2.

Solution.

Digital Differential Analyzer Algorithm is a line drawing algorithm that is based on incremental method or slope-intercept equation, which calculates all intermediate points over the interval between start and end points.

For any given x interval ∂x along a line, we can compute the corresponding y interval, ∂y, as

∂y = m∂x
=> ∂x = ∂y/m (m is slope)
PSUEDO-CODE
//Get the input of two end points (X1,Y1) and (X2,Y2). Calculate dx, dy.

dx = X2 - X1;
dy = Y2 – Y1;

A line is sampled at unit intervals in one coordinate and the corresponding integer values nearest to the line path are determined for the other coordinate.

If the slope is **more than 1**, we sample at unit y intervals (∂y = 1) and compute successive x values as:-

$$x_{k+1} = x_k + (1/ m)$$

// Depending upon absolute value of dx & dy, choose number of steps to put pixel as:
steps = abs(dy) > abs(dx) ? abs(dy) : abs(dx)
Subscript k takes integer values starting from 0, for the first point, and increase by 1 until the final endpoint is reached.
// Calculate the increment in x coordinate and y coordinate for each step
xIncrement = dx / (float) step;
yIncrement = dy / (float) step;
Because m can be any real number between 0.0 and 1.0, each calculated x value must be rounded to the nearest integer corresponding to a screen pixel position in the y column that we are processing.
These equations are based on the assumption that lines are to be processed from the left end point to the right end point. If this processing is reversed, so that the staring endpoint is at the right, then we have, ∂y = -1, therefore,

$$x_{k+1} = x_k – (1/m)$$

// Put the pixel by successfully incrementing x and y coordinates accordingly and complete the line drawing.

```
X = X1;
Y = Y1;
//for first starting pixel of line
putpixel (X,Y,COLOR);
for (int i = 1; i <= step; i++)
{
        putpixel (X,Y,COLOR);
        X += xIncrement;
        Y += yIncrement;
}Exit
```

3.

Solution. No. of control points = 4

$\Rightarrow$ No. of blending functions = 4

And these blending functions are –

$$B_{0,3}(u) = (1-u)^3$$
$$B_{1,3}(u) = 3u(1-u)^2$$
$$B_{2,3}(u) = 3u^2(1-u)$$
$$B_{3,3}(u) = u^3 \qquad (0 \le u \le 1)$$

$\therefore$ ***x*** **- component** of the Bezier curve is –

$$x(u) = x_0B_{0,3}(u) + x_1B_{1,3}(u) + x_2B_{2,3}(u) + x_3B_{3,3}(u)$$
$$= (1-u)^3 + 6u(1-u)^2 + 12u^2(1-u) + 6u^3 \qquad ...(1)$$

and **y – component** of the Bezier curve is –

$$y(u) = (1-u)^3 + 9u(1-u)^2 + 12u^2(1-u) + u^3 \qquad ...(2)$$

$\because$ u ranges from 0 to 1, so for midpoint of the curve, $u = 1/2$.

Putting $u = 1/2$ in equations (1) and (2) we get

Parametric midpoint = (27/8, 23/8)

Differentiate equation ...(1) *w.r.t.* u : –

$$\frac{dx}{du} = -3(1-u)^2 - 12u(1-u) + 6(1-u)^2 - 12u^2 + 24u(1-u) + 18u^2$$

Similarly,

$$\frac{dx}{du} = -3(1-u)^2\ u(1-u) + 9(1-u)^2 - 12u^2 + 24(1-u)u + 3u^2$$

$\therefore$ Gradient at any point is given by –

$$\left.\frac{dy}{dx}\right|_u = \left(\frac{dy}{du}\right) \Big/ \left(\frac{dx}{du}\right)\Big|_u$$

$\because$ $$\left.\frac{dy}{du}\right|_{u=\frac{1}{2}} = \frac{3}{4}$$

& $$\left.\frac{dx}{du}\right|_{u=\frac{1}{2}} = \frac{21}{4}$$

$\because$ $$\left.\frac{dy}{dx}\right|_{u=\frac{1}{2}} = \frac{\cancel{3}}{\cancel{4}} \times \frac{\cancel{4}}{\cancel{21}} = \frac{1}{7}$$

Hence proved.

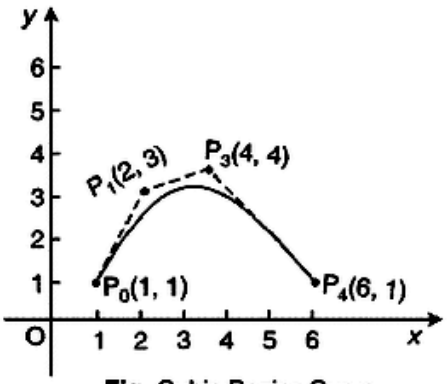

Fig. Cubic Bezier Curve

4.

Ans. Given: $ax + by + cz + d = 0$ be projection plane.

(p, q, r) direction of projection.

(x_0, y_0, z_0) point on object to project.

We start at (x_0, y_0, z_0) and travel along the line in direction (p, q, r) until the plane $ax + by + cz + d = 0$ is hit.

Now, the parametric equation of line is–

$$x = x_0 + pt \quad \ldots (1)$$

$$y = y_0 + qt \quad \ldots (2)$$

$$z = z_0 + rt \quad \ldots (3)$$

Where 't' is a parameter. At some value of t, when the plane equation is satisfied, we are on the projection plane

$$ax + by + cz + d = 0 \quad \ldots (4)$$

Putting (1), (2) and (3) in (4) we get–

$$a(x_0 + pt) + b(y_0 + qt) + c(z_0 + rt) + d = 0 \quad \ldots (5)$$

or $$ax_0 + by_0 + cz_0 + t(ap + bq + cr) + d = 0 \quad \ldots (6)$$

Solving for unknown parameter value, t we get–

$$t = -\left[\frac{ax_0 + by_0 + cz_0 + d}{ap + bq + cr}\right] \quad \ldots (7)$$

Where $ap + bq + cr \neq 0$

Putting equation (7) in (1), (2) and (3), we get:–

$$x_p = x_0 - p\left[\frac{ax_0 + by_0 + cz_0 + d}{ap + bq + cr}\right] \quad \ldots (8)$$

$$y_p = y_0 - p\left[\frac{ax_0 + by_0 + cz_0 + d}{ap + bq + cr}\right] \quad \ldots (9)$$

& $$z_p = z_0 - r\left[\frac{ax_0 + by_0 + cz_0 + d}{ap + bq + cr}\right] \quad \ldots (10)$$

Where (x_p, y_p, z_p) are projected points.

$\therefore$ In matrix form–

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

$\therefore$ $$m_{11} = (bq + cr) / (aq + bp + cr)$$

$$m_{13} = (-cp) / (ap + bq + cr)$$

$$
\begin{aligned}
m_{21} &= (-ap) / (ap + bq + cr) \\
m_{23} &= (-cq) / (ap + bq + cr) \\
m_{31} &= (-ar) / (ap + bq + cr) \\
m_{33} &= (ap + bq) / (ap + bq + cr) \\
m_{12} &= (-bq) / (aq + bq + cr) \\
m_{14} &= (-dq) / (ap + bq + cr) \\
m_{22} &= (aq + cr) / (ap + bq + cr) \\
m_{24} &= (-dq) / (ap + bq + cr) \\
m_{32} &= (-br) / (ap + bq + cr) \\
m_{34} &= (-dr) / (ap + bq + cr)
\end{aligned}
$$

5.
Solutions.

(a) What rendering technique handles reflections and refractions well? Environment Mapping(EM)

(b) What is the OpenGL name for a potential pixel? Fragment

(c) What feature in OpenGL is used to display the closest object when several objects overlap the same pixel? Z-Buffer

(d) What's the 2-word name for the technique for storing the graphics on the graphics card (if there's space) so that it does not have to be repeatedly sent down the network each time the window is redisplayed? Display List