B.E.

Fifth Semester Examination, May-2009

COMPUTER GRAPHICS

Note : Attempt any five questions.

Q. 1. (a) Differentiate between image processing and computer graphics? Briefly explain the components of an image processing system.

Ans. Computer graphics concerns the pictorial synthesis of real or imaginary objects from their computer based models. Whereas related field of image processing treats the converse process. The analysis of scenes, or the reconstruction of models of 2D or 3D objects from their pictures.

Image processing is important in many ways or are no serial surveillance photographs, chromosome scans, x-ray images etc.

Subareas :

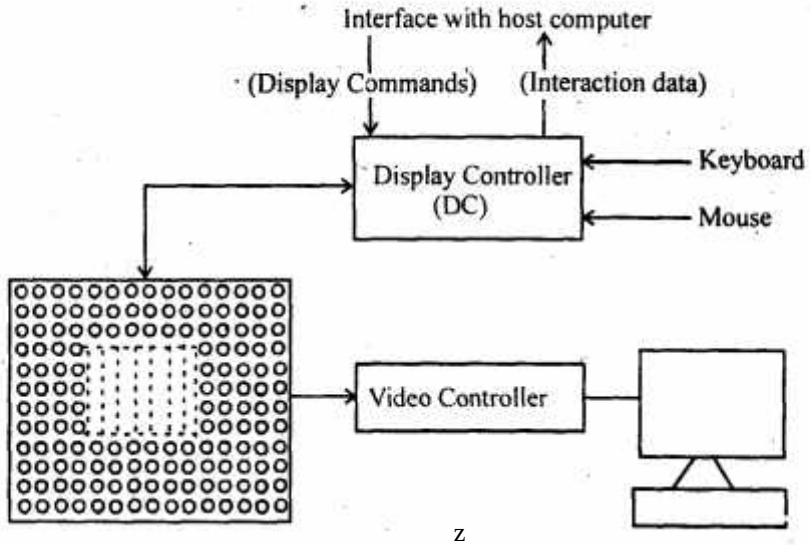
1. Image enhancement : Deals with improving image quality by eliminating noise.

2. Pattern detection : Deal with detecting & clarifying standard patterns & finding deviations from these patterns.

3. Scene analysis : Allow scientists to recognize & reconstruct a 3D model of a scene from several 2D images.

Components of Image Processing System.

Raster Display Architecture :



z

Raster display store the display primitives in a refresh buffer in terms of their components pixels.

There is a hardware display controller that receives & interprets sequences of output commands similar to those of the vector displays.

The complete image on a raster display is formed from the raster, which is a set of horizontal raster lines, each a row of individual pixels; the raster is thus formed as a matrix of pixels representing the entire screen area. The entire image is scanned out sequentially by the video controller, one raster line at a time, from top to bottom & then back to top.

In colored systems, three beams are controlled. Solid state random access memory (RAM) for bitmaps, bilevel CRTs draw images in black & white or black & green; some plasma panels use orange & black, are also used in image processing.

(a) Ideal line drawing

(b) Raster scan with fixed primitives

Scan conversion algorithms, some software waves are also used.

For inputs, mouse, the data tablet, & the transparent touch sensitive panel mounted on screen & other interactive devices are also used.

Q. 1. (b) How transparency modelling can be used to form real images.

Ans. Transparent surfaces can also be useful in picture making. Simple models of transparency do not include the refraction of light through a transparent solid. Lack of refraction can be a decided advantage if transparency is being used not so much to simulate reality as to reveal an object's inner geometry.

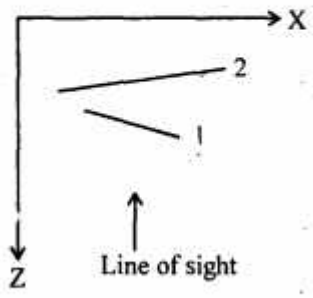
More complex models include refraction, diffuse translucency & the attenuation of light with distance. Similarly, a model of light reflection may simulate the sharp reflections of a perfect mirror reflecting another object or the diffuse reflections of a less highly polished surface.

Transparency is of 2 types :

1. Non-refractive Transparency :

In this we ignore refraction, so light rays are not bent as they pass through the surface.

Interpolated Transparency :



Determine shade of pixel in the intersection of two polygons projections by linearly interpolating the individual shades calculated for two polygons :

$$I_\lambda = (1 - K_{t_1}) I_{\lambda_1} + K_{t_1} I_{\lambda 2}$$

K_{t_1} is transmission coefficient.

Filtered Transparency :

Treats a polygon as a transparent filter that selectively passes different wave lengths.

$$I_\lambda = I_{\lambda_1} + K_{t_1} O_{t_\lambda} I_{\lambda_2}$$

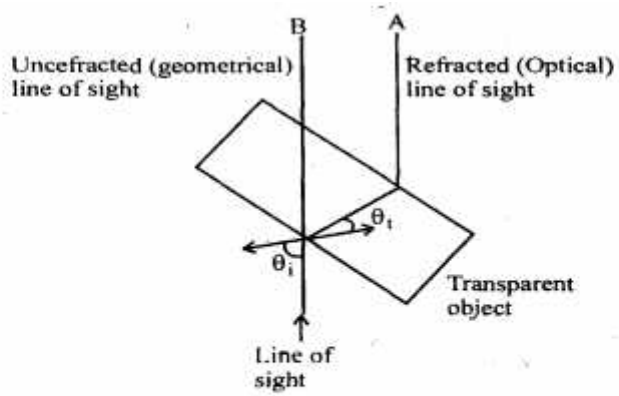
O_{t_λ} is polygon is transparency color.

2. Refractive Transparency :

In this geometrical & optical lines of sight are different :

By Snell's law :

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{n_{t\lambda}}{n_{i\lambda}}$$



Q. 2. Write and explain Sutherland-Cohen algorithm for line clipping? Also simulate the algo for some example.

Ans. Sutherland-Cohen Line Clipping algorithm :

```
typedef unsigned int outcode;

                enum {TOP = 0×1, BOTTOM = 0×2, RIGHT = 0×4, LEFT =0×8};

                void Cohen Suthterland Line (lip And Draw(

double x0 , double y0 , double x1 , double y1 , double xmin, double xmax, double ymin, double ymax, int value)

{

        outcode outcode 0, outcode 1, outcode Out);

        boolean accept = FALSE, done =FALSE;

        outcode 1 = comp out code ( x0 , y0 , xmin, xmax, ymin, ymax);

        outcode 1 = comp out code ( x1 , y1 , xmin, xmax, ymin, ymax);

        do {
```

```
if (: coutcode 0 /outcode 1) {
accept = TRUE, done =FALSE;
}
        else if i(cout code 0 & outcode 1)
        done = TRUE
        else {
        Outcode Out = Outcode 0? Outcode 0 : Outcode 1;
        if (outcode out & top){
                x = x^0 + (x^1 + x^0) * (y max - y^0) / (y^1 - y^0);
                y = ymax;
            } else if (outcode out & BOTTOM) {
                x = x^0 + (x^1 - x^0) * (y min - y^0) / (y^1 - y^0);
                y = ymin;
            } else if (outcodeOut & RIGHT) {
                y = y^0 + (y^1 - y^0) * (x max - x^0) / (x^1 - x^0);
                x = x max;
            } else {
                y = y^0 + (y^1 - y^0) * (x min - x^0) / (x^1 - x^0);
                x = x min ;
            }
        if (Outcode Out == Outcode 0){
        x^0 = x , y^0 = y ; Outcode 0 = compout code (x^0, y^0, xmin, xmax, ymin, ymax);
```

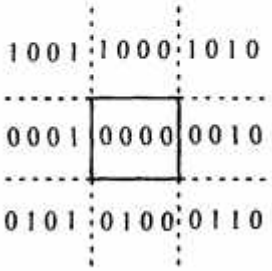
```
}else{
        x1 = x ; y1 = y ; Outcode = compout code (x1,y1,xmin, xmax, ymin, ymax)
    }
}
        { while (done = = FALSE);
        if (accept)
MidpointLine Real (x0,y0,x1,y1, value);
}
outcode compout code (
double x, double y, double xmin, double xmax, double ymin, double ymax);;
{
    outcode code = 0;
    if(y>ymax)
    code 1 = TOP;
    else if (y < y min)
    code 1 = BOTTOM;
    if(x>x max)
    Code 1 = RIGHT;
    else if(x<xmin)
    code 1 = LEFT;
    return code;
}.
```

This code performs initial tests on a line to determine whether intersection calculus can be avoided.

First endpoints pairs are checked for trivial acceptance. It can't then regional checks are done.

If the line can't be trivially accepted or rejected, it is divided into two segments at a clip edge. So that one segment can be trivially rejected. Thus, a segment is iteratively clipped by testing for trivial acceptance or rejection & is then subdivided if neither test is successful, until what remains is completely inside clip rectangle or can be trivially rejected.

We extend the edges of clip rectangle to divide the plane of clip rectangle into nine regions.



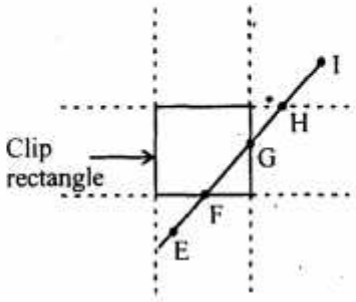
Each region is assigned a 4 bit code, determined by where the region lies with respect to the outside half planes of clip rectangle edges.

First bit	above top edge	y > y max
Second bit	below bottom edge	y > ymin
Third bit	right of right edge	x > xmax
Fourth bit	left of left edge	x > xmin

We compute outcodes of both end points & check for trivial acceptance & rejection. If neither test is successful, we find an endpoint that lies outside & then test the outcode to find the edge i.e., crossed & to determine the corresponding intersection point.

We can then clip off the line segment from the outside end point to the intersection point by replacing the outside endpoint with the intersection point & compute outcode of this new endpoint to prepare for the next iteration.

For example :



Firstly EI is clipped to F1 closing E as outside point. Then FI is clipped to FH closing I as outside point. Now, FG comes after clipping FH & now FG is trivialiy accepted.

This type of its rations can be performed on any type of line resulting in clipped line by seeing the outcodes of the end points.

Q. 3. Briefly discuss the following :

(i) Locator input devices

(ii) Valuator input devices

(iii) Video mixing

(iv) LCD display.

Ans. (i) Locator input devices :

These devices are used to indicate a position or orientation

1. Tablet :

A tablet is a flat surface, ranging in size from about 6 by 6 inches upto 48 by 72 inches or more, which can detect the position of a movable stylus or puck held in the user's hand.

2. Mouse :

It is a small hand held device whose relative motion across a surface can be measured to move the pointer on the screen.

3. Trackball :

It is an upside down mechanical mouse. The motion of trackball is sensed by potentiometers or shaft encoders.

4. Joystick :

Can be moved left or right, forward or backward & potentiometers sense the movements. Mainly used in games & springs return it to original position.

5. Touch panel :

Allows user to point at screen directly with a finger to move the cursor. Various technologies are used. For e.g., low resolution panels use a series of infrared LEds & light sensors to form a grid of invisible light beams.

6. Light pen :

It detects light pulses, on a raster display the event caused can be used to save the video controllers X & Y registers & interrupt the computer.

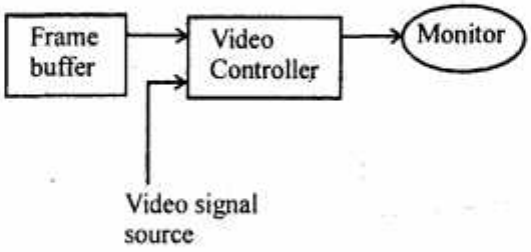
(ii) Valuator input devices :

To give single scalar values these devices are used. Most that provide scalar values are based on potentiometers, like the volume & tone controls of a stereo set. Valuators are usually rotary potentiometers, typically mounted in a group of eight or ten. Simple rotary potentiometers can be rotated through about 330°; this may not be enough to provide both adequate range & resolution.

Continuous-turn potentiometers can be rotated freely in either direction, & hence are unbounded in range. Linear potentiometers, which are of necessity bounded devices, are used infrequently in graphics systems.

(iii) Video Mixing :

In this, two images, one defined in the frame buffer & the other defined by a video signal coming from a television camera, recorder or other source can be merged to form a composite image.



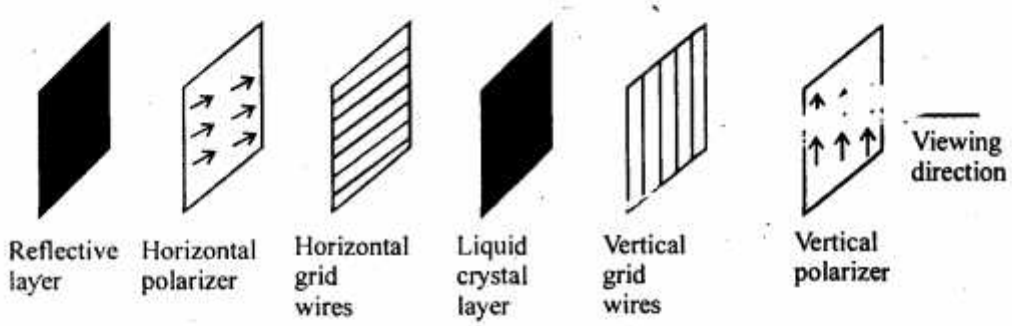
There are 2 types of mixing. In one, a graphics image is set into a video image. The mixing is accomplished with h/w that treats a designated pixel value in the frame buffer as a flag to indicate that the video signal should be shown instead of signal from frame buffer.

The second type of mixing places the video image on top of frame buffer image, as when a weather

reporter stands in front of a full screen weather map. The reporter is actually standing in front of a backdrop, whose color is used to control the mixing. Whenever the incoming video is blue, the frame buffer is shown; otherwise, the video image is shown.

(iv) LCD Display :

A Liquid-crystal display is made up of 6 layers.



The liquid-crystal material is made up of long crystalline molecules. The individual molecules are arranged in a spiral fashion such that the direction of polarization of polarized light passing through is rotated 90° light entering through the front layer is polarized vertically. As the light passes through the liquid crystal, the polarization is rotated 90° to horizontal so light now passes through the rear horizontal polarizer, is reflected, & returns through the two polarizers & crystal.

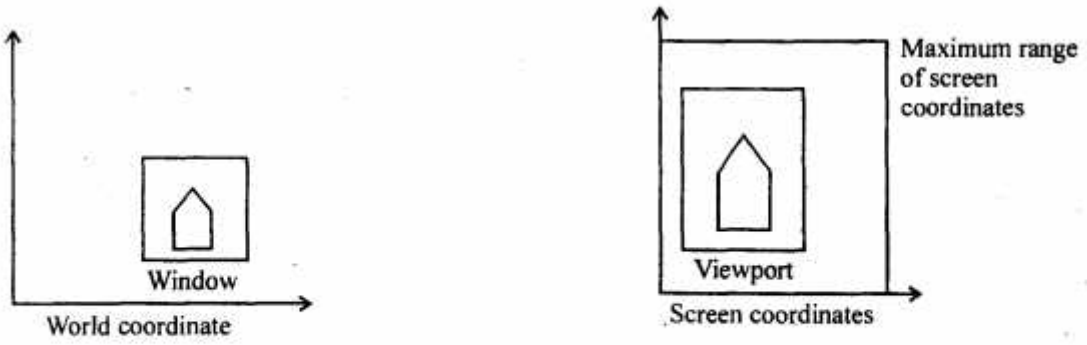
When the crystals are in an electric field, they all line up in the same direction & thus have no polarizing effect. Hence, crystals in electric field do not change the polarization of transmitted light, so light remains vertically polarized & does not pass through the rear polarizer. The light is absorbed, so the viewer sees a dark spot on the display.

A dark spot at point (x, y) is created via matrix addressing.

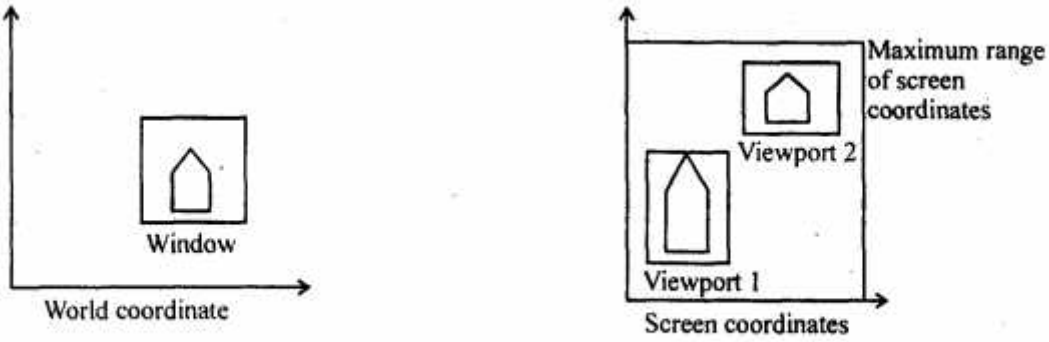
Q. 4. (a) Write the steps required to transform a world co-ordinate window to view last part. Write all the equations.

Ans. Some graphics packages allow the programmer to specify output primitive coordinates in a floating-point world-coordinate system, using whatever units are meaningful to the application program.

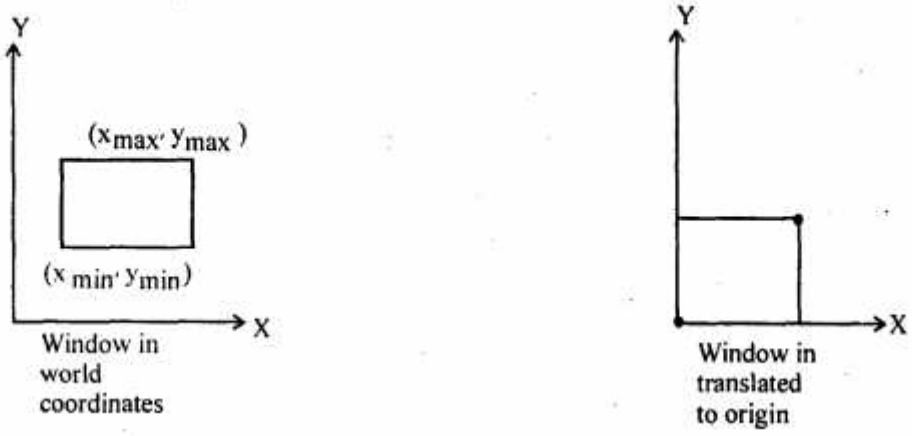
The graphics subroutine package must be told how to map world coordinated onto screen coordiantes. We could do this mapping by having the application programmer provide the graphics package with a transformation matrix, or by specifying a rectangular region in world coordiantes, called "world-coordinate-window" & a corresponding rectangular region in the screen coordiantes called the viewport, into which the world-coordinate-window is to be mapped. The transformation that maps the window into the viewport is applied to all of the output primitives in world coordiantes, thus mapping them into screen coordinates.

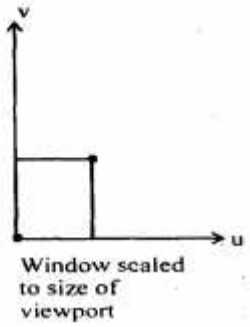


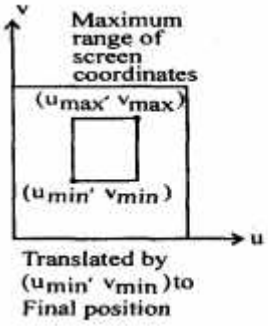
If the window & viewport do not have the same height-to-width ratio, a non-uniform scaling occurs. If the application program changes the window or viewport, then new output primitives drawn onto the screen will be affected by the change. Existing output primitives are not affected.



Given a window & viewport, the transformation matrix that maps the window from world coordiantes into the viewport in screen coordinates. This matrix can be developed as a three-step transformation composition.







Window in world coordinates Window translated to origin Window scaled to size of viewport Translated by (u_{min}, v_{min}) + final position.

$$M_{wv} = T(u_{min}, v_{min}).S\left(\frac{u_{max}-u_{min}}{x_{max}-x_{min}}, \frac{v_{max}-v_{min}}{y_{max}-y_{min}}\right).T(-x_{min}, -y_{min})$$

$$= \begin{bmatrix} 1 & 0 & u_{min} \\ 0 & 1 & v_{min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & 0 \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{min} \\ 0 & 1 & y_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & -x_{min}.\frac{u_{max}-u_{min}}{x_{max}-x_{min}} + u_{min} \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & -y_{min}.\frac{v_{max}-v_{min}}{y_{max}-y_{min}} + v_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

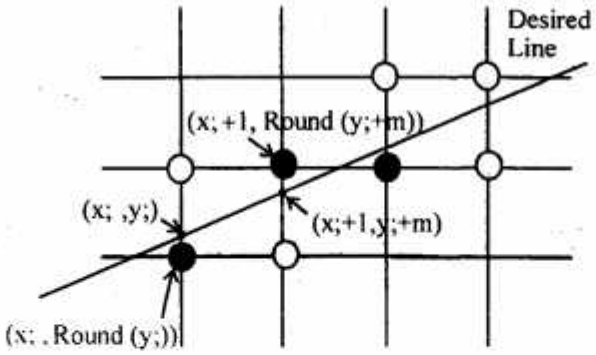
Multiplying $P = M_{wv}[x \quad y \quad 1]^T$ gives the expected result :

$$P = \left[(x - x_{min}) \cdot \frac{u_{max} - u_{min}}{x_{max} - x_{min}} + u_{min} \quad (y - y_{min}) \cdot \frac{v_{max} - v_{min}}{y_{max} - y_{min}} + v_{min} \quad 1 \right]$$

Q. 4. (b) Use pseudo-code to describe the DDA algorithm for scan converting a line whose slope is between 45° and –45°.

Ans.

```
Void line (
int x0, int y0,
int x1, int y1,
int value)
{
    int x;
    double dy = y1–y0;
    double dx = x1 – x0;
    double m = dy/dx;
    double y = y0;
    for (x = x0; x <= x1; x++) {
    write Pixel (x, Round (y), value);
    y + = m;
}
```



Incremental calculation of (x_i, y_i)

This algorithm is called digital differential analyzer (DDA) algorithm. The DDA is a mechanical device that solves differential equations by numerical methods. It traces out successive (x, y) values by simultaneously incriminating x & y by small steps proportional to first dericative of x and y.

If $|m| > 1$, a step in x creates a step in y that is greater than 1. Thus, we must reverse the roles of x & y by assigning a unit step to y & incriminating x by $\Delta x = \Delta y / m = 1 / m$.

Q. 5. (a) What do you mean by instance transformation? Explain with suitable example.

Ans. Instance transformations are done by transforming each part of an object.

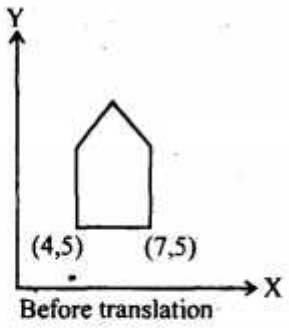
(i) Translation :

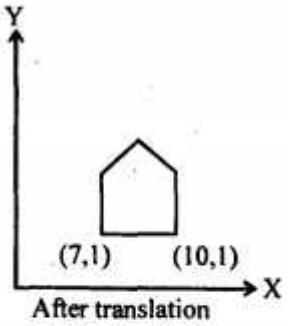
For each point p(x, y) to be moved by d_x units parallel to the x-axis & by d_y units parallel to y-axis to new point p'(x', y')

$$x' = x + d_x \qquad y' = y + d_y$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \qquad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \qquad T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$P' = P + T$$



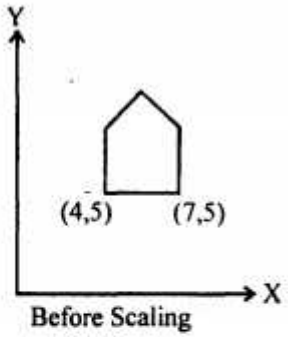


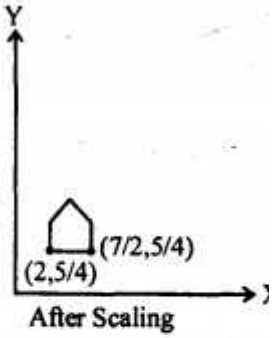
(ii) Scaling :

$$x' = S_x . x \qquad y' = S_y . y$$

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

$$P' = S.P$$



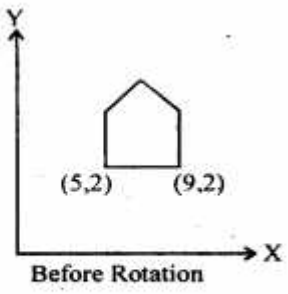


(iii) Rotation :

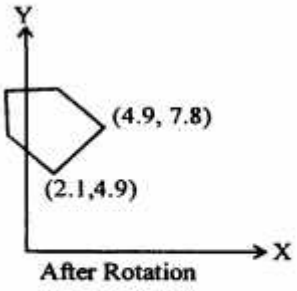
$$x' = x.\cos\theta - y.\sin\theta \text{ v}$$

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

$$P' = R.P$$



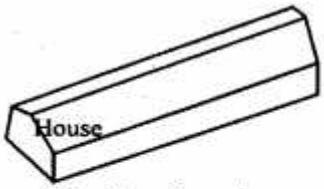
Before Rotation



After Rotation

For example : House

Before Transformation

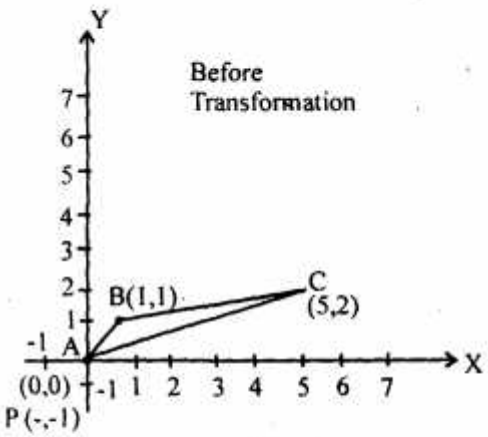


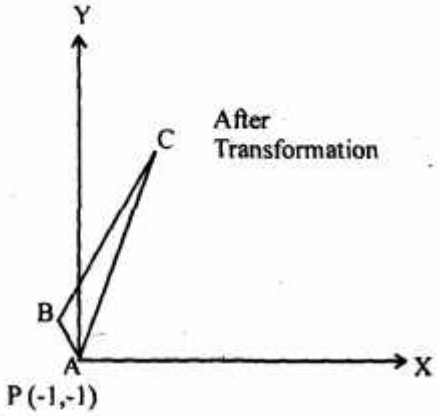
After Transformation

```
SPH_open Structure (STREET_STRUCT)
SPH_set Edge Color (COLOR_WHITE);
SPH_set Interior color (COLOR_YELLOW);
Set up transformation;
SPH_execute structure (HOUSE_STRUCT);
SPH_set Interior color (COLOR_NAVY);
Set up transformation;
SPH_execute Structure (HOUSE_STRUCT);
Set up transformation;
SPH_execute Structure (HOUSE_STRUCT);
SPH_Close Structure( );
```

Q. 5. (b) Perform 45° rotation of triangle A(0, 0), B (1, 1), C (5, 2) about point P (–1, –1).

Ans.





A (0, 0), B(1, 1), C(5, 2)

About $P(-1,-1)$ rotate 45°

$T(x_1,y_1).R(\theta).T(-x_1,y_1)$

$$= \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta)+y_1\sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta)-x_1\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$x_1 = -1 \;\; y_1 = -1 \;\; \theta = 45°$

$$= \begin{bmatrix} \cos 45° & -\text{sion}45° & -1(1-\cos 45°)-1\sin 45° \\ \sin 45° & \cos 45° & -1(1-\cos 45°)+1\sin 45° \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1\left(1-\frac{1}{\sqrt{2}}\right)-\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -1\left(1-\frac{1}{\sqrt{2}}\right)+\frac{1}{\sqrt{2}} \\ 0 & 0 & 1 \end{bmatrix}$$

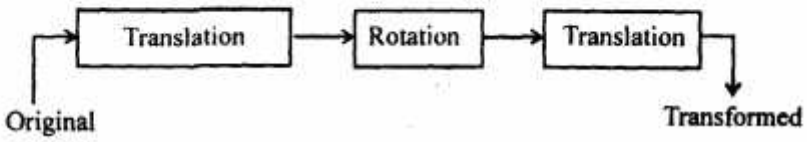
$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}-1+\frac{1}{\sqrt{2}} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \sqrt{2}-1 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying this by $[x \quad y \quad 1]^T$ we get required result.

For this transformation, three steps were followed :

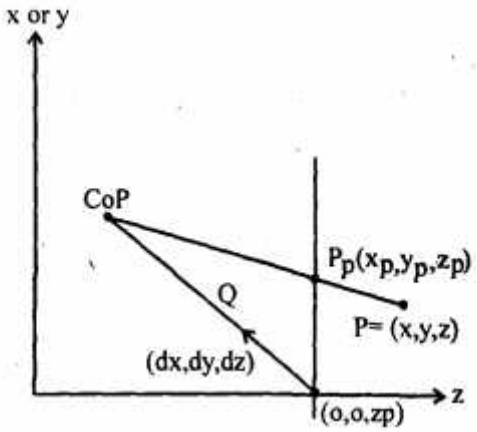
1. Translate point $P(-1,-1)$ to origin.
2. Rotate about 45° on origin.
3. Translate back



Q. 6. (a) Derive the general equation of parallel projection onto a given view plane in the direction of a given projector $\vec{v}$.

Ans. *The projection of general point* $P(x,y,z)$ *onto the projection plane is* $P_p = (x_p, y_p, z_p)$. The projection plane is perpendicular to the z-axis at a distance z_p from the origin & the centre of projection (COP) is a distance Q from the point $(0,0,z_p)$. The direction from $(0,0,z_p)$ to COP is given by the normalized direction vector (d_x, d_y, d_z). P_p is on the line between COP & P, which can be specified parametrically as

$$COP + t(P - COP), \quad 0 \le t \le 1 \qquad ...(1)$$



Rewriting above equation as separate equations for the arbitrary point $P' = (x', y', z')$ on the line, which $COP = (0,0,z_p) + \theta(d_x, d_y, d_z)$, yields

$$x' = \theta d_x + (x - \theta d_x)t \qquad ...(4)$$

$$y' = \theta d_y + (y - \theta d_y)t \qquad ...(5)$$

$$y' = (z_p + \theta d_z) + (z - (z_p + \theta d_z))t \qquad ...(6)$$

We find the projection P_p of point P, at the intersection of line between COP & P with the projection plane, by & then substituting $z' = zp$ into equation (6) to find 't' & then $x' = x_p$ & $y' = y_p$ in equations (4) & (5).

$$t = \frac{z_p - (z_p + \theta d_z)}{z - (z_p + \theta d_z)}$$

$$x_p = \frac{x - z\frac{d_x}{d_z} + z_p\frac{d_x}{d_z}}{\frac{z_p - z}{\theta d_z} + 1}$$

$$y_p = \frac{y - z\frac{d_y}{d_z} + z_p\frac{d_y}{d_z}}{\frac{z_p - z}{\theta d_z} + 1}$$

Multiplying $z_p = z_p$ on RHS by a fraction whose numerator & denominator are both,

$$\frac{z_p - z}{\theta d_z} + 1$$

Maintains the identity & gives z_p the same denominator as, x_p & y_p :

$$z_p = z_p \frac{\frac{z_p - z}{\theta d_z} + 1}{\frac{z_p - z}{\theta d_z} + 1} = \frac{-z\frac{z_p}{\theta d_z} + \frac{z_p^2 + z_p \theta d_z}{\theta d_z}}{\frac{z_p - z}{\theta d_z} + 1}$$

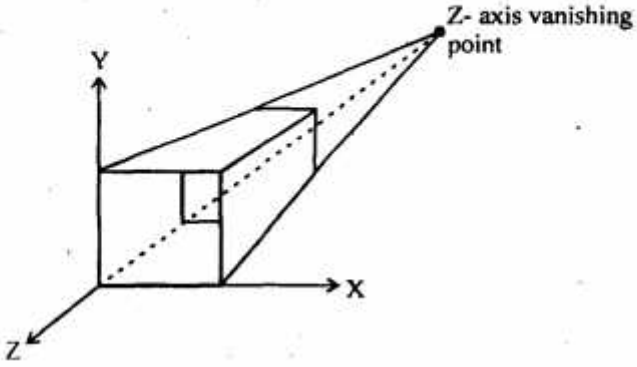
These can be written as a 4×4 matrix $M_{general}$ arranged so that the last row of $M_{general}$ multiplied by $[x \quad y \quad z \quad 1]^T$ produces their common denominator, which is the homogeneous coordiantes W & is hence the divisor of X, Y & Z.

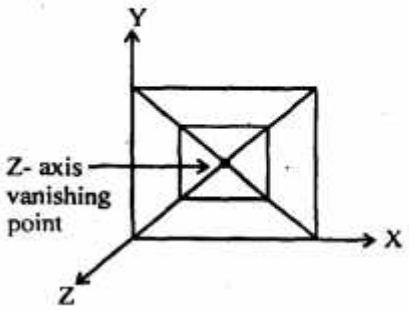
$$M_{general} = \begin{bmatrix} 1 & 0 & -\frac{d_x}{d_z} & z_p\frac{d_x}{d_z} \\ 0 & 1 & -\frac{d_y}{dz} & z_p\frac{d_y}{d_z} \\ 0 & 0 & \frac{-z_p}{\theta d_z} & \frac{z_p^2}{\theta d_z}+z_p \\ 0 & 0 & \frac{-1}{\theta d_z} & \frac{z_p}{\theta d_z}+1 \end{bmatrix}$$

	z_p	θ	$[d_x \quad d_y \quad d_z]$
m_{art}	0	∞	$[0 \quad 0 \quad -1]$
M_{per}	d	d	$[0 \quad 0 \quad -1]$
M'_{per}	0	d	$[0 \quad 0 \quad -1]$
Cavalier	0	∞	$[\cos\alpha \quad \sin\theta \quad -1]$
Cabinet	0	∞	$\left[\frac{\cos\alpha}{2} \quad \frac{\sin\alpha}{2} \quad -1\right]$

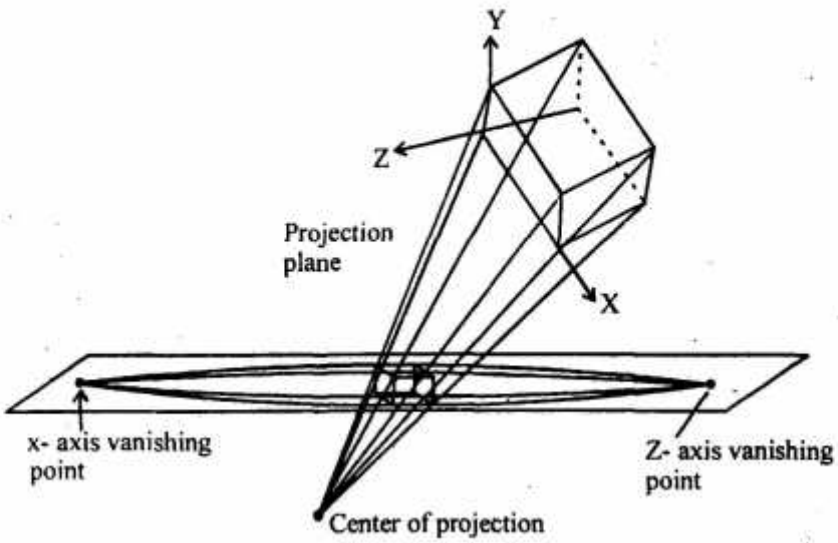
Q. 6. (b) What are principal vanishing points for standard perspective transformation?

Ans. Perspective projections are categorized by their number of principal vanishing points & therefore by the number of axes the projection plane cuts.





Shows two different one-point perspective projections of a cube. It is clear that they are one-point projections because lines parallel to x & y axes do not converges.

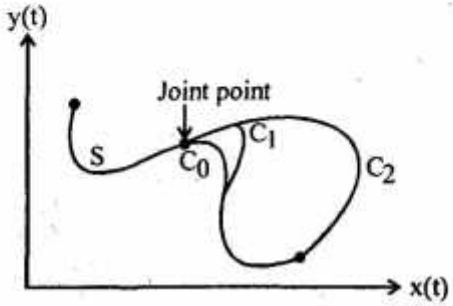


Two-point perspective. Lines parallel to y-axis do not converges.

Q. 7. (a) What do you mean by a Spline? How it differs from Hermite curves?

Ans. Splines :

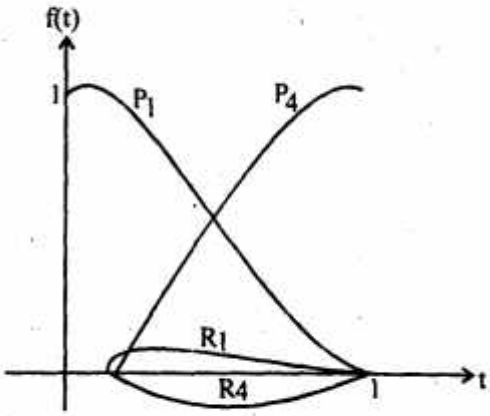
If two curve segments join together, the curve has G° geometric continuity. If the directions of the two segments' tangent vectors are equal at a join point, the curve has G' geometric continuity.



If the tangent vectors of two cubic curves segments are equal at the segments join point, the curve has first-degree continuity &-is said to be C' continuous. If the direction & magnitude of $d^n / dt^n[Q+(t)]$ through the n^{th} f derivative are equal at the join point, the curve is called c^n continuous.

A curve segment Q(t) is defined by constraints on endpoints, tangent vectors, & continuity between curve segments. Each cubic polynomial has four coefficients, so four constraints will be needed. The splines have C^1 & C^2 continuity at the join points & come close to their control points, but generally do not interpolate the points. The types of splines are uniform B-splines, non-uniform B-splines & B-splines.

Hermite curves : The Hermite form of the cubic polynomial curve segment is determined by constraints on the end points P_1 & P_4 & tangent vectors at the end-points R_1 & R_4.



For splines,

$$Q(t) = T.M.G$$

$$Q(t) = [x(t) \quad y(t) \quad z(t)]$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

$$Q(t) = T.M_H.G_H = B_H.G_H$$

$$= \left(2t^3 - 3t^2 + 1\right)P_1 + \left(-2t^3 + 3t^2\right)P_4 + \left(t^3 - 2t^2 + t\right)R_1 + \left(t^3 - t^2\right)R_4 .$$

Q. 7. (b) What is polygon mesh? How a polygon mesh is represented in memory?

Ans. Polygon mesh :

A polygon mesh is a collection of edges, vertices & polygons connected such that each edge is shared by at most two polygons. An edge connects two vertices & a polygon is a closed sequence of edges.

A polygon mesh can be represented in many ways. Two basic criteria, space & time, can be used to evaluate different representations.

Three polygon-mesh representations : Explicit, pointers to a vertex list & pointers to an edge list.

In explicit representation, each polygon is represented by a list of vertex coordinates :

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\ldots\ldots\ldots, (x_n, y_n, z_n))$$

The vertices are stored in the order in which they would be encountered traveling around the polygon. For a single polygon, this is space efficient; for a polygon mesh, however, much space is lost because the co-ordinates of shared vertices are duplicated. With this representation, displaying the mesh either as filled polygons or as polygon outlines necessitate transforming each vertex & clipping each edge of each polygon.

Polygons defined with pointers to a vertex list, the method used by SPHIGS, have each vertex in the polygon mesh stored just once, in the vertex list $P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots\ldots\ldots\ldots, (x_n, y_n, z_n))$. A polygon is defined by a list of indices into the vertex list.

Since each vertex is stored just once, considerable space is saved. On the other hand, it is difficult to find polygons that share an edge & shared polygon edges are drawn twice.

When defining polygons by pointers to an edge list. We have the vertex list V, but represent a polygon as a list of pointers not to vertex list, but rather to an edge list, in which each edge occurs just once. In turn, each edge in the edge list points to the two vertices in the vertex list defining the edge & also to the once or two polygons to which the edge belongs. Hence, we describe a polygon as $P = (E_1, \ldots\ldots, E_n)$ & an edge as

$E = (V_1, V_2, \ldots P_1, P_2)$. When an edge belongs to only one polygon, either P_1 or P_2 is null.

Q. 8. (a) How various visible surface algorithms take advantage of "coherence"? What are various types of coherence identified?

Ans. Coherence :

The degree to which parts of an environment or its projection exhibit local similarities. Environments typically contain objects whose properties vary smoothly from one part to another. It is the less frequent discontinuities in properties (such as depth, color & texture) & the effects that they produce in pictures, that less us distinguish between objects. We exploit coherence when we reuse calculations made for one part of the environment or picture for other nearby parts, either without changes or with incremental changes that are more efficient to make than recalculating the information from scratch.

Different kind of coherence :

1. Object coherence :

If one object is entirely separate from another, comparisons may need to be done only between the two objects & not between their component faces or edges.

2. Face coherence :

Surface properties typically vary smoothly across a face, allowing computations for one part of face to be modified incrementally to apply to adjacent parts.

3. Edge coherence :

An edge may change visibility only where it crosses behind a visible edge or penetrates a visible face.

4. Implied coherence :

If one planar face penetrates another, their line of intersection can be determined from the points of intersection.

5. Scan-line coherence :

The set of visible object spans determined for one scan line of an image typically differs little from the set

on the previous line.

6. Area coherence : A group of adjacent pixels is often covered by the same visible face.

7. Depth coherence : Adjacent parts of some surface are close in depth. Once the depth at one point of the surface is calculated, the depth of points on the rest of surface can be determined by simple equations.

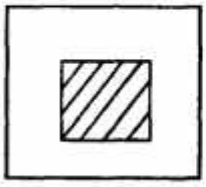
8. Frame coherence : Calculations made for one picture can be used for next in sequence.

Q. 8. (b) Write and explain the Warnock's algorithm for visible surface determination.

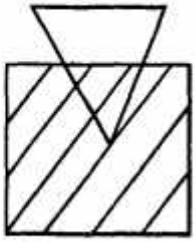
Ans. Warnock's algorithm :

The area-subdivision algorithm subdivides each area into four equal squares. At each stage in the ecursive-subdivision process, the projection of each polygon has one of our relationships to the area of nterest.

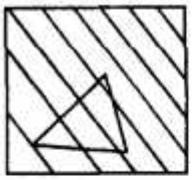
1. Surrounding polygons completely contain the area of interest.
2. Intersecting polygons intersect the area.
3. Contained polygons are completely inside the area.
4. Disjoint polygons are completely outside the area.



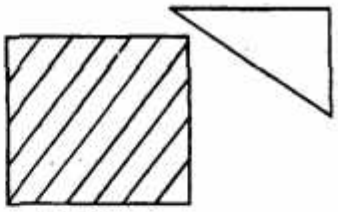
(a) Surrounding



(b) Intersecting



(c) Contained



(d) Disjoint

Two steps strategy :

1. Decide which projected polygons overlap a given area A on screen & are therefore potentially visible in that area.

2. In each area these polygons are further tested to determine which one will be visible within this area & should therefore be displayed. If a visibility decision can't be made, this screen area usually a rectangular window is further subdivided either until a visibility decision can't be made or until screen area is a single pixel.

Algorithm :

1. Initialize the area to be whole screen.

2. Create a PVPL (Potentially Visible Polygon List) with respect to an area, sorted on Z_{min} (smallest $Z_{coordinate}$ of the polygon within the area). Place the polygons in their appropriate categories. Remove polygons hidden by a surroundings polygon & remove disjoint polygons.

3. Perform the visibility decision tests :

(a) If the list is empty, set all pixels to background color.

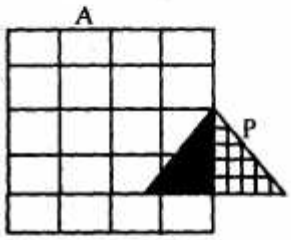
(b) If there is exactly one polygon in the list & it is classified as intersecting or contained, color (scan convert) the polygon & color the remaining area to the background color.

(c) If there is exactly one polygon on the list & it is a surrounding one, color the area the color of the surrounding polygon.

(d) If the area is the pixel (x, y) & neither a, b, nor C applies, compute the z-coordiante z(x, y) at pixel (x, y) of all polygons on the PVPL. The pixel is then set to the color of the polygon with the smallest z-

coordinate.

4. If none of the above cases has occurred, subdivide the screen area into four parts. For each area go to step number.



P = Polygon

A = Area of interest

Subdivision into 4 parts (squares)