Question Bank Solution of CAT-1 COMPUTER GRAPHICS

By **JAAT**

FOR MORE INFO ABOUT US VISIT: https://youtu.be/0B5ctkBttDY

UNIT-1

Q1. An e-publishing company is in the process of converting e-book in the form of document image to text. Discuss on the challenges faced by the company in implementing the process.

ANS The challenges faced by an e-publishing company in implementing the process of converting e-books in the form of document image to text include issues related to the quality of the source material, text recognition errors, formatting errors, language support, time and resource constraints, and the need for trained personnel.

Q2. Compute the resolution of a 2 X 2 inch image that has 512 X 512 pixels.

ANS The resolution of an image is the number of pixels per unit of measurement, usually expressed in pixels per inch (ppi). To compute the resolution of a 2 X 2 inch image that has 512 X 512 pixels, we can use the following formula:

Resolution = Number of Pixels / Size in Inches

In this case, the number of pixels is 512 X 512 = 262,144, and the size of the image is 2 X 2 inches. Therefore, the resolution of the image is:

Resolution = 262,144 / (2 X 2) = 65,536 pixels per inch (ppi)

So the resolution of the image is 65,536 ppi.

Q3. If we want to resize a 1024 X 768 image to one that is 640 pixels wide the same aspect ratio, find out the height of the resized image?

ANS To maintain the same aspect ratio, the height of the resized image can be found using the following formula:

new height = (original height * new width) / original width

Substituting the given values, we get:

new height = (768 * 640) / 1024

Simplifying this expression, we get:

new height = 480

Therefore, the height of the resized image would be 480 pixels.

Q4. If we use 2-byte pixel values in a 24 bit lookup table representation, Find how many bytes does the lookup table occupy?

ANS In a 24-bit lookup table, each pixel is represented using 3 bytes (i.e., 24 bits), where each byte corresponds to the intensity value of one of the primary colors (red, green, and blue).

If we use 2-byte pixel values, each pixel would occupy 2 x 3 = 6 bytes. Therefore, the lookup table would occupy:

256 x 6 = 1536 bytes

This is because the lookup table contains 256 entries (since 24 bits can represent 256 different intensity values for each primary color), and each entry corresponds to a pixel value of 6 bytes.

Hence, the lookup table would occupy 1536 bytes if we use 2-byte pixel values in a 24-bit lookup table representation.

Q5. List the steps required to plot a line whose slope is between 0 degree and 45 degree using the slope intercept equation?

ANS To plot a line whose slope is between 0 and 45 degrees using the slope-intercept equation (y = mx + b), we can follow these steps:

1. Choose a suitable range for the x-axis and y-axis that will include all the points we want to plot.
2. Choose a value for the y-intercept (b) that will position the line correctly on the y-axis.
3. Choose a value for the slope (m) that corresponds to the angle we want the line to make with the x-axis.
4. Calculate the y-value for several x-values using the equation y = mx + b. These values will form the set of points that define the line.
5. Plot the points on the graph and connect them with a straight line.

For example, let's say we want to plot a line with a slope of 1/3 and a y-intercept of 2. We could follow these steps:

6. Choose a range for the x-axis and y-axis, such as x = [-10, 10] and y = [-5, 10].
7. Choose a y-intercept of 2.
8. Choose a slope of 1/3.
9. Calculate the y-values for several x-values using the equation y = (1/3)x + 2. For example, when x = -9, y = -1; when x = -6, y = 0; when x = -3, y = 1; when x = 0, y = 2; and so on.
10. Plot the points (-9, -1), (-6, 0), (-3, 1), (0, 2), (3, 3), (6, 4), and (9, 5) on the graph, and connect them with a straight line.

This would result in a line with a slope of 1/3 and a y-intercept of 2, making an angle of approximately 18 degrees with the x-axis.

Q6. List steps are required to plot a line whose slope is between 0 degree and 45 degree using Bresenham's Algorithm?

ANS Bresenham's algorithm is an efficient method for drawing lines on a raster display. To plot a line whose slope is between 0 degree and 45 degree using Bresenham's Algorithm, the following steps are required:

Determine the two endpoints of the line (x1, y1) and (x2, y2).

Calculate the slope of the line using the formula: m = (y2 - y1) / (x2 - x1).

If the slope is less than 1, set dx to 1 and dy to the rounded value of m. If the slope is greater than or equal to 1, set dy to 1 and dx to the rounded value of 1/m.

Set the error term (e) to 0.

Set the initial values of x and y to the first endpoint (x1, y1).

Plot the first pixel at (x, y).

While x is less than x2, do the following: a. Add dx to x. b. Add the rounded value of m times dx to y. c. Add the error term to y. d. If the error term is greater than or equal to 0.5, add 1 to y and subtract 1 from the error term. e. Plot the pixel at (x, y).

Repeat step 7 until x reaches x2.

By following these steps, a line whose slope is between 0 degree and 45 degree can be plotted using Bresenham's Algorithm.

Q7. Calculate the pixel location approximating the first octant of a circle having centre at (4, 5) and radius 4 units using Bresenham algorithm.

ANS To plot the first octant of a circle using Bresenham's algorithm, we need to determine which pixels to plot at each step by calculating the error term and adjusting the coordinates accordingly. The first octant of a circle is the region between 0 and 45 degrees, so we only need to plot pixels in the positive x and y directions.

The centre of the circle is (4, 5) and the radius is 4 units. We can use the standard form of the equation for a circle to calculate the coordinates of the points on the circle:

$(x - 4)^2 + (y - 5)^2 = 4^2$

Expanding and simplifying:

$x^2 - 8x + 16 + y^2 - 10y + 25 = 16$

x^2 - 8x + y^2 - 10y + 25 = 0

We can use Bresenham's algorithm to plot the circle by starting at the top point of the first octant, which is (4, 9), and moving clockwise. The steps are as follows:

Set the initial values of x and y to the first point, (4, 9).

Plot the first pixel at (x, y).

Calculate the error term e = 2 * (x - 4) + 1.

While x is less than y, do the following: a. If the error term is greater than or equal to 0, subtract 2y from e and subtract 8 from y. b. Add 1 to x and add 2 to the error term. c. Plot the pixel at (x, y).

Repeat step 4 until x is greater than or equal to y.

The resulting pixels to be plotted in the first octant of the circle are:

(4, 9) (5, 9) (6, 8) (7, 8) (8, 7) (8, 6)

Therefore, the pixel locations approximating the first octant of the circle having centre at (4, 5) and radius 4 units using Bresenham algorithm are (4, 9), (5, 9), (6, 8), (7, 8), (8, 7), and (8, 6).

Q8. List the steps are required to generate a circle using the polynomial method?

ANS The polynomial method for generating a circle involves the following two steps:

Determine the center of the circle and its radius.

Choose a value of x, calculate the corresponding value of y using the equation of a circle (y = sqrt(r^2 - (x - center_x)^2) + center_y), and plot the resulting point (x, y). Repeat for each value of x in the range of the circle.

Q9. List the steps are required to generate a circle using the trigonometric method?

ANS The trigonometric method for generating a circle involves the following two steps:

Determine the center of the circle and its radius.

Choose a value of the angle theta (in radians) from 0 to 2*pi and calculate the corresponding values of x and y using the equations x = r*cos(theta) + center_x and y = r*sin(theta) + center_y. Plot the resulting point (x, y). Repeat for a range of values of theta to generate the entire circle.

Q10. Define Computer Graphics.

ANS Computer Graphics refers to the creation, manipulation, and display of visual content using computers. It involves the use of algorithms and mathematical equations to generate images, animations, and videos on a computer screen or other digital display devices. Computer graphics are used in a variety of applications, including video games, films, architectural visualizations, medical imaging, and scientific simulations.

Q11. Write bresenham's line drawing algorithm an d trace the algorithm for the given points (2, 1) to (10, 12)

ANS Bresenham's line drawing algorithm is a computer algorithm used to draw a line between two given points on a computer screen or other digital display devices. The algorithm is based on the concept of error accumulation and uses only integer arithmetic to calculate the pixels to be drawn.

Here is the Bresenham's line drawing algorithm:

Initialize the starting point (x1, y1) and ending point (x2, y2) of the line.

Calculate the differences between x2 and x1 (dx) and y2 and y1 (dy).

Calculate the sign of dx and dy, and set the step size accordingly: if dx is negative, the step in x direction is -1, otherwise, it is 1. Similarly, if dy is negative, the step in y direction is -1, otherwise, it is 1.

Calculate the initial value of the error term e = dx - dy.

Loop through the line from x1 to x2 and for each x, calculate the corresponding y value based on the error term: a. If e is greater than or equal to 0, increment y by the step in y direction and subtract dx from e. b. If e is less than 0, don't change y and add dy to e.

Plot the pixel at (x, y) for each x.

Now, let's trace the algorithm for the given points (2, 1) to (10, 12):

The starting point is (2, 1) and the ending point is (10, 12).

dx = 8, dy = 11.

dx is positive, so the step in x direction is 1. dy is positive, so the step in y direction is 1.

e = dx - dy = -3.

Loop through the line from x = 2 to x = 10: a. For x = 2, y = 1. Plot the pixel at (2, 1). b. For x = 3, e = e + 2*dy = 19. y = 2. Plot the pixel at (3, 2). c. For x = 4, e = e - 2*dx = 11. y = 3. Plot the pixel at (4, 3). d. For x = 5, e = e + 2*dy = 23. y = 4. Plot the pixel at (5, 4). e. For x = 6, e = e - 2*dx = 15. y = 5. Plot the pixel at (6, 5). f. For x = 7, e = e + 2*dy = 27. y = 6. Plot the pixel at (7, 6). g. For x = 8, e = e - 2*dx = 19. y = 7. Plot the pixel at (8, 7). h. For x = 9, e = e + 2*dy = 31. y = 8. Plot the pixel at (9, 8). i. For x = 10, e = e - 2*dx = 23. y = 9. Plot the pixel at (10, 9).

The line from (2, 1) to (10, 12) is drawn using Bresenham's line drawing algorithm.

Q12 Consider a raster system with a resolution of 1024 X 1024. Calculate the size of raster needed to store 4 bit per pixel.

ANS To store 4 bits per pixel, we need a color depth of 2^4 = 16 colors. Each pixel can be represented by 4 bits, which means that we need 4/8 = 0.5 bytes or 0.5 * 1024 = 512 bits to store one pixel.

Since the raster system has a resolution of 1024 X 1024, the total number of pixels in the image is 1024 * 1024 = 1,048,576. Therefore, the size of raster needed to store 4 bit per pixel is:

1,048,576 pixels X 0.5 bytes/pixel = 524,288 bytes or 0.5 megabytes.

Q13. Define halftone image?

ANS A halftone image is a type of image that uses small dots of varying sizes and densities to create the illusion of continuous tone. It is a technique commonly used in printing, where the image is reproduced by printing dots of ink in different sizes and spacing to simulate the appearance of a continuous-tone image. Halftone images are often used in newspapers, magazines, and other printed materials where high-quality images are required but full-color printing is not practical or affordable.

Q14. What is aspect ratio?

ANS Aspect ratio refers to the proportional relationship between the width and height of an image or screen. It is expressed as the ratio of the image or screen's width to its height. For example, a standard aspect ratio for television screens is 16:9, which means that the screen is 16 units wide for every 9 units of height. Aspect ratio is an important consideration in graphic design and video production, as it can affect the composition and visual impact of the image or video.

Q15. Using the Bresenham line generation algorithm, digitize the line with end points (20, 10) and (30, 18).

To use the Bresenham line generation algorithm to digitize the line with endpoints (20, 10) and (30, 18), we can follow these steps:

Calculate the change in x and y between the two endpoints: dx = 30 - 20 = 10 dy = 18 - 10 = 8

Determine the sign of dx and dy to determine the direction of the line: sx = sign(dx) = 1 (because dx is positive) sy = sign(dy) = 1 (because dy is positive)

Calculate the absolute values of dx and dy: dx = abs(dx) = 10 dy = abs(dy) = 8

Calculate the decision parameter: d = 2*dy - dx

Set the starting point to the first endpoint (20, 10).

For each x from the starting x-coordinate (20) to the ending x-coordinate (30), do the following:

If d is less than 0, the next pixel is (x+1, y), and the decision parameter is updated as follows: d = d + 2*dy

If d is greater than or equal to 0, the next pixel is (x+1, y+1), and the decision parameter is updated as follows: d = d + 2*(dy - dx)

Repeat step 6 until the ending x-coordinate (30) is reached.

Using these steps, we can generate the following digitized line:

```
scss
```

```
(20, 10)
(21, 11)
(22, 12)
(23, 13)
(24, 14)
(25, 15)
(26, 16)
(27, 17)
(28, 18)
(29, 18)
(30, 18)
```

Thus, the digitized line with endpoints (20, 10) and (30, 18) using the Bresenham line generation algorithm is the set of pixels above.

Q16. Compare Bresenham line generation with DDA line Generation.

ANS Both Bresenham's and DDA (Digital Differential Analyzer) algorithms are used for line generation in computer graphics. Here's how they compare:

Accuracy: DDA algorithm produces lines with decent accuracy but sometimes the endpoint of the line may not match the exact desired coordinate. Bresenham's algorithm provides higher accuracy by using integer arithmetic and minimizing round-off errors.

Speed: Bresenham's algorithm is generally faster than DDA algorithm as it only involves integer arithmetic and avoids time-consuming floating-point calculations.

Line direction: DDA algorithm can handle any line direction (i.e., both positive and negative slopes) by simply incrementing or decrementing coordinates in each step. In contrast, Bresenham's algorithm works best for lines with slopes between 0 and 1.

Memory usage: DDA algorithm requires storage of real numbers for each coordinate point, whereas Bresenham's algorithm only requires integer variables. Thus, Bresenham's algorithm is more memory-efficient.

Implementation complexity: DDA algorithm is relatively easy to implement as it only involves simple arithmetic operations. Bresenham's algorithm is slightly more complex and involves more calculations, such as decision variable computation.

Round-off errors: DDA algorithm suffers from round-off errors due to floating-point calculations, whereas Bresenham's algorithm minimizes round-off errors by using integer arithmetic.

In summary, both algorithms have their own advantages and disadvantages. Bresenham's algorithm provides higher accuracy and is faster and more memory-efficient, but works best for lines with slopes between 0 and 1. DDA algorithm is easier to implement and can handle any line direction, but suffers from round-off errors and can be slower due to floating-point calculations.

Q17. Using the Bresenham line generation algorithm, digitize the line with end points (15, 5) and (25, 13).

ANS The Bresenham line generation algorithm is a popular algorithm used to draw lines on a grid by selecting the pixels closest to the ideal line. Here are the steps to digitize the line with endpoints (15, 5) and (25, 13) using the Bresenham line generation algorithm:

Step 1: Calculate the difference between the x-coordinates and y-coordinates of the two endpoints. dx = 25 - 15 = 10 dy = 13 - 5 = 8

Step 2: Calculate the sign of the differences. sx = sign(dx) = 1 (since dx is positive) sy = sign(dy) = 1 (since dy is positive)

Step 3: Initialize the error term and the current pixel position. err = 0 x = 15 y = 5

Step 4: Digitize the line by plotting the pixels that are closest to the ideal line using the Bresenham algorithm. Repeat until the endpoint (25, 13) is reached. for i in range(dx): plot(x, y) err += dy if (2 * err) >= dx: y += 1 err -= dx x += 1

Here, the "plot" function is used to indicate that a pixel should be drawn at the current position (x, y). The "if" statement checks whether the error term has reached or exceeded half the x-difference, and if so, increments the y-coordinate and adjusts the error term.

Using this algorithm, we can generate the following points along the line from (15, 5) to (25, 13): (15, 5), (16, 6), (17, 6), (18, 7), (19, 8), (20, 9), (21, 10), (22, 11), (23, 12), (24, 13), (25, 13)

So, the digitized line with endpoints (15, 5) and (25, 13) would pass through these points.

Q18. Rasterize the line from (-1,1) to (5, -8) using Bresenham ’s line drawing Algorithm.

ANS here are the steps to rasterize the line from (-1, 1) to (5, -8) using the Bresenham's line drawing algorithm:

Step 1: Calculate the difference between the x-coordinates and y-coordinates of the two endpoints. dx = 5 - (-1) = 6 dy = -8 - 1 = -9

Step 2: Calculate the sign of the differences. sx = sign(dx) = 1 (since dx is positive) sy = sign(dy) = -1 (since dy is negative)

Step 3: Take the absolute value of dy, double it, and subtract it from dx to get the initial error term. err = 2 * abs(dy) - dx

Step 4: Initialize the current pixel position to the starting point (-1, 1). x = -1 y = 1

Step 5: Digitize the line by plotting the pixels that are closest to the ideal line using the Bresenham algorithm. Repeat until the endpoint (5, -8) is reached. for i in range(abs(dx)): plot(x, y) while err >= 0: y += sy err -= 2 * dx x += sx err += 2 * abs(dy)

Here, the "plot" function is used to indicate that a pixel should be drawn at the current position (x, y). The inner "while" loop adjusts the y-coordinate and error term until the error term becomes less than zero.

Using this algorithm, we can generate the following points along the line from (-1, 1) to (5, -8): (-1, 1), (0, 0), (1, -1), (2, -2), (3, -3), (4, -4), (5, -5), (6, -6), (7, -7), (8, -8), (9, -8)

So, the digitized line with endpoints (-1, 1) and (5, -8) would pass through these points.

Q19. Given radius r=10 determine positions along with the circle octants in 1st Quadrant from x=0 to x=y.

ANS The circle with radius r=10 and center at the origin (0,0) can be represented by the equation:

x^2 + y^2 = r^2 or x^2 + y^2 = 100

In the 1st quadrant, x and y are positive, and x is less than or equal to y along the line x=y. Therefore, we can solve the equation for y in terms of x and divide the 1st quadrant into 8 octants based on the sign of x and y.

Solving the equation for y, we get:

y = sqrt(100 - x^2)

Along the line x=y, we have:

y = sqrt(100 - y^2) y^2 + y^4 = 100

Solving this equation for y, we get:

y = 5.303 or y = -5.303

Since y is positive in the 1st quadrant, we take y = 5.303 as the solution.

Using this value of y, we can determine the positions along with the circle octants in the 1st quadrant from x=0 to x=y as follows:

Octant 1: (0,10) to (5.303,5.303) Octant 2: (0,10) to (5.303,-5.303) Octant 3: (0,10) to (-5.303,-5.303) Octant 4: (0,10) to (-5.303,5.303) Octant 5: (0,10) to (-10,0) Octant 6: (0,10) to (-5.303,-5.303) Octant 7: (0,10) to (5.303,-5.303) Octant 8: (0,10) to (10,0)

Q20. Do you need to generate the full circumference of the circle using the algorithm, or can we generate it in a quadrant or octant only and then use it to produce the rest of the circumference?

ANS We can generate the circle in a quadrant or octant only and then use it to produce the rest of the circumference.

Q21. Given a circle radius r=5 determine positions along the circle octants in 1st Quadrant from x=0 to x=y.

ANS The circle with radius r=5 and center at the origin (0,0) can be represented by the equation:

x^2 + y^2 = r^2 or x^2 + y^2 = 25

In the 1st quadrant, x and y are positive, and x is less than or equal to y along the line x=y. Therefore, we can solve the equation for y in terms of x and divide the 1st quadrant into 8 octants based on the sign of x and y.

Solving the equation for y, we get:

y = sqrt(25 - x^2)

Along the line x=y, we have:

y = sqrt(25 - y^2) y^2 + y^4 = 25

Solving this equation for y, we get:

y = 3.536 or y = -3.536

Since y is positive in the 1st quadrant, we take y = 3.536 as the solution.

Using this value of y, we can determine the positions along the circle octants in the 1st quadrant from x=0 to x=y as follows:

Octant 1: (0,5) to (3.536,3.536) Octant 2: (0,5) to (3.536,-3.536) Octant 3: (0,5) to (-3.536,-3.536) Octant 4: (0,5) to (-3.536,3.536) Octant 5: (0,5) to (-5,0) Octant 6: (0,5) to (-3.536,-3.536) Octant 7: (0,5) to (3.536,-3.536) Octant 8: (0,5) to (5,0)

Q22. Using midpoint circle generation algorithm, compute the coordinates of points that lie on the circumference of the circle with radius 5 and center as (7,7).

ANS The midpoint circle generation algorithm can be used to generate the coordinates of points that lie on the circumference of a circle. The algorithm works by iteratively calculating the points on the circle using the midpoint between two previously calculated points.

The equation for a circle with center (a,b) and radius r is:

(x-a)^2 + (y-b)^2 = r^2

Using this equation and the midpoint circle generation algorithm, we can generate the coordinates of points that lie on the circumference of the circle with radius 5 and center at (7,7).

Starting from the point (7, 12), which lies on the top of the circle, we can use the algorithm to generate the remaining points on the circle. The steps of the algorithm are as follows:

Initialize the coordinates of the first point on the circle: (x0, y0) = (7, 12).

Calculate the initial value of the decision parameter: p0 = 5/4 - r.

Initialize the values of x and y to x0 and y0.

Repeat until x <= y: a. If p < 0, update p and x as follows: i. p = p + 2x + 1 ii. x = x + 1 b. If p >= 0, update p, x, and y as follows: i. p = p + 2x - 2y + 1 ii. x = x + 1 iii. y = y - 1 c. Generate the coordinates of the points on the circle using the symmetry properties of the circle. In the 1st octant (x>=0, y>=0), we have: i. (x, y) ii. (y, x) iii. (-x, y) iv. (-y, x) v. (-x, -y) vi. (-y, -x) vii. (x, -y) viii. (y, -x)

Using the algorithm, we can calculate the coordinates of points that lie on the circumference of the circle with radius 5 and center at (7,7) as follows:

Initialize the coordinates of the first point on the circle: (x0, y0) = (7, 12).

Calculate the initial value of the decision parameter: p0 = 5/4 - 5 = -3/4.

Initialize the values of x and y to x0 and y0: x = 7, y = 12.

Repeat until x <= y: a. If p < 0, update p and x as follows: i. p = p + 2x + 1 = p + 15 ii. x = x + 1 = 8 b. If p >= 0, update p, x, and y as follows: i. p = p + 2x - 2y + 1 = p + 22 ii. x = x + 1 = 9 iii. y = y - 1 = 11 c. Generate the coordinates of the points on the circle using the symmetry properties of the circle: i. (9, 11) ii. (11, 9) iii. (-9, 11) iv. (-11, 9) v. (-9, -11) vi. (-11, -9) vii. (9, -11) viii. (11, -9)

Therefore, the

Q23. Distinguish between scan line polygon fill and seed fill (flood fill) algorithm.

ANS Scan line polygon fill and seed fill (flood fill) are two different algorithms used for filling a closed polygon with a color. Here are the main differences between these two algorithms:

Scan line polygon fill algorithm:

- This algorithm fills a polygon by scanning horizontal lines across the polygon and filling in the pixels that lie between the left and right edges of the polygon.
- It requires the edges of the polygon to be stored as a set of connected line segments.
- It is relatively faster than the seed fill algorithm for complex polygons.
- However, it can be more difficult to implement as it requires the edges to be sorted and handled carefully to avoid artifacts such as gaps or overlaps in the filled region.

Seed fill (flood fill) algorithm:

- This algorithm fills a polygon by starting at a known point inside the polygon (known as the seed point) and filling in adjacent pixels until the entire polygon is filled.
- It does not require the edges of the polygon to be known or stored.
- It is generally slower than scan line polygon fill for complex polygons because it may need to fill in many pixels.
- It can be easier to implement than scan line polygon fill as it is a straightforward recursive or iterative process.

In summary, scan line polygon fill and seed fill algorithms are both used for filling closed polygons with a color, but they use different approaches to achieve this. Scan line polygon fill scans horizontal lines across the polygon and fills in pixels between the left and right edges, while seed fill starts at a known point and fills in adjacent pixels until the entire polygon is filled.

Q24. How vector CRT is different from Raster CRT.

ANS Vector CRT and Raster CRT are two types of CRT displays used for displaying images and videos. Vector CRT uses a vector graphics system to draw precise graphics using an electron beam, while Raster CRT uses a raster graphics system to paint images on the screen one line at a time, from top to bottom. In summary, the main difference between the two is that vector CRT uses vector graphics, while raster CRT uses raster graphics.

Q26. Consider a Non -Interlaced raster system with resolution of 1280 By 1024, a refresh rate of 60 Hz, a horizontal retrace time of 5 Microseconds and a vertical retrace time of 500 µs.

What is the fraction of the total refresh time per frame spent in horizontal retrace of the electron beam?

ANS The fraction of the total refresh time per frame spent in horizontal retrace of the electron beam in the given Non-Interlaced raster system is approximately 30.7%, as calculated in the previous answer.

Q27. Find the equation of the line y'=mx'+b in xy coordinates if the x'y' coordinate system results from a 90 degree rotation of the xy coordinate system.

ANS To find the equation of the line y'=mx'+b in the xy coordinate system after a 90 degree rotation, we need to use a rotation matrix.

Let R be the 2x2 rotation matrix that rotates a point (x,y) in the xy coordinate system by 90 degrees counterclockwise to get the point (x',y') in the x'y' coordinate system:

```
CSSCopy code

R = [0 -1]
    [1  0]
```

To convert the equation y'=mx'+b to the xy coordinate system, we need to apply the inverse rotation matrix:

```
CSSCopy code

R^-1 = [ 0 1]
       [-1 0]
```

Multiplying both sides of y'=mx'+b by R^-1 gives:

```
CSSCopy code

[ 0 1] [y']   [b]
[-1 0] [x'] = [mx']
```

Simplifying the matrix multiplication, we get:

```
CSSCopy code

y = -x' m + b
```

So the equation of the line y'=mx'+b in the xy coordinate system after the 90 degree rotation is y = -x m + b.

Q28. What are the criteria of generating a straight line on raster scan display device?

ANS A raster scan display device generates images by scanning the electron beam across the screen from left to right and top to bottom, illuminating pixels as it goes. To generate a straight line on a raster scan display device, the following criteria must be met:

The line must be defined by two endpoints, which are located at integer pixel coordinates. This is because the electron beam can only illuminate whole pixels, not fractional ones.

The line must not have a slope greater than 1 or less than -1. This is because the electron beam scans across the screen horizontally, illuminating one row of pixels at a time. If the line's slope is greater than 1 or less than -1, there will be rows where no pixels are illuminated, resulting in a stair-step effect.

The line must be drawn using an algorithm that takes into account the discrete nature of the pixels. One commonly used algorithm is Bresenham's line algorithm, which determines which pixels to illuminate based on the error between the true line and the nearest pixel. This algorithm ensures that the line appears straight and smooth, without any gaps or overlaps between pixels.

By meeting these criteria, it is possible to generate straight lines on a raster scan display device with high accuracy and precision.

Q29. Explain shadow mask and beam penetration method.

ANS Shadow mask and beam penetration method are two different techniques used to produce color images on CRT (Cathode Ray Tube) displays.

Shadow mask is a technique where a shadow mask is placed between the electron guns and the phosphor coating on the inside of the screen. The mask is a thin sheet of metal with holes punched in it, and each hole corresponds to a different color of phosphor. The electron guns fire beams of electrons through the holes in the shadow mask, which strike the phosphor and cause it to emit light. By adjusting the intensity of the electron beams and the size and placement of the holes in the shadow mask, different colors can be produced on the screen. Shadow mask is a simpler and more reliable method, but can suffer from distortion or misalignment if the shadow mask is damaged or poorly manufactured.

Beam penetration method is another technique used to produce color images on CRT displays. In this method, a single electron gun fires a beam of electrons at the screen, which passes through a series of color filters before striking the phosphor coating. Each filter allows only one color of light to pass through, so by rapidly switching between different filters as the electron beam scans across the screen, different colors can be produced. This method can produce brighter and more

vivid colors, but is more complex and expensive to implement. Precise timing and control is required, as the filters must be switched on and off very quickly to produce a seamless image.

In summary, both shadow mask and beam penetration method are used to produce color images on CRT displays. Shadow mask is simpler and more reliable, while beam penetration method can produce brighter and more vivid colors but is more complex and expensive to implement. The choice of which technique to use depends on the specific requirements and constraints of the display application.

Q30. Explain Scan line fill polygon filling algorithm.

ANS Scan line polygon fill algorithm is a technique used to fill the interior of a closed polygon with a specified color. The algorithm works by dividing the polygon into a series of horizontal scan lines and determining which pixels intersect with the interior of the polygon on each line. The pixels that intersect with the polygon are then filled with the desired color.

The scan line algorithm proceeds as follows:

Sort the vertices of the polygon by their y-coordinates. This creates a list of edges that intersect the scan lines, ordered by the y-coordinate of their lowest endpoint.

Initialize an active edge list (AEL) and an edge table (ET) for each scan line. The AEL contains edges that intersect the current scan line, while the ET contains all the edges of the polygon.

Iterate over each scan line from the bottom to the top of the polygon. For each scan line, add any edges from the ET that intersect the scan line to the AEL. Remove any edges from the AEL that have reached their highest endpoint.

Sort the edges in the AEL by their x-coordinate at the intersection with the current scan line.

Fill the pixels between each pair of adjacent edges in the AEL. The pixels are filled using the desired color.

Repeat steps 3-5 for each subsequent scan line until the entire polygon has been filled.

This algorithm is efficient because it only processes the pixels that intersect with the polygon, rather than filling the entire area of the polygon. It is also capable of filling polygons with holes, as long as the edges of the holes are defined in the ET with the opposite winding order. The scan line algorithm is widely used in computer graphics and is a common technique for filling polygons in 2D and 3D applications.

Q31. Explain midpoint ellipse algorithm.

ANS The midpoint ellipse algorithm is a method for drawing ellipses using the midpoint algorithm. It works by generating points along the perimeter of the ellipse by incrementally stepping through each quadrant of the ellipse, starting from the origin and moving outward. The algorithm calculates the coordinates of the points along the perimeter using the midpoint formula, which takes into account the distance from the center of the ellipse and the curvature of the ellipse at each point. The midpoint ellipse algorithm is a fast and efficient way to draw ellipses on a raster display.

Q32. Differentiate emissive and non -emissive displays?

ANS Emissive displays produce their own light, while non-emissive displays rely on external light sources.

Q33. Explain: Sampling and quantization

ANS Sampling is the process of converting a continuous signal into a discrete signal by taking measurements at regular intervals. Quantization is the process of converting the continuous amplitude range of the signal into a finite set of discrete values.

Q34. State whether the given statement is true or false: “Fluorescence is the term used to describe the light off by a phosphor after it has been exposed to an electron beam”. Explain your answer.

ANS The given statement is false.

Fluorescence is a phenomenon in which a material absorbs light energy of a particular wavelength and re-emits it as light of a longer wavelength. This process occurs almost instantaneously and does not require exposure to an electron beam.

On the other hand, the term used to describe the light emitted by a phosphor after it has been exposed to an electron beam is called "phosphorescence". When an electron beam strikes a phosphor, the energy of the electrons is absorbed by the atoms in the phosphor material. This energy causes the atoms to become excited and move to a higher energy state. When these atoms return to their normal energy state, they release the excess energy in the form of light. This process is called phosphorescence.

Therefore, the given statement is false, as fluorescence and phosphorescence are two different phenomena with different underlying mechanisms.

Q35. If a boundary is 8 -connected, can 8 -boundary fill algorithm be used to fill the region bounded by that boundary? If no, Why?

ANS Yes, the 8-boundary fill algorithm can be used to fill the region bounded by an 8-connected boundary.

In 8-connected boundary fill, a pixel is filled if it is connected to any of its 8 neighboring pixels. Therefore, this algorithm can accurately fill regions bounded by an 8-connected boundary, which includes diagonally connected pixels as well as horizontally and vertically connected pixels.

On the other hand, if the boundary is 4-connected, which means a pixel is connected to only its horizontally and vertically adjacent pixels, then using the 8-boundary fill algorithm may lead to overfilling or underfilling of the region, as it may not account for the diagonally connected pixels.

In summary, the 8-boundary fill algorithm can be used for filling regions bounded by an 8-connected boundary, but not for filling regions bounded by a 4-connected boundary.

Q36. Define seed -fill algorithms? Write 8 -connected region filling algorithm? Out of 4 -connected and 8 -connected seed fill algorithm, which algorithm would you use to fill 8 -connected boundary region?

ANS Seed-fill algorithms are a type of region filling algorithm that starts from a seed point, which is a point known to be inside the region to be filled. The algorithm then examines neighboring pixels to determine whether they should also be included in the region.

The 8-connected region filling algorithm is a type of seed-fill algorithm that examines all 8 neighboring pixels of each point as it fills the region. The algorithm starts by setting the color of the seed point to the desired fill color and then examines the eight neighboring pixels. If a neighboring pixel is of the same color as the seed point, it is also colored with the fill color, and its eight neighbors are examined in turn. The process continues until all pixels connected to the seed point have been colored.

Out of the 4-connected and 8-connected seed-fill algorithms, the 8-connected algorithm should be used to fill an 8-connected boundary region, as it takes into account all 8 neighboring pixels, including the diagonally connected pixels, while the 4-connected algorithm only considers the horizontally and vertically adjacent pixels.

Q37. Write are the principal of vanishing point?

ANS The principle of vanishing point is based on the fact that parallel lines appear to converge at a single point in the distance, which is known as the vanishing point. The principle is used in art and design to create the illusion of depth and distance in a two-dimensional image.

The principle of vanishing point is based on three main principles:

Parallel lines appear to converge: When two parallel lines extend into the distance, they appear to get closer and closer until they eventually converge at a single point. This is known as the vanishing point.

The vanishing point is on the horizon line: The horizon line is a horizontal line that runs across the image at the eye level of the viewer. The vanishing point is always located on the horizon line.

The size and spacing of objects change with distance: As objects move further away from the viewer, they appear smaller and more closely spaced. This principle is known as linear perspective, and it is based on the fact that the eye can only perceive a limited range of depth and distance.

By using the principle of vanishing point and linear perspective, artists and designers can create the illusion of depth and distance in their work, and create more realistic and engaging images.

Q38. Explain in detail working of shadow mask and beam penetration CRT.

ANS It is not possible to explain the working of shadow mask and beam penetration CRT in detail in just 2 marks. Both shadow mask and beam penetration CRTs are complex technologies that involve electron beams, phosphors, masks, and other components to create an image on a screen. A detailed explanation of their workings would require a much longer answer.

Q39. List steps are required to plot a line whose slope is between 0 to 60 degree using Bresenham's method?

ANS The steps required to plot a line using Bresenham's method when the slope is between 0 and 60 degrees are:

Determine the two endpoints of the line in pixel coordinates (x1,y1) and (x2,y2).

Calculate the difference in x and y coordinates between the endpoints: dx = x2 - x1 and dy = y2 - y1.

If the slope m = dy/dx is greater than 1, swap the x and y coordinates of the endpoints.

If x1 > x2, swap the endpoints.

Set the starting point to (x1, y1) and calculate the initial decision parameter p0 = 2dy - dx.

For each x-coordinate along the line, starting at x1+1, do the following: a. If p < 0, set the next point to (x+1, y) and update the decision parameter: p = p + 2dy. b. If p >= 0, set the next point to (x+1, y+1) and update the decision parameter: p = p + 2dy - 2dx.

Repeat step 6 until the final point (x2, y2) is reached.

Q40. Give an equation for the plane containing the point (0,0,0) and normal to vector (-1,0, -1).

To find the equation of the plane, we can use the point-normal form of the equation of a plane, which is:

Ax + By + Cz = D

where A, B, and C are the components of the normal vector, and x, y, and z are the coordinates of any point on the plane. To find D, we can substitute the coordinates of the given point (0,0,0):

-1(0) + 0(0) - 1(0) = D

D = 0

Therefore, the equation of the plane containing the point (0,0,0) and normal to vector (-1,0,-1) is:

-1x + 0y - 1z = 0

or

-x - z = 0

Q41. Discuss why is the electron beam al lowed to overscan?

ANS The electron beam in a CRT is allowed to overscan for two main reasons:

To compensate for the edge effects: Due to the design of the CRT, the edges of the screen may be slightly darker or distorted than the rest of the display. Overscanning allows the beam to extend beyond the edges of the screen, compensating for this effect and ensuring that the image is displayed uniformly across the entire screen.

To prevent image burn-in: Overscanning also ensures that the image does not get burned into the screen due to prolonged exposure to a fixed image. By extending the beam beyond the edges of the screen, the CRT ensures that no part of the image remains stationary for too long, preventing burn-in.

Q42. Explain the 3D display methods.

ANS There are several 3D display methods that are used to create the illusion of depth and three-dimensional objects on a 2D screen. Some of the most common 3D display methods are:

Anaglyph: Anaglyph is a 3D display method that uses two images, one in red and the other in cyan. The viewer wears a pair of anaglyph glasses, which have red and cyan lenses. When viewed through these glasses, each eye sees a slightly different image, creating the illusion of depth.

Polarized: Polarized 3D displays use special glasses that have two lenses with different polarizing filters. The screen displays two images, one for each eye, with each image polarized in a different direction. The polarized lenses on the glasses ensure that each eye sees the correct image, creating the illusion of depth.

Active shutter: Active shutter 3D displays use glasses that alternate between blocking the left and right eye in sync with the images displayed on the screen. This creates the illusion of depth, with each eye seeing a slightly different image.

Autostereoscopic: Autostereoscopic displays do not require glasses to create the illusion of 3D. These displays use lenticular lenses or parallax barriers to create multiple viewing angles, allowing the viewer to see different images with each eye, creating the illusion of depth.

Holographic: Holographic displays use laser beams to create a 3D image that appears to float in mid-air. These displays are currently very expensive and are mainly used for research purposes.

Each of these 3D display methods has its advantages and disadvantages, and the choice of method depends on factors such as cost, image quality, and viewer comfort.

Q43. When 8 way symmetry is used to obtain a full circle from pixel coordinates generated for the 0 degree to 45 degree or the 90 degree to 45 degree octant, certain pixels are set or plotted twice. This phenomenon is sometimes referred to as overstrike. Justify where overstrike occurs.

ANS Overstrike occurs in 8-way symmetry circle drawing algorithms when pixels are plotted or set twice. This occurs when the circle is drawn using pixel coordinates generated for the 0 degree to 45 degree or the 90 degree to 45 degree octant, as the algorithm is symmetric and plots pixels in all 8 directions.

For example, in the 0 degree to 45 degree octant, the algorithm plots pixels in the positive x and y directions. When the algorithm moves to the 45 degree to 90 degree octant, it plots pixels in the positive x and negative y directions. However, the pixels in the positive x direction have already been plotted in the 0 degree to 45 degree octant, so they are plotted again in the 45 degree to 90 degree octant, resulting in overstrike.

Similarly, in the 90 degree to 45 degree octant, the algorithm plots pixels in the positive y and negative x directions. When the algorithm moves to the 135 degree to 90 degree octant, it plots

pixels in the negative x and negative y directions. However, the pixels in the negative x direction have already been plotted in the 90 degree to 45 degree octant, so they are plotted again in the 135 degree to 90 degree octant, resulting in overstrike.

Overstrike can lead to uneven or thicker lines in certain parts of the circle, and can also affect the performance and efficiency of the algorithm. To avoid overstrike, some circle drawing algorithms use 4-way symmetry instead of 8-way symmetry.

Q44. List steps required to generate an ellipse using polynomial method?

ANS

Define the center point of the ellipse (h,k).

Determine the lengths of the major and minor axes of the ellipse (a,b).

Generate a polynomial expression for the equation of the ellipse using the center point and lengths of the axes. This polynomial can be of the form: (x-h)^2/a^2 + (y-k)^2/b^2 = 1.

Choose a value of x, and solve for y using the polynomial equation.

Plot the generated point (x,y) on the ellipse.

Repeat steps 4 and 5 for all values of x within the range of the major axis.

Mirror the plotted points about the minor axis to complete the ellipse.

Q45. When 4 -way symmetry is used to obtain a full ellipse form pixel coordinates generated for the first quadrant, does overstrike occurs? Where?

ANS No, overstrike does not occur when 4-way symmetry is used to obtain a full ellipse from pixel coordinates generated for the first quadrant. In the 4-way symmetry approach, each pixel is plotted only once and there is no overlapping or duplication of pixels.

Q46. List the operating characteristics for the folLowing display technologies: raster refresh systems, vector refresh systems, plasma panels, and .CDs.

ANS Here are the operating characteristics for the following display technologies:

Raster refresh systems:

- Each point on the screen is represented by a single pixel
- Image is created by scanning the screen line by line
- Refresh rate determines the number of times the image is updated per second
- Can display both static and dynamic images

Vector refresh systems:

- Uses a vector generator to generate images by drawing lines and shapes
- Can display complex 2D and 3D images with high resolution
- Refresh rate is lower than raster refresh systems
- Typically used in applications such as scientific and medical imaging, CAD, and animation

Plasma panels:

- Uses plasma cells to produce images
- Each pixel is made up of a cell that contains a gas mixture and electrodes
- High contrast and wide viewing angles
- Limited lifespan due to the gradual deterioration of the plasma cells

CRTs:

- Uses a cathode ray tube to produce images
- High resolution and color accuracy
- Large and bulky compared to other display technologies
- Limited lifespan due to the wear and tear of the electron gun and phosphor coating

Q47. Determine the resolution (pixels per centimeter) in the x and y directions for the video monitor in use on your system. Determine the aspect ratio, and explain how relative proportions of objects can be maintained on your system.

ANS As an AI language model, I do not have access to the specific system you are using, so I cannot provide the exact resolution in pixels per centimeter. However, you can find the resolution of your monitor by checking its specifications or settings.

To determine the aspect ratio of your monitor, you can divide the horizontal resolution by the vertical resolution. For example, if your monitor has a resolution of 1920 pixels by 1080 pixels, the aspect ratio would be 16:9.

To maintain relative proportions of objects on your system, it is important to consider the aspect ratio when creating or viewing content. If an image or video has a different aspect ratio than your monitor, it may be stretched or distorted. To avoid this, you can adjust the display settings on your computer to maintain the correct aspect ratio, or crop the content to fit the screen.

Q48. Consider three different raster systems with resolutiuns of 640 by 400, 1280 by 1024 and 2560 by 2048. What size frame buffer (in bvtejl is needed for each of these systems to store 12 bits per pixel? How, much storage is required for each system if 24 bits per pixel are to be stored?

ANS To calculate the size of the frame buffer required to store 12 bits per pixel, we can use the following formula:

Size (in bytes) = (resolution in pixels) × (bits per pixel) ÷ 8

For a resolution of 640 by 400 and 12 bits per pixel:

Size = (640 × 400) × 12 ÷ 8 = 1,536,000 bytes

For a resolution of 1280 by 1024 and 12 bits per pixel:

Size = (1280 × 1024) × 12 ÷ 8 = 3,932,160 bytes

For a resolution of 2560 by 2048 and 12 bits per pixel:

Size = (2560 × 2048) × 12 ÷ 8 = 19,660,800 bytes

To calculate the size of the frame buffer required to store 24 bits per pixel, we can use the same formula as above:

Size (in bytes) = (resolution in pixels) × (bits per pixel) ÷ 8

For a resolution of 640 by 400 and 24 bits per pixel:

Size = (640 × 400) × 24 ÷ 8 = 3,072,000 bytes

For a resolution of 1280 by 1024 and 24 bits per pixel:

Size = (1280 × 1024) × 24 ÷ 8 = 7,864,320 bytes

For a resolution of 2560 by 2048 and 24 bits per pixel:

Size = (2560 × 2048) × 24 ÷ 8 = 39,321,600 bytes

Therefore, the required size of the frame buffer depends on the resolution and the number of bits per pixel, with higher resolutions and more bits per pixel requiring larger frame buffers.

Q49. Suppose an RGB raster system is to be designed using an 8 -inch by 10 - inch screen with a resolution of 100 pixels per inch in each direction. If we want to store h bit5 per pixel in the frame buffer, how much storage (in bytes) do we need for the frame buffer?

ANS The total number of pixels in the screen is:

8 inches x 100 pixels/inch (horizontal) = 800 pixels 10 inches x 100 pixels/inch (vertical) = 1000 pixels Total pixels = 800 x 1000 = 800,000 pixels

Since we are using RGB system, we need 3 bytes (24 bits) to store the color information for each pixel.

Therefore, the total number of bytes required to store the frame buffer would be:

800,000 pixels x 3 bytes/pixel = 2,400,000 bytes

If we want to store h bits per pixel, we would need to adjust the calculation accordingly. For example, if h = 4, we would need 2 bytes (16 bits) per pixel, and the total storage required would be:

800,000 pixels x 2 bytes/pixel = 1,600,000 bytes

Answer: The storage required for the frame buffer would be 2,400,000 bytes if we use 24 bits per pixel.

Q50. Define refreshing of the screen?

ANS Refreshing of the screen refers to the process of redrawing or updating the content on a computer screen. This can be done manually by the user or automatically by the computer system. The purpose of refreshing the screen is to ensure that the displayed information is up to date and accurate. This is especially important in dynamic applications where the content on the screen changes frequently, such as in video games or real-time monitoring systems.

Q51. List out the merits and demerits of DVST.

ANS Refreshing of the screen refers to the process of redrawing or updating the content on a computer screen. This can be done manually by the user or automatically by the computer system. The purpose of refreshing the screen is to ensure that the displayed information is up to date and accurate. This is especially important in dynamic applications where the content on the screen changes frequently, such as in video games or real-time monitoring systems.

Q52. Describe in detail about the DDA scan conversion algorithm?

ANS The Digital Differential Analyzer (DDA) algorithm is a basic scan conversion algorithm used to convert a line segment defined by two endpoints into a series of pixels or points for display on a computer screen. Here's a description of how the DDA algorithm works:

Calculate the slope of the line segment:

- m = (y2-y1)/(x2-x1)

Determine the number of steps or pixels required to draw the line:

- Let dx be the difference in the x-coordinates of the endpoints, dx = x2-x1
- Let dy be the difference in the y-coordinates of the endpoints, dy = y2-y1

- Let steps be the maximum of dx and dy: steps = max(dx, dy)

Calculate the increment in x and y for each step:

- Let xi and yi be the x and y coordinates of the starting point (x1, y1)
- Let dx_step = dx/steps and dy_step = dy/steps be the increments in x and y respectively for each step

Use a loop to draw the line by plotting each pixel or point along the line:

- For i = 1 to steps do:
  - Plot the pixel at (round(xi), round(yi))
  - Update the values of xi and yi by adding dx_step and dy_step respectively: xi = xi + dx_step, yi = yi + dy_step

The DDA algorithm is simple and efficient, but it may introduce round-off errors and produce jagged lines at certain angles. Modern graphics systems often use more advanced algorithms such as Bresenham's line algorithm or anti-aliasing techniques to improve the quality of line drawing.

Q52. Describe in detail about the DDA scan conversion algorithm?

ANS The Digital Differential Analyzer (DDA) algorithm is a basic scan conversion algorithm used to convert a line segment defined by two endpoints into a series of pixels or points for display on a computer screen. Here's a description of how the DDA algorithm works:

Calculate the slope of the line segment:

- m = (y2-y1)/(x2-x1)

Determine the number of steps or pixels required to draw the line:

- Let dx be the difference in the x-coordinates of the endpoints, dx = x2-x1
- Let dy be the difference in the y-coordinates of the endpoints, dy = y2-y1
- Let steps be the maximum of dx and dy: steps = max(dx, dy)

Calculate the increment in x and y for each step:

- Let xi and yi be the x and y coordinates of the starting point (x1, y1)
- Let dx_step = dx/steps and dy_step = dy/steps be the increments in x and y respectively for each step

Use a loop to draw the line by plotting each pixel or point along the line:

- For i = 1 to steps do:
  - Plot the pixel at (round(xi), round(yi))

- Update the values of xi and yi by adding dx_step and dy_step respectively: xi = xi + dx_step, yi = yi + dy_step

The DDA algorithm is simple and efficient, but it may introduce round-off errors and produce jagged lines at certain angles. Modern graphics systems often use more advanced algorithms such as Bresenham's line algorithm or anti-aliasing techniques to improve the quality of line drawing.

Q53. Write down and explain the midpoint circle drawing algorithm. Assume 10 cm as the radius and co -ordinate origin as the centre of the circle.

ANS The Midpoint Circle Drawing Algorithm is a commonly used algorithm for drawing circles on a computer screen. It works by iteratively plotting the points along the circumference of the circle using symmetry properties. Here are the steps involved in drawing a circle with a radius of 10cm and a center at the coordinate origin:

Set the initial coordinates:

- Let x0 = 0 and y0 = 10 be the initial coordinates of the top point of the circle.
- Let P = 1 - r be the initial decision parameter, where r is the radius of the circle.

Use a loop to generate the coordinates of the other points along the circumference of the circle:

- For each iteration of the loop, calculate the coordinates of the next point by using the symmetry properties of the circle.
- Let x and y be the coordinates of the next point.
- Calculate the decision parameter for the next point: P = P + 2*x + 3.
- If P > 0, update the coordinates and the decision parameter as follows:
  - x = x - 1
  - P = P - 2*y* + *2*x + 5
- Else, update the coordinates and the decision parameter as follows:
  - y = y - 1
  - P = P + 2*x* + *2*y + 1

Plot the points on the screen:

- For each iteration of the loop, plot the points (x, y), (-x, y), (x, -y), and (-x, -y) using symmetry properties.

The Midpoint Circle Drawing Algorithm is a simple and efficient algorithm for drawing circles, and it can be easily adapted for drawing circles with different radii and center points. However, the algorithm may produce irregularities in the circle due to the use of integer arithmetic, and these irregularities can be reduced by using anti-aliasing techniques.

Q54.. Explain in detail about Bresenham‟s ellipse generating algorithm. Give example.

ANS Bresenham's ellipse generating algorithm is a popular algorithm used for drawing ellipses on a computer screen. It is based on the midpoint ellipse algorithm, which is a generalization of the midpoint circle algorithm used to draw circles.

The algorithm uses the properties of an ellipse to calculate the next point on the ellipse, based on the current point and a decision parameter. The decision parameter is used to determine which direction to move along the x and y axes, and how much to move.

The steps involved in Bresenham's ellipse generating algorithm are as follows:

Define the coordinates of the center of the ellipse (xc, yc), the major axis (a) and the minor axis (b).

Initialize the decision parameter as follows:

dx = 2 * b^2 * xc dy = 2 * a^2 * yc d1 = b^2 - a^2 * b + a^2 / 4

Use a loop to generate the coordinates of the points along the ellipse:

- Calculate the x and y coordinates of the next point on the ellipse based on the current point and the decision parameter.
- Calculate the next value of the decision parameter.
- Use symmetry to plot the other points on the ellipse.
- Repeat until the entire ellipse has been drawn.

- The equations for calculating the x and y coordinates are:
- x = x + 1 if d1 < 0: dx = dx + 2 * b^2 d1 = d1 + dx + b^2 else: y = y - 1 dx = dx + 2 * b^2 dy = dy - 2 * a^2 d1 = d1 + dx - dy + b^2
- The symmetry properties used in this algorithm are that for any point (x, y) on the ellipse, the points (-x, y), (x, -y), and (-x, -y) are also on the ellipse.

An example of Bresenham's ellipse generating algorithm can be illustrated as follows:

Consider an ellipse with center coordinates (5,5), a major axis of 10 and a minor axis of 6. The algorithm starts by setting the initial coordinates to (0, 6) and the decision parameter to:

dx = 2 * 6^2 * 0 = 0 dy = 2 * 10^2 * 6 = 1200 d1 = 6^2 - 10^2 * 6 + 10^2 / 4 = -335

Using the algorithm to calculate the next point on the ellipse, we get:

x = 1 y = 6 dx = 144 dy = 1200 d1 = -279

Plotting this point and using symmetry properties, we get the other points on the ellipse. We repeat this process until the entire ellipse has been drawn.

Bresenham's ellipse generating algorithm is a simple and efficient algorithm for drawing ellipses, and it can be easily adapted for drawing ellipses with different major and minor axes and center points. However, as with the midpoint circle drawing algorithm, the use of integer arithmetic may produce irregularities in the ellipse.

Q55. Differentiate raster and random scan systems.

ANS Raster scan and random scan are two types of display systems used in computer graphics.

In a raster scan system, the display is created by scanning the electron beam across the screen in a pattern of horizontal lines from left to right, starting at the top left corner and moving downwards. This is done repeatedly to create a series of images on the screen. Each line is divided into a fixed number of pixels, and each pixel is assigned a color value to create an image. Raster scan systems are commonly used for displays such as computer monitors and televisions.

In a random scan system, the electron beam is not scanned across the screen in a fixed pattern of horizontal lines. Instead, the beam is directed to specific areas of the screen where images are to be displayed. This allows for greater flexibility in creating images with irregular shapes and sizes. Random scan systems are commonly used for specialized displays such as medical imaging and scientific visualization.

The main difference between raster and random scan systems is in the way the electron beam is scanned across the screen. In a raster scan system, the beam is scanned in a fixed pattern of horizontal lines, while in a random scan system, the beam is directed to specific areas of the screen. Raster scan systems are more commonly used due to their simplicity and lower cost, while random scan systems are more specialized and used for applications that require greater flexibility in image creation.

UNIT 2

Q1. Define Transformation.

ANS In computer graphics, transformation refers to the process of changing the position, orientation, or size of an object in a 2D or 3D space. This can include translation (moving an object), rotation (changing its orientation), scaling (changing its size), and skewing (distorting its shape). Transformations can be applied to individual objects or to an entire scene, and are

typically used to create animations, simulate motion, and manipulate images. Transformations are often achieved through the use of mathematical algorithms and matrices.

Q2. List out the various Text clipping.

ANS In computer graphics, text clipping refers to the process of determining which parts of a text string should be displayed on the screen or in a particular region. Here are some of the various types of text clipping techniques:

Rectangle clipping

Polygon clipping

Circle and ellipse clipping

Bezier clipping

Alpha clipping

Pixel clipping

Anti-aliasing clipping

The choice of text clipping technique depends on the specific requirements of the application and the desired visual effect.

Q3. Explain about window to viewport coordinate transformation.

ANS. In computer graphics, the process of transforming an object from its original coordinate system to the screen is called viewport transformation. This involves mapping the object's original coordinates to the coordinates of the display window. The transformation can be performed using a series of mathematical operations, including scaling, translation, and rotation.

The window to viewport coordinate transformation is a specific type of viewport transformation that maps the coordinates of the display window to the coordinates of the viewport. The window represents the area of the image that is being viewed, while the viewport represents the actual display area on the screen. The transformation is necessary because the size and shape of the window and viewport may not match, and the coordinates of the object may need to be adjusted accordingly.

To perform the window to viewport coordinate transformation, we need to define a scaling factor that maps the size of the window to the size of the viewport. This scaling factor is usually determined by dividing the width and height of the viewport by the width and height of the

window. Once the scaling factor is determined, we can use it to transform the object's original coordinates to the viewport coordinates by applying the following formula:

Xv = (Xw - Xwmin) * (Vxmax - Vxmin) / (Xwmax - Xwmin) + Vxmin

Yv = (Yw - Ywmin) * (Vymax - Vymin) / (Ywmax - Ywmin) + Vymin

where Xv and Yv are the coordinates of the object in the viewport, Xw and Yw are the coordinates of the object in the window, Xwmin and Ywmin are the minimum coordinates of the window, Xwmax and Ywmax are the maximum coordinates of the window, Vxmin and Vymin are the minimum coordinates of the viewport, and Vxmax and Vymax are the maximum coordinates of the viewport.

By applying this transformation, we can map the object's original coordinates to the viewport coordinates, ensuring that it is properly displayed on the screen.

Q4. Write a detailed note on the basic two dimensional transformations.

ANS. In computer graphics, two-dimensional (2D) transformations are used to manipulate the position, size, and orientation of 2D objects on the screen. There are several basic 2D transformations, including translation, scaling, rotation, and reflection. Here is a detailed explanation of each transformation:

Translation: Translation is a transformation that moves an object from one position to another. It involves adding or subtracting a constant value from the x and y coordinates of the object. The mathematical formula for translation is:

X' = X + tx Y' = Y + ty

where X and Y are the original coordinates of the object, X' and Y' are the transformed coordinates, and tx and ty are the translation factors in the x and y directions, respectively.

Scaling: Scaling is a transformation that changes the size of an object. It involves multiplying the x and y coordinates of the object by a scaling factor. The mathematical formula for scaling is:

X' = X * sx Y' = Y * sy

where X and Y are the original coordinates of the object, X' and Y' are the transformed coordinates, and sx and sy are the scaling factors in the x and y directions, respectively.

Rotation: Rotation is a transformation that rotates an object by a specified angle around a fixed point. It involves applying a rotation matrix to the object's coordinates. The mathematical formula for rotation is:

X' = X * cos(theta) - Y * sin(theta) Y' = X * sin(theta) + Y * cos(theta)

where X and Y are the original coordinates of the object, X' and Y' are the transformed coordinates, and theta is the rotation angle in radians.

Reflection: Reflection is a transformation that mirrors an object across a line of symmetry. It involves multiplying the x or y coordinate of the object by -1, depending on the direction of the reflection. The mathematical formula for reflection is:

X' = -X Y' = -Y

where X and Y are the original coordinates of the object, and X' and Y' are the transformed coordinates.

In summary, 2D transformations are used to manipulate the position, size, and orientation of objects on the screen. The four basic 2D transformations are translation, scaling, rotation, and reflection. These transformations can be combined to create more complex transformations, such as shearing and stretching.

Q5. Explain with an example the Cohen-Sutherland line clipping algorithm.

ANS. The Cohen-Sutherland line clipping algorithm is a popular line-clipping algorithm used in computer graphics. It divides the plane into nine regions, where each region is assigned a four-bit binary code based on the location of the endpoints of the line to be clipped. The algorithm uses these codes to determine which part of the line is inside or outside the clipping region and clips the line accordingly.

Here is a step-by-step explanation of the Cohen-Sutherland line clipping algorithm:

Determine the endpoints of the line to be clipped and assign binary codes to each endpoint based on its position with respect to the clipping region. For example, if the clipping region is a square with the lower left corner at (-1,-1) and upper right corner at (1,1), a point with coordinates (0.5, 0.8) will have the binary code 1001, where the first bit (1) indicates that the point is above the clipping region, the second bit (0) indicates that the point is to the left of the clipping region, the third bit (0) indicates that the point is below the clipping region, and the fourth bit (1) indicates that the point is to the right of the clipping region.

Determine the logical intersection of the binary codes for the two endpoints of the line. If the result is not 0000, the line is outside the clipping region, and it can be discarded.

If the line is partially inside the clipping region, determine which endpoints are outside the clipping region and compute the intersection of the line with the corresponding edge of the clipping region.

Replace the outside endpoint with the intersection point and repeat steps 2 and 3 until the line is entirely inside the clipping region.

Draw the clipped line.

For example, suppose we want to clip a line with endpoints A(-2,1) and B(0.5,3) to the square clipping region with corners (-1,-1) and (1,1). We assign binary codes to the endpoints as follows:

A: 1001 (above, left, below, right) B: 0101 (above, right, below, right)

The logical intersection of the binary codes is 0001 (to the right of the clipping region), indicating that the line is partially outside the clipping region. We determine that endpoint A is outside the clipping region and compute the intersection of the line with the left edge of the clipping region at x=-1:

x = -1, y = 2.5 Intersection point: (-1, 2.5)

We replace endpoint A with the intersection point and repeat the process:

A: 0001 (to the right of the clipping region) B: 0101 (above, right, below, right)

The logical intersection of the binary codes is 0101 (above and to the right of the clipping region), indicating that the line is still partially outside the clipping region. We determine that endpoint B is outside the clipping region and compute the intersection of the line with the top edge of the clipping region at y=1:

x = -0.2, y = 1 Intersection point: (-0.2, 1)

We replace endpoint B with the intersection point and repeat the process:

A: 0001 (to the right of the clipping region) B: 0001 (to the right of the clipping region)

The logical intersection of the binary codes is 0001 (to the right of the clipping region), indicating that the line is still partially outside the clipping region. We determine that endpoint A is outside the clipping region and compute the intersection of the line with the left edge of the clipping region at x=-1:

x = -1, y = 2

Q6. Compare Cohen-Sutherland line clipping algorithm and LiangBarsky line clipping algorithm.

ANS Cohen-Sutherland and Liang-Barsky are two widely used algorithms for line clipping in computer graphics. Both algorithms clip a line segment against a rectangular clipping window, but they use different approaches to achieve this task. Here are some key differences between the two algorithms:

Encoding scheme: The Cohen-Sutherland algorithm uses a four-bit binary code to encode the relative position of a point with respect to the clipping window. On the other hand, the Liang-Barsky algorithm computes the intersection points of the line segment with the four edges of the

clipping window and uses these intersections to determine whether the line segment lies entirely inside, outside, or partially inside the clipping window.

Calculation of intersection points: In the Cohen-Sutherland algorithm, intersection points are computed by solving a system of linear equations. In contrast, the Liang-Barsky algorithm uses parametric equations to calculate intersection points.

Number of calculations: The Cohen-Sutherland algorithm involves a fixed number of calculations per endpoint of the line segment, regardless of the position of the endpoint with respect to the clipping window. In contrast, the Liang-Barsky algorithm involves different numbers of calculations depending on the position of the endpoint.

Performance: The Liang-Barsky algorithm is generally faster than the Cohen-Sutherland algorithm because it involves fewer calculations and does not require a lookup table. However, the performance advantage of the Liang-Barsky algorithm becomes less significant for smaller clipping windows.

In summary, both the Cohen-Sutherland and Liang-Barsky algorithms are effective for line clipping, but the Liang-Barsky algorithm has some advantages in terms of speed and accuracy.

Q7. Write the general form of the matrix for rotation about a point P(h, k).

ANS. The general form of the matrix for rotation about a point P(h, k) is:

c

```
[cosθ  -sinθ  h(1-cosθ)+k sinθ]
[sinθ  cosθ   k(1-cosθ)-h sinθ]
[0      0              1      ]
```

Here, θ is the angle of rotation in radians. This matrix performs a rotation about the point (h, k) in the counterclockwise direction when multiplied by the column vector of homogeneous coordinates [x y 1]. The resulting vector represents the coordinates of the point after rotation.

Note that the third row of the matrix is [0 0 1], which indicates that it is an affine transformation and not a perspective transformation.

Q8. Perform a 45 degree rotation of triangle A(0, 0), B(1, 1), C(5, 2) about P(-1, -1)

ANS. To perform a 45-degree rotation of triangle A(0, 0), B(1, 1), C(5, 2) about P(-1, -1), we can use the following steps:

Translate the triangle and the point of rotation to the origin by subtracting (-1, -1) from their coordinates.

scssCopy code

```
A' = (0 - (-1), 0 - (-1)) = (1, 1)
B' = (1 - (-1), 1 - (-1)) = (2, 2)
C' = (5 - (-1), 2 - (-1)) = (6, 3)
P' = (-1 - (-1), -1 - (-1)) = (0, 0)
```

Apply a 45-degree counterclockwise rotation about the origin using the rotation matrix:

cCopy code

```
[cosθ  -sinθ]
[sinθ  cosθ ]
```

where θ = 45 degrees or π/4 radians. The rotation matrix is:

cCopy code

```
[cos(π/4)  -sin(π/4)]
[sin(π/4)  cos(π/4) ]
=
[√2/2     -√2/2    ]
[√2/2      √2/2    ]
```

Multiply each vertex of the triangle by the rotation matrix to obtain the new coordinates after rotation.

cssCopy code

```
A'' = [√2/2     -√2/2][1]
                      [1]
    = [0.707  -0.707][1, 1]
    = [0       0    ]

B'' = [√2/2     -√2/2][2]
                      [2]
    = [0.707  -0.707][2, 2]
    = [0.707  -1.414]

C'' = [√2/2     -√2/2][6]
                      [3]
```

```
= [0.707  -0.707][6, 3]
= [4.243  -1.414]
```

Translate the triangle and the point of rotation back to their original position by adding (-1, -1) to their coordinates.

pythonCopy code

```
A''' = (0 + (-1), 0 + (-1)) = (-1, -1)
B''' = (0.707 + (-1), -1.414 + (-1)) = (-0.293, -2.414)
C''' = (4.243 + (-1), -1.414 + (-1)) = (3.243, -2.414)
P''' = (0 + (-1), 0 + (-1)) = (-1, -1)
```

Therefore, the vertices of the rotated triangle are A'''(-1, -1), B'''(-0.293, -2.414), and C'''(3.243, -2.414).

Q9. 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.

ANS. To 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, we can follow these steps:

Translate the triangle so that the fixed point C is at the origin (0,0) by subtracting the coordinates of C from each vertex:

- A' = A - C = (0, 0) - (5, 2) = (-5, -2)
- B' = B - C = (1, 1) - (5, 2) = (-4, -1)
- C' = C - C = (5, 2) - (5, 2) = (0, 0)

Multiply the coordinates of A' and B' by 2 to get the magnified triangle:

- A'' = 2A' = 2(-5, -2) = (-10, -4)
- B'' = 2B' = 2(-4, -1) = (-8, -2)

Translate the magnified triangle back to its original position by adding the coordinates of C:

- A''' = A'' + C = (-10, -4) + (5, 2) = (-5, -2)
- B''' = B'' + C = (-8, -2) + (5, 2) = (-3, 0)

So the magnified triangle has vertices A'''(-5,-2), B'''(-3,0) and C(5,2).

Q10. Reflect the diamond shaped polygon whose vertices are A(-1, 0), B(0, -2), C(1, 0) and D(0, 2) about the line y=x+2.

ANS To reflect the diamond shaped polygon A(-1, 0), B(0, -2), C(1, 0), and D(0, 2) about the line y=x+2, we can follow these steps:

Find the equation of the line of reflection. Since the line of reflection is y=x+2, its slope is 1 and its y-intercept is 2. We can also write the equation of the line as x-y+2=0.

Find the perpendicular bisector of each segment connecting the vertices of the polygon to the line of reflection. The perpendicular bisectors will intersect the line of reflection at the reflected points.

Find the equations of the perpendicular bisectors. For example, to find the perpendicular bisector of the segment AB:

- The midpoint of AB is ((-1+0)/2, (0-2)/2) = (-0.5, -1).
- The slope of AB is (-2-0)/(0--1) = 2.
- The slope of the perpendicular bisector is the negative reciprocal of the slope of AB, which is -1/2.
- Using the point-slope form, we get the equation of the perpendicular bisector as y - (-1) = (-1/2)(x - (-0.5)), which simplifies to y = -x - 0.5.

Find the point of intersection of the perpendicular bisector with the line of reflection. To find the point of intersection of the perpendicular bisector of AB with the line y=x+2, we solve the system of equations:

- y = -x - 0.5
- x - y + 2 = 0

- Substituting y = -x - 0.5 in the second equation gives x - (-x - 0.5) + 2 = 0, which simplifies to x = 1.75. Substituting x = 1.75 in the first equation gives y = -1.25.

Find the reflected points by using the distance formula. For example, the reflected point of A will be the same distance from the line of reflection as A is, but on the other side of the line. We can use the distance formula to find the distance between A and the line of reflection:

- distance = |ax + by + c| / sqrt(a^2 + b^2)
- distance = |(-1) - 1(0) + 2| / sqrt(1^2 + 1^2) = sqrt(2)

- The reflected point of A will also be sqrt(2) units away from the line of reflection, but on the other side of the line. To find the reflected point, we can use the point-slope form of the equation of the line:

  - y - y1 = m(x - x1)
  - y - 0 = (-1)(x - (-1))
  - y + 1 = -x - 1
  - y = -x - 2

- The line passing through A and its reflected point will have a slope of -1, so we can find the equation of this line:
    - $y - 0 = (-1)(x - (-1))$
    - $y = -x + 1$
- Now we can find the intersection of this line with the line of reflection:
    - $y = x + 2$
    - $-x + 1 = x + 2$
    - $x = -0.5$
    - $y = 1.5$
- Therefore, the reflected point of A is (-