used for color TV. Here Y is the luminance (the only component necessary for B&W-TV). The conversion from RGB to YIQ is given by

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.30 & 0.59 & 0.11 \\ 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

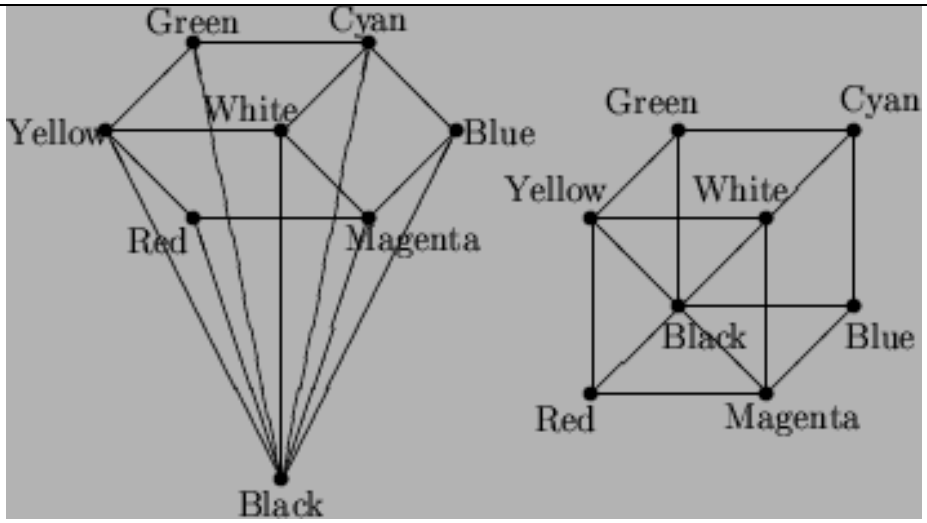
for standard NTSC RGB phosphor with chromaticity values

	R	G	B
x	0.67	0.21	0.14
y	0.33	0.71	0.08

The advantage of this model is that more bandwidth can be assigned to the Y-component (luminance) to which the human eye is more sensible than to color information. So for NTSC TV there are 4MHz assigned to Y, 1.5MHz to I and 0.6MHz to Q.

The HSV color model

All color models treated so far are hardware oriented. The Hue-Saturation-Value model is oriented towards the user/artist. The allowed coordinates fill a six sided pyramid the 3 top faces of the color cube as base. Note that at the same height colors of different perceived brightness are positioned. Value is given by the height, saturation is coded in the distance from the axes and hue by the position on the boundary.



S.NO	RGPV QUESTION	YEAR	MARKS
1	Define RGB Color Model?	June 2013	10
2	Explain CIE Chromaticity diagram?	Dec,2012	10
3	Explain various color Model?	Dec,2011	10