PYQ

2021

① (a) Define refresh rate of a CRT monitor.

⇒ The refresh rate of a CRT (cathode Ray Tube) monitor is a number of times the electron gun redraws the entire image on the screen per second, typically measured in Hertz (Hz).

(b) What is the advantage of using LCD monitor then to use CRT monitor.

⇒ some Advantages of LCD over CRT is given below:

1. Space-saving and lightweight.
2. Lower power consumption.
3. No radiation emissions.
4. Higher resolution and clarity.
5. Wider view angles.
6. Less heat generation.
7. No flicker or glare.

(c) What is Pixel?

⇒ A Pixel (short for picture element) is the smallest unit of digital information that makes up an image on a digital display device, such as a computer moniter or television.

It's essentially a tiny square of color that, when combined with other pixels, forms a Digital Image.

(d) What is the role of electron gun in a CRT monitor?

→ The electron gun generates a stream of electrons that are focused ~~phosphor~~ onto the phosphor coating on the inside of the CRT screen, creating the images we see.

Specially, the electron gun:

1. Produces a beam of electrons.
2. Accelerates and focuses the beam
3. Directs the beam horizontally and vertically to create the image

(e) What is 3D transformation?

→ 3D transformation refers to the process of changing the position, orientation or scale of a 3D object in a three-Dimensional space.

(f) What is reflection in 2D graphics?

→ In 2D graphics, reflection refers to the process of creating a mirror image of an object or shape around a specific axis, usually the x-axis or y-axis.

(g) Define composite transformation.

→ Composite transformation refers to the process of applying multiple transformations (such as translation, rotation, scaling, and reflecting) to an object in a specific order, resulting in a combined transformation

(4) What is fixed point scaling?

⇒ Fixed-Point scaling is a type of scaling where an object is scaled with respect to a fixed point, rather than the origin.

Math 2021

(1) (a) Define computer graphics.

⇒ Computer Graphics is the field of study and applications of creating and manipulating visual content using computer technology.

It involves the use of computers to generate, manipulate, and display visual information, including:

1. Images
2. Animation
3. 3D models
4. Special effects.

(b) What are input and output devices in CG?

⇒ Input Devices: These devices allow users to interact with the computer and provide data for graphic content.

1. Keyboard
2. Mouse
3. Graphic Tablet
4. Scanner
5. Digital camera
6. Track ball
7. Joystick

Output Devices: These devices display the graphical output.

1. Monitor (CRT, LCD, LED)
2. Printer (Inkjet, Laser)
3. Plotter
4. Projector
5. Virtual Reality - (VR) headset
6. Augmented Reality (AR) display.

(c) What is transformation?

→ In computer graphics transformation refers to the process of changing the position, orientation, or size of graphical object, such as a point, line, or shape.

(d) What is scaling?

→ In computer graphics, scaling refers to the process of resizing a graphical object, making it larger or smaller, while maintaining its original shape and ~~properties~~ proportions.

(e) What is persistence?

→ In computer graphics, persistence refers to the ability of a display device to retain an image for a short period of time, even after the original signal has been removed.

(f) Why Bresenham's algorithm is better than DDA?

→ Bresenham's algorithm is ~~best~~ better than DDA because:

1. it uses only Integer arithmetic which is faster and more accurate than floating-point Arithmetic used in DDA.
2. it ~~doesn't~~ avoid division operations, which can be slow and prone to errors.
3. It is faster and efficient because it uses simple and more efficient method for calculating pixel co-ordinates.

(g) Define clipping

⇒ In computer graphics, clipping refers to the process of removing or truncating parts of a graphical object that fall outside a defined boundary or region, known as the clipping region or viewport.

(h) What do you mean by Resolution?

⇒ In computer graphics, resolution refers to the number of pixels (tiny dots) used to display an image on a screen or other display device.

2022

(1) (a) State an objective of studying computer graphics.

⇒ To enable the creation and manipulation of visual content for effective communication education and entertainment.

(b) Name some Hardware devices commonly used to support computer graphics.

⇒ Graphics Processing Unit (GPU), Graphics Card, Monitor, Graphics Tablet, 3D mouse, Game controller, Virtual Reality (VR) ~~devi~~ headset, 3D scanner, Plotter, Graphics Accelerator.

(c) Name a polygon filling algorithm.

= Scan-Line Algorithm.

(d) Name a hidden surface removal algorithm.

⇒ Z-Buffer Algorithm.

(e) What is projection?

⇒ In computer Graphics, projection refers to the process of transforming a 3D object or scene onto a 2D surface, such as a screen or image plane.

(f) What do you mean by vanishing point?

⇒ In perspective drawing and computer Graphics, a vanishing point is a point in the distance where parallel lines appear to converge and disappear.

(g) Name some Issues in geometric modelling.

⇒ 1) Errors in the connectivity and structure of geometric models.

2) Inaccuracies in the shape and size of geometric models.

3) Instability or errors in geometric computations due to numerical precision issues.

(h) Name a software which is commonly used to support animation.

⇒ Blender, which is open-source and free, there are others like Maya (Autodesk)
Animate (Adobe)
3ds Max
Cinema 4D

2022-Math

(1) (a) Define Computer Graphics?

(b) What are Input and output devices in computer Graphics?

(c) Give matrix representation of 2D rotation

→ The matrix representation of a 2D rotation by an angle θ is:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

(d) What is aspect ratio in display?

→ The aspect ratio in display refers to the proportional relationship between it's width and height.

It is typically expressed as a ratio of width to height, such as: [4:3] or [16:9] etc.

(e) What are the Advantages of using LCD monitor than to use CRT monitors?

(f) Define Pixel and resolution.

→ Pixel: A pixel (short for "picture element") is the smallest unit of digital information that makes up an image on a digital display. It is tiny, square-shaped element that represents a single color value.

Resolution: Resolution refers to the total number of pixels that make up an image or display. It is usually measured in terms of the number of pixels along the horizontal and vertical axies and is expressed in the format:

width x Height (e.g: 1920 x 1080).

(f) Define clipping?

(h) What is the difference between random and raster scan?

⇒ Random scan: Also known as vector scan, this method draws images by tracking lines and curves directly to the display. The electron beam or drawing mechanism moves randomly to specific points on the screen to draw the image.

Raster scan: This method draws images by scanning the electron beam horizontally and vertically across the screen, one row at a time. The beam illuminates pixels in a fixed pattern, creating the image.

2022-23

(1)(a) What is the role of electron gun in a CRT monitor?

(b) What is 3D transformation?

(c) What is projection?

(d) What is color lookup table?

⇒ A color look-up table (CLUT) is:

A precomputed table that maps into color values to output color values.

In other words, a CLUT is a table that contains a set of predefined color values that can be used to quickly lookup and replace color values in an image or video.

(e) What do you mean by projection reference point?

⇒ The (PRP) is a point from which the projection is performed. It's point of view from which the 3D scene is projected onto a 2D surface, such as a screen or image plane.

(f) Define Aspect ratio.

(g) what is the difference between window and Viewport?

(A) ⇒ Window: The window defines the 3D region to be projected.

Viewport: The viewport defines the 2D screen area where the projection is displayed.

(h) What is persistence?

2023-24

(1)(a) what is vertical retrace?

⇒ Vertical retrace refers to the process of the ~~electr~~ electron beam or display controller returning to the top of the screen to start a new frame.

(b) Name some hardware devices commonly used to support computer graphics.

(c) Why do we need homogenous co-ordinates?

→ 1) Translation can be represented as matrix ~~manip~~ multiplication.

2) Perspective projection can be represented as a matrix multiplication.

(d) What is composite transformation?

(e) Name a process used for eleminating part of a ~~scren~~ scene outside a specified window.

⇒ clipping

(f) Write two examples of scan converted objects.

⇒ Lines: Scan conversion is used to render lines on a raster display.

Circles: Scan conversion is used to render circles and other curves on a raster display.

Images: Scan conversion is used to render bitmap images on a raster display.

(g) What is projection?

(h) What do you mean by Vanishing Point?

5x2

2021 CG- ~~CH2~~ ~~CH3~~

(a) What is a 4-connected region? Write down the boundary fill algorithm. (5 marks).

⇒ A 4-connected region refers to a set of pixels in a digital image. where each pixels in connected to it's four immediate neighbors: up, down, left, right. This connectivity is used in region filling algorithms to determine how pixels are grouped or filled. unlike a 8-connected region (which includes diagonal neighbors), a 4-connected region restricts

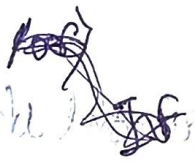
movement to cardinal directions only

Boundary Fill Algorithm

⇒ This algorithm fills a region with a specified color until it encounters a boundary color.
It uses a recursive approach, here's the pseudocode for a 4-connected boundary fill:

Our Boundary fill algorithm takes 4 input/Argument (x, y), fill_color, boundary_color

1. If (x,y) is outside the image or already filled with fill_color or is boundary color: return



2. Set pixel at (x, y) to fill_color
3. Recursively call Boundaryfill for all 4 neighbors:
   - Boundaryfill (x+1, y, fill_color, boundary_color)
     // call the boundary fill with x-incremented.
   - Boundary fill (x-1, y, fill_color, boundary_color)
   - Boundary fill (x, y+1, fill_color, boundary_color)
   - Boundary fill (x, y+1, fill_color, boundary_color);

Explanation

→ Starts at seed point (x,y) inside the region.

→ Checks if the current pixels is within bounds and not a boundary or already filled.

→ Fills the pixel and recursively fills its 4-connected neighbors.
This is efficient for simple shapes but may cause stack overflow for large regions.

(b) write down the Bresenham's line drawing algorithm. Using this algorithm, find out points of a line with end points (1,4) and (4,5).

→ Algorithm

This algorithm efficiently plots a straight line between two points (by determining which pixels to illuminate based on minimize error.
It works for lines with slopes between 0 and 1 (adjustable for other steps).

two line) (x_0, y_0)
(x_1, y_1)

Pseudocode

1. $\Delta x = x1 - x0$,
   $\Delta y = y1 - y_0$
2. $p = 2 * \Delta y - \Delta x$ //initial decision parameter.
3. plot (x_0, y_0)

4. While $(x_0 < x_1)$:

- $x_0 = x_0 + 1$
- If $P < 0$:

$$P = P + 2 * \Delta Y$$

ELSE:

$$Y_0 = Y_0 + 1$$

$$P = P + 2 * dy - 2 * dx$$

Plot (x_0, y_0)

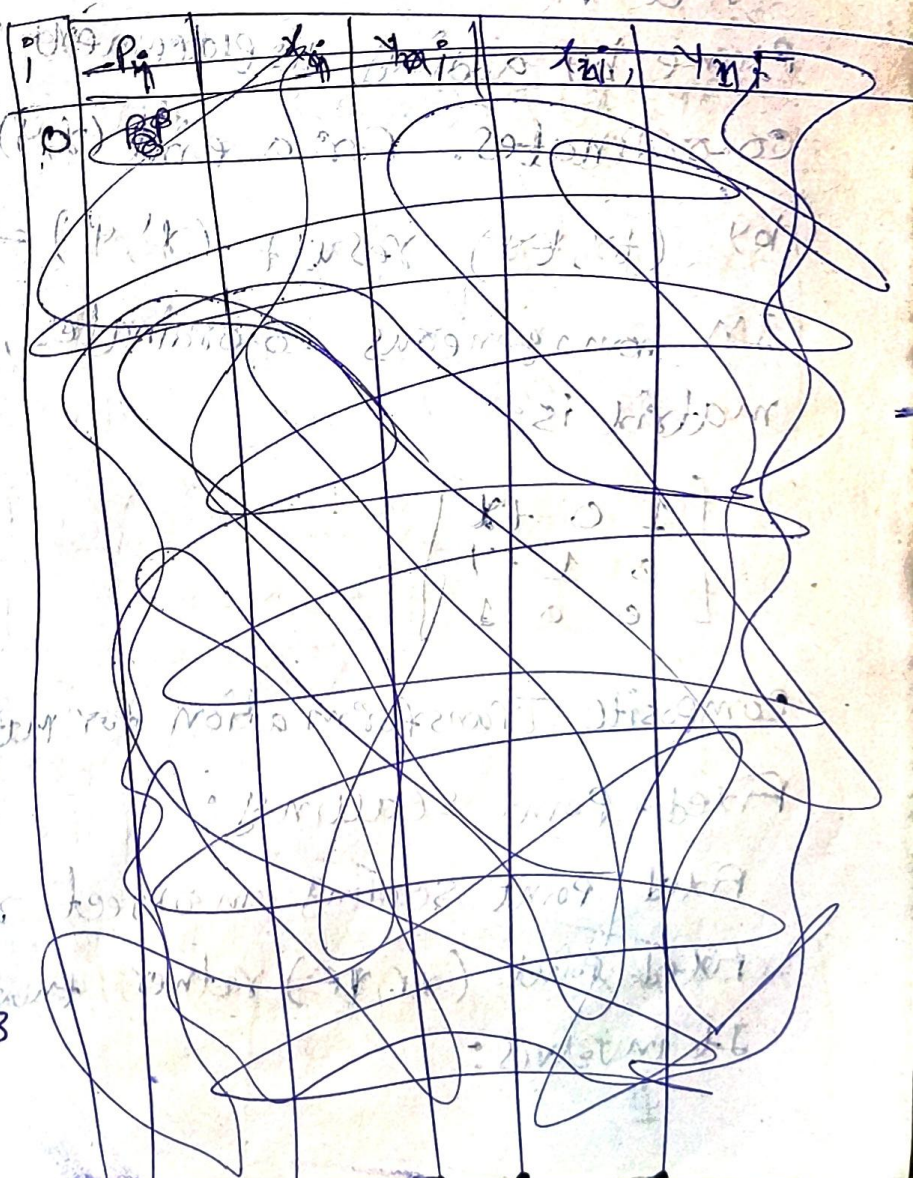
Calculation for (1,4) to (4,5)

- $x_0 = 1, y_0 = 4, x_1 = 4, y_1 = 5$

$\Delta x = 4 - 1 = 3$

$\Delta y = 5 - 4 = 1$

$P_0 = 2 \times 1 - 3 = -1$



Steps

a) Start at (1,4),

$P = -1$

- $P < 0$: $P = 1$

$x_0 = 2$

$y_0 = 4$

b) At (2,4) $P = 1$

- $P \geq 0$: $y_0 = 5$

$P = -1$

$x_0 = 3$

c) At (3,5), $P = -3$

$P < 0$: $P = -1$

$x_0 = 4$

$y_0 = 5$

Points => (1, 4)
(2, 4)
(3, 5)
(4, 5)

	Plot Lines
0	1, 4
1	2, 4
2	3, 5
3	4, 5
4	Done

Q What is 2D translation? Obtain the composite transformation matrix for general fixed-point scaling.

⇒ 2D Translation

Translation in 2D is the process of moving an object from one position to another in a plane by adding displacement value to its co-ordinates. For a point (x, y) translation by (tx, ty) result $(x', y') = (x + tx, y + ty)$.

In homogeneous co-ordinates, the transformation matrix is:

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

Composite Transformation matrix for Fixed-Point scaling:

Fixed point scaling an object relative to a fixed point (xf, yf) rather than the origin. It involves:

1. Translate the fixed point to the origin: $T(-x_f, -y_f)$.
2. Apply scaling: $S(s_x, s_y)$.
3. Translate back: $T(x_f, y_f)$.

Matrices:

- $T(-x_f, -y_f) = \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$

- $S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- $T(x_f, y_f) = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix}$

Composite matrix $= T(x_f, y_f) \times S(s_x, s_y) \times T(-x_f, -y_f)$

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

$$= \begin{bmatrix} s_x & 0 & -s_x * x_f \\ 0 & s_y & -s_y * y_f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

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

1) $T(x_f, y_f) * \text{Result} = \begin{bmatrix} s_x & 0 & -s_x * x_f + x_f \\ 0 & s_y & -s_y * y_f + y_f \\ 0 & 0 & 1 \end{bmatrix}$

(d) Define Bezier curve. Write some characteristics of this curve.

→ Definition

A Bezier curve is a parametric curve used in computer graphics to model smooth curves. Defined by control points, it is represented as a polynomial function of a parameter t ($0 \le t \le 1$).

For n control points $P0, P1, \ldots, Pn$, the curve is given by

$$B(t) = \sum (n\,C\,i) * (1-t)^{(n-i)} * t^{i} * Pi$$

[where $i = 0$ to n]

($n\,C\,i$ is a binomial coefficient).

Characteristics

1. Control Points Influence: The curve starts at $P0$ ($t=0$) and ends at Pn ($t=1$), with intermediate points shaping the curve without necessarily lying on it.
2. Convex Hull Property: The curve lies within the convex hull of its control points.
3. Smoothness: The curve is continuous and has continuous derivatives upto the $(n-1)^{th}$ order.

4. Affine Invariance: Applying an affine transformation (e.g., scaling, rotation) to control points transforms the curve equivalently.

5. Degree: A Bezier curve with n+1 control points has degree n (e.g., quadratic for 3 points, cubic for 4 points).

Example: For a quadratic Bezier curve with P0(0,0), P1(1,2), P2(2,0), It smoothly interpolates between the end points, controlled by P1.

2023

(a) Describe typical scan Display, random and raster.

→ Scan display are techniques used in computer graphics to render images on a screen. Two primary types are random scan and raster scan:

1) Random Scan Display (Vector Display)

- Draws images by directing an electron beam to specific points on the screen, connecting them with straight lines.
- Only the end points of lines are specified, and the beam moves / display between them (no fixed pattern).

Advantages: High resolution, smooth lines, ideal for line drawings (e.g., CAD systems).

Disadvantages: Cannot display filled areas or complex shaded images; slower for intricate scenes.

2) Raster scan Displays:

- Scans the entire screen line by line (horizontally) from top to bottom, illuminating pixels in a fixed grid pattern.
  - Uses a frame buffer to store pixel intensity values, ~~&~~ refreshed ~~typically~~ periodically. (e.g., 60 Hz)

Comparison: Random scan is like drawing with a pen (precise lines), while raster scan is like painting a grid (pixel-based). Raster ~~scan~~ scan dominates modern displays due to versatility.

(b) Describe Parallel and Perspective projection in brief.

⇒ Projections in computer graphics map a 3D objects onto a 2D plane. Two main types are Parallel and Perspective:

1) Parallel Projection

- Rays from the object to the projection plane are parallel, with no convergence.
- Preserves relative dimensions (no foreshortening), making it useful for technical drawings (e.g., blueprints).
- Types: orthographic (rays perpendicular to plane) and oblique (rays at an angle).

Example: A cube's edges remain parallel on the 2D plane.

2) Perspective Projection:

- Rays converge at a single point (the center of projection or vanishing point mimicking human vision).
- Objects further away appear smaller (foreshortening), adding realism.
- Defined by a view frustum, with a near and far plane.

Example: Railway tracks appear to meet at a point in the distance.

Key Difference:

Parallel projection maintains size regardless of distance, while perspective projection scales objects based on depth, enhancing 3D realism.

(c) Write and explain an algorithm used for hidden surface removal.

→ Z-Buffer (Depth-Buffer) Algorithm:

This is a widely used algorithm used for hidden surface removal, that determines which surfaces are visible by comparing depth values.

Algorithm

1) Initialize two buffers:
   - Frame buffer: stores color values for each pixel (initially background color).
   - Z-buffer: store depth (z-value) for each pixel (initially maximum depth, e.g., infinity).

2) for each polygon in the scene:
   - Rasterize the polygon into pixels.
   - for each pixel (x,y):
     - calculate the depth z of the polygon at (x,y).
     - if z < z-buffer[x,y]:
       - update z-buffer[x,y] = z
       - update frame buffer[x,y] = polygon's color.

3) Repeat for all polygons.

4) Display the frame buffer.

Explanation

- Works by storing the closest depth value for each pixel in the z-buffer.
- If a new pixel's depth is less than the stored value. It's closer and overwrites the previous color and depth.
- Advantages: simple, hardware-supported, works with any polygon order.
- Disadvantages: Requires large memory for the z-buffer, less efficient for transparent objects
- Used in real-time rendering (e.g., video games).

(4) Describe the basic color modeling used in surface rendering in brief.

⇒ Answer: The RGB color model is the best color model used in surface rendering for digital displays.

Descriptions:

- Based on additive color mixing using three primary colors: Red (R), Green (G), Blue (B).

- Each color components is assigned an intensity value (typically 0 to 1 or 0 to 255 in 8-bit systems).
- Combining these primaries produces all visible colors:
  - R+G+B = white (full intensity).
  - (R, G, B) = (0, 0, 0) (is Black, no intensity).
  - R+G = yellow, R+B = magneta, G+B = cyan.

Use in Rendering

- Surface rendering assigns RGB values to pixels based on lighting, materials properties, and shading models (e.g., Phong).
- Stored in frame buffer for display on raster devices like monitors.

Advantages: matches human vision (trichromatic theory), widely supported in hardware.

Limitations: Doesn't account for perceptual uniformity (addressed by models like HSV).

Example: A surface lit with red light might have (255, 0, 0) or (1, 0, 0).

2022-23

@ write the DDA line drawing algorithm. Use the algorithm to find out 4 points between (2,5) and (6,8).

→ The DDA (Digital Differential Analyzer) algorithm generates pixel co-ordinates for a straight line by incrementally calculating points based on the slop. It minimizes computational complexity by using integer arithmetic where possible. The pseudocode is:

Step 1: $\Delta x = x_1 - x_0$

$\Delta y = y_1 - y_0$

Step 2: steps = max(mod(Δx); mod(Δy))

Step 3: x_increment = Δx / steps;

y_increment = Δy / steps

Step 4: $x = x_0$

$y = y_0$

Step 5: Plot (round(x), round(y))

Step 6: for i = 1 to steps:

→ x = x + x_increment

→ y = y + y_increment

→ Plot (round(x), round(y))

Calculate for points between (2,5) and (6,8)

→ Endpoints: $(x_0, y_0) = (2, 5)$

$(x_1, y_1) = (6, 8)$

→ $\Delta x = 6 - 2 = 4$

$\Delta y = 8 - 5 = 3$

→ steps = max $(|\Delta x|, |\Delta y|)$

= max $(4, 3)$

= 4

→ x_increment = Δx / steps

= 4 / 4

= 1

→ y_increment = Δy / steps

= 3 / 4

= 0.75

Starting from (2,5):

1. initial point: x = 2, y = 5 → Plot (2,5)
2. Step 1: x = 2 + 1 = 3

y = 5 + 0.75 = 5.75

Plot (3, 6)

3. Step 2: x = 3 + 1 = 4

y = 5.75 + 0.75 = 6.5

Plot (4, 7)

4. Step 3: x = 4 + 1 = 5

y = 6.5 + 0.75 = 7.25

Plot (5, 7)

5. Step 4: $x = 5+1 = 6$

$y = 7.25 + 0.75 = 8$

plot(6,8)

Points plotted: (2,5)
(3,6)
(4,7)
(5,7)
(6,8)

the 4 points are as follows

(3,6), (4,7), (5,7), (6,8)

(b) ~~2D Translation~~

What is 2D translation? Obtain the composite transformation matrix for general fixed point rotation. (2021 - C)

(c) What is 4 connected region? Write down the flood fill algorithm.

⇒ The flood fill algorithm fills a connected region starting from a seed point, selecting all connected pixels of the same color with a new color. The recursive implementation is:

```
procedure floodFill(x, y, new_color):
    original_color = get_color(x,y)
    if(original_color == new_color or out_of_bounds):
        return
    set_color(x, y, new_color)
    floodFill(x+1, y, new_color) // Right
    floodFill(x-1, y, new_color) // Left
    floodFill(x, y+1, new_color) // Up
    floodFill(x, y-1, new_color) // Down
```

This ensures all 4-connected pixels with the original color are filled, useful for coloring regions in graphics applications. Note, this differs from boundary fill, which stops at a boundary color, highlighting the question's ~~specificity~~ specificity.

(a) write and explain an algorithm used for hidden surface removal. (2022-C)

(b) ~~obtain the composite transformation matrix for general fixed point rotation~~

⇒ ~~Composite~~ Fixed point rotation rotates an object around a specific point (x_f, y_f) by an angle θ, rather than the origin. This requires three steps:

1. Translate the fixed point (x_f, y_f) to the origin: $T(-x_f, -y_f)$.
2. Rotate by angle θ around the origin: $R(\theta)$.
3. Translate back to the original fixed point: $T(x_f, y_f)$

The

(b) What is 2D translation? Obtain the composite transformation matrix for general fixed-point rotation.

⇒ **2D-Translation**

2D Translation refers to the operation of moving every point of an object in a 2D by a certain distance in the horizontal and vertical directions. This transformation is represented by a translation vector $T = (t_x, t_y)$, where

- t_x is the translation distance along the x-axis
- t_y is the translation distance along the y-axis

In matrix form, the translation of a point (x, y) is represented by the following equation:

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where:

- (x, y) is the original point,
- (x', y') is the translated point,
- t_x and t_y are the translation values along the x and y axes, respectively.

-Composite Transformation matrix for general Fixed-Point Rotation

A fixed point rotation is a rotation around a specific point other than the origin. To perform this, we need to ⇒

1. Translate the point of rotation to the origin.
2. Rotate the object around the origin.
3. Translate the point back to its original position.

Given the fixed point rotation (x_0, y_0), the steps are:

1. Translation to the origin:

$$T_1 = \begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Rotation around the origin by an angle θ:

$$R = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3. Translate back to original point:

$$T_2 = \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Now, to get the composite transformation matrix, we multiply these matrices in the order of the operations (first translate to the origin, then rotate, and finally translate back):

$$M = T_2 \cdot R \cdot T_1$$

$$M = \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Carrying out the multiplication gives the composite transformation matrix M:

$$M = \begin{pmatrix} \cos\theta & -\sin\theta & x_0(1-\cos\theta) + y_0\sin\theta \\ \sin\theta & \cos\theta & y_0(1-\cos\theta) - x_0\sin\theta \\ 0 & 0 & 1 \end{pmatrix}$$

This is the composite matrix that represents the general fixed-point rotation.

2023-24

Q Briefly Explain the functioning of CRT monitor.

→ A cathode Ray Tube (CRT) monitor displays images by using electron beams to luminate phosphors on a screen. It's functioning involves:

1. Electron Gun: Located at the back of the tube, it emits electrons when heated. Three guns are used for three colors {red, green, blue}.
2. Deflection coils: Electromagnetic coils (horizontal and vertical) deflect the electron beams to scan the screen in a raster pattern, line by line, from top to bottom
3. Phosphor-coated screen: The front of the tube is coated with phosphor dots or stripes. When electron strike this they emmit light (R, G, B).

4. Shadow Mask: A metal grid ensures electrons hit the correct phosphor dots, producing precise colors and images.

5. Refresh cycle: The beam refreshes the screen 60-100 times per second to maintain a flicker free image.

Key-Points: CRT's are widely used due to their color accuracy and high refresh rates but are bulky and power-intensive compared to modern LCDs.

(b) Consider the line from (0,0) to (4,6). Use the DDA algorithm to rasterize this line. Write one disAdvantage of the DDA algorithm.

→ DDA (Digitally Differential Analyzer) Algorithm

The DDA algorithm calculates pixel co-ordinates for a line by incrementing based on the slope.

for (0,0) to (4,6)

dx = 4-0 = 4 dy = 6-0 = 6

steps = max(|dx|, |dy|) = 6

x_increment = dx/steps = 4/6 = 2/3 = 0.67

y_increment = dy/steps = 6/6 = 1

Starting at (0,0):

1) $x = 0, y = 0$ | Plot (0,0)

2) $x = 0 + 0.67 = 0.67, y = 0 + 1 = 1$ | Plot (1,1)

3) $x = 0.67 + 0.67 = 1.34, y = 1 + 1 = 2$ | Plot (1,2)

4) $x = 1.34 + 0.67 = 2.01, y = 2 + 1 = 3$ | Plot (2,3)

5) $x = 2.1 + 0.67 = 2.68, y = 3 + 1 = 4$ | Plot (3,4)

6) $x = 2.68 + 0.67 = 3.35, y = 4 + 1 = 5$ | Plot (3,5)

7) $x = 3.35 + 0.67 = 4.02, y = 5 + 1 = 6$ | Plot (4,6)

points rasterized ⇒ (0,0), (1,1), (1,2), (2,3), (3,4), (3,5), (4,6)

One Dis Advantage of DDA Algorithm

→ Floating-Point Arithmetic: DDA uses floating points for calculations (e.g., 0.67) which are computationally expensive and slower on older hardware compared to integer-based algorithm like Bresenham's.

(c) Write and explain an algorithm used for hidden surface removal. (2022-E)

(d) What do you mean by screen resolution? If a video has a resolution of 1920×1080 pixels and needs to be resized to width of 1280 pixels while maintaining the same aspect ratio, what would be the new height of the video? What is Persistence?

⇒ **Screen Resolution**

Screen Resolution refers to the number of pixels (width * height) displayed on a screen, determining image clarity. Higher resolutions (e.g., 1920 * 1080) offer sharper details. It's typically measured in pixels, like 1920×1080 (Full HD).

New Height calculation

- Original resolution: 1920 * 1080
- Aspect ratio = width/height = $\frac{1920}{1080} = \frac{16}{9} \approx 1.777$
- New width = 1280 pixels
- To maintain Aspect ratio:

  New height = new width / Aspect ratio

  $= 1280 \times \frac{9}{16} = 720$ px (height)

Persistence:

Persistence (or persistence of vision) is the phenomenon where the human eye retains an image for a brief moment (about 1/25th of a second) after it disappears. In display, this allows smooth motion perception when frames refresh rapidly (e.g., 60 Hz), as the eye blends consecutive images.

5x4/SEC-4 MATH

Q what are the advantages of laser printers? write down the difference between random scan and raster scan display?

→ Advantages of Laser Printers

1. High quality: Produce sharp text and graphics due to precise toner placement.
2. Speed: Faster printing, especially for large volumes, compared to inkjets.
3. Durability: Toner doesn't smudge or fade easily unlike ink.
4. Cost Efficiency: Lower cost per page for higher volume printing.

Difference between Random scan and Raster scan

Random scan	Raster scan
⇒ Draws lines directly between specified points using an electron beam	Scan the screen line by line in a fixed grid, illuminating pixels.
High resolution, smooth lines, used in early CAD systems	Uses a frame buffer, supports filled areas and shading (e.g., modern monitors).
Can't display filled areas or complex images.	Lower resolution, may show aliasing (jagged edges).

Key Difference:
Random scan is line-based and precise,
Raster scan is pixel-based and versatile.

(b) Define image and object. How an image is represented mathematically?

⇒ Image: A 2D representation of an object or scene, typically as a grid of pixels with color / intensity values.

Object: ~~A physi~~ A physical or virtual entity in a scene (e.g., a 3D model or 2D shape) with geometric properties like position and size.

Mathematical Representation of an Image

- An image is represented as a 2D function $(f(x,y))$, where:
  - x and y are spatial coordinates (pixel positions).
  - $(f(x,y))$ is the intensity or color value at that point.
- For a grayscale image: $f(x,y)$ ranges from 0 (black) to 255 (white).
- For a color image: $f(x,y) = R(x,y), G(x,y), B(x,y)$ where RGB are Red, Green, Blue intensities (e.g., 0-255 each).
- Example: A 3×3 grayscale image might be:

$$\begin{bmatrix} 100 & 150 & 200 \\ 50 & 75 & 125 \\ 0 & 25 & 50 \end{bmatrix}$$

This matrix defines pixel values; enabling digital processing and rendering.

(c) Draw a line from (0,0) to (-6,-4) by using Bresenham's line drawing algorithm.

→ for a line with slope $|m| \leq 1$ it uses a decision parameter to choose pixels. Since the slope here is > 1, we adjust by stepping along y.

- Endpoints: (0,0) to (-6,-4)
- $dx = -6 - 0 = -6$

  $dy = -4 - 0 = -4$
- $|dy| > |dx|$, so step along y (4 steps); x decreases
- Slope $m = dy/dx = -4/-6 = 2/3$, but we use y as the driving axis.
- Initial $P = 2|dx| - |dy| = 2*6 - 4 = 8$

Steps

1) • Start: (0,0), Plot (0,0)

2) $y = -1$ $P = 8$

   • $P > 0$: $x = -1$

   $P = 8 + 12 - 8 = 12$ | Plot (-1, -1)

3) $y = -2$, $P = 12$

   • $P > 0$: $x = -2$

   $P = 12 + 12 - 8 = 16$ | Plot (-2, -2)

4) $y = -3$, $p = 16$

$p > 0$: $x = -3$

$p = 16 + 12 - 8 = 20$ Plot $(-3, -3)$

5) $y = -4$, $p = 20$

$p > 0$: $x = -4$.

$p = 20 + 12 - 8 = 24$ Plot $(-4, -4)$

- Adjust to reach to $(-6, -4)$.

Continue logic but typically Interpolate $(-5, -4)$ $(-6, -4)$.

Points: $(0,0)$, $(-1,-1)$, $(-2,-2)$, $(-3,-3)$, $(-4,-4)$, $(-5,-4)$, $(-6,-4)$.

⑥ Define aspect ratio. A screen has 1024 scan lines with aspect ratio (4:3) and bit depth 16. Then how many bit per pixel are required to show 60 frames per second?

⇒ Aspect Ratio:

Aspect Ratio is the ratio of width to height of a display or image (e.g., 4:3 means 4 unit wide to 3 unit high). It determines the shape of the screen.

Calculations

- Given: 1024 scan lines (height), aspect ratio (4:3).
- Width = (4/3) * height = 4/3 * 1024
  = 1366 pixels. (common resolution)
- Resolution = 1366 * 1024 pixels.
- Bit depth = 16 bits per pixel (color depth)
- Bits per frame = 1366 * 1024 * 16 = 22401536 bits.
- Frames per second = 60
- Bits per second = 22401536 * 60
  = 1344092160 bits/second.

~~Bits per pixel requirements~~

- ~~The question asks~~

(c) Explain Cohen-Sutherland line clipping algorithm with suitable example.

⇒ Cohen-Sutherland Algorithm

This clips lines against a rectangular window by assigning 4-bit region codes to endpoints and checking visibility.

- Window: Define (xmin, ymin) to (xmax, ymax)

- Region codes: 0000 (inside), bits for left (1000), right (0100), below (0001), above (0010).

- Steps:
  a. Assign codes to endpoints
  b. If both 0000: fully inside
  c. If AND ≠ 0: fully outside
  d. If partially outside: clip at boundaries, recompute codes.

Example: window (1,1) to (4,4) line (0,2) to (5,3).

- (0,2): code = 1000 (left of $x_{min}=1$)
- (5,3): code = 0100 (right of $x_{max}=4$)
- AND ≠ 0000 (not fully outside)
- clip at $x=1$: $y = 2 + (3-2)/(5-0) * (4-0) = 2.8$
  → (4, 2.8)
- Clipped line: (1, 2.2) to (4, 2.8).

(F) Explain DDA line drawing algorithm with an example.

⇒ **DDA Algorithm**: The Digital Differential Analyzer calculates pixels point incrementally based on the slope.

- Steps:
  a. $dx = x1 - x0$
     $dy = y1 - y0$
  b. $steps = \max(|dx|, |dy|)$
  c. $x\_inc = dx/steps$
     $y\_inc = dy/steps$
  d. Plot points by incrementing x, y, rounding to integers.

Example: Line from (1,1) to (3,2)

- $dx = 3-1 = 2$, $dy = 2-1 = 1$
- $steps = \max(2,1) = 2$
- $x\_inc = 2/2 = 1$, $y\_inc = 1/2 = 0.5$
- Points:
  a. (1,1) → Plot (1,1)
  b. $x = 1+1 = 2$, $y = 1+0.5 = 1.5$ → Plot (2,2)
  c. $x = 2+1 = 3$, $y = 1.5+0.5 = 2$ → Plot (3,2)

Points are: (1,1)
(2,2)
(3,2)

Sec-4 2022

(a) Write down the applications of computer Graphics.

⇒ Computer Graphics has a wide range of applications across industries:

1. Entertainment

used in video games, movies, and animations for rendering 3D models, special effects, and virtual environments (e.g., Pixar films).

2. Design and Engineering

CAD (Computer-Aided Design) for architectural blueprints, mechanical parts, and product prototypes.

3. Scientific Visualization

Simulating and visualizing complex data, like weather patterns, molecular structure, or medical imaging (e.g., MRI scans).

4. Education and Training

Interactive simulations, such as flight simulators or virtual labs, for skill development.

5. Advertising and Art

Creating digital illustrations, logos, and marketing visuals with tools like Adobe Photoshop. These applications leverage graphics for visualization, interaction, and realism, making it integral to modern technology.

(b) A straight line joining the origin and the point (4,3) is rotated counter clockwise by an angle of 45° (degree). Find the rotation matrix and the resultant point.

⇒ The rotation matrix for counter clockwise rotation by an angle θ is:

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

Given $\theta = 45°$, we can substitute the values:

$$R(45°) = \begin{bmatrix} \cos(45°) & -\sin(45°) \\ \sin(45°) & \cos(45°) \end{bmatrix}$$

$$R(45°) = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Resultant point:

The original point is (4,3). To find the resultant point, we multiply the rotation matrix by the original point.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} * \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

Performing the multiplication

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} (1/\sqrt{2})(4) + (-1/\sqrt{2})(3) \\ (1/\sqrt{2})(4) + (1/\sqrt{2})(3) \end{vmatrix}$$

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} 4/\sqrt{2} - 3/\sqrt{2} \\ 4/\sqrt{2} + 3/\sqrt{2} \end{vmatrix} = \begin{vmatrix} 1/\sqrt{2} \\ 7/\sqrt{2} \end{vmatrix}$$

$x' = 1/\sqrt{2}$

$y' = 7/\sqrt{2}$

So the resultant point is

$$(1/\sqrt{2}, 7/\sqrt{2}) \approx (0.707, 2.474)$$

(c) What is 4-connected region? Write down boundary fill algorithm. (2021-a)

(d) Write midpoint circle generation algorithm.

→ This algorithm draws a circle by calculating pixel positions using the midpoint of possible points, optimizing with integer arithmetic. For a circle centered at (x_c, y_c) with radius r.

Procedure

Midpoint circle (x_c, y_c, r):

1) Initialize $x = 0$ $y = r$

2) $p = 1 - r$ // Initial decision parameter

3) ~~while x<y:~~

3) while x<y:

- Plot all 8 symmetric points:

  (xc+x, yc+y), (xc-x, yc+y)

  (xc+x, yc-y), (xc-x, yc-y)

  (xc+y, yc+x), (xc-y, yc+x)

  (xc+y, yc-x), (xc-y, yc-x)

- $x = x+1$
- If $P<0$:
  - $P = P+2x+1$
- Else:
  - $y = y-1$
  - $P = P+2x-2y+1$

Explaination

- uses symmetry to plot 8 points per iteration.
- Decision parameter P decides weather to move vertically, reducing floating-point calculations.
- Efficient for raster displays (e.g., drawing a circle of radius 5).

(E) Define linear transformation. write the procedure to fill polygon using flood fill.

→ Linear transformation

A linear transformation is a function ~~$T: \mathbb{R}^n \to \mathbb{R}^m$~~ $T: \mathbb{R}^n \to \mathbb{R}^m$

that preserves vector addition and scalar multiplication: $(T(u+v) = T(u) + T(v))$ and $(T(cu) = cT(u))$. In graphics, it's represented by a matrix (e.g., scaling, rotation) applied to coordinates, maintaining straight lines and origin mapping.

Flood fill Procedure

This fills a polygon by replacing a region's color with a new color straight from a seed point.

```
FloodFill(x, y, new_color, old_color):
1) If (x,y) is outside of bounds or not old_color:
      -Return
2) Set pixel (x,y) to new_color
3) Recursively fill 4 neighbors:
   - FloodFill(x+1, y, new_color, old_color) // Right
   - FloodFill(x-1, y, new_color, old_color) // Left
   - FloodFill(x, y+1, new_color, old_color) // UP
   - FloodFill(x, y-1, new_color, old_color) // down
```

(F) Explain DDA line drawing algorithm with an example. (SEE-4, 2021-F)

10 x 1

2021

(a) Explain the working principle of CRT monitor using

1. Beam penetration method
2. Shadow-mask method

⇒ A color CRT (cathode ray tube) monitor displays images by directing electron beams to excite phosphors on a screen. Two method for producing color in CRTs are the Beam Penetration method and the Shadow-mask method.

1) Beam Penetration method

principle: This method uses a single electron gun and phosphor-coated screen with multiple layers of different phosphors (typically red and green). The color displayed depends on the depth to which the electron beam penetrates the phosphor layers, controlled by varying the beam's voltage.

Working

- The screen is coated with two phosphor layers: an outer layer emitting red light and an inner layer emitting green light.
- Low-voltage electron beams excite only the outer (red) layer, producing red.
- High-voltage beams penetrate deeper, exciting the inner (green) layer, producing green.
- Intermediate voltages create a mix of red and green (e.g., orange, yellow).
- The electron gun's voltage is ~~modulated~~ modulated to control beam energy, thus selecting the desired color.

Advantages: simple design, no need for multiple voltage guns or complex alignment.

Limitations: Limited to a few colors (red, green, and blends), poor color precision, and slower response due to voltage switching.

Use case: Early color CRTs, for basic displays (e.g., oscilloscopes).

2) Shadow-Mask Method

Principle: This method uses three electron guns (for red, green, blue) and shadow mask to direct beams to specific phosphor dots, producing a full range of colors via additive mixing.

Working:

- The screen is coated with triads of phosphor dots (red, green, blue).
- Three electron guns emit beams, each aimed at one type of phosphor.
- A shadow mask, a perforated metal sheet is placed behind the screen. It ensures each beam hits only its corresponding phosphor (e.g., red beam hits red phosphor).
- Deflection coils scan the beams in a raster pattern, illuminating pixels.
- Color is produced by varying the intensity of each gun (e.g., red + green = yellow, all three = white).

Advantages: Produces a wide range of colors, high resolution, and accurate color reproduction, making it standard for modern CRTs.

Limitations: Complex alignment of mask and guns, lower brightness due to mask blocking some electrons, and bulky design.

Use case: ~~Coins~~ TV's, computer monitors before LCDs.

Summery: Beam penetration is simpler but limited in color range, while shadow-mask offers vibrand, full-color displays at the cost of complexity.

(b) Develop: Cohen-Sutherland line clipping algorithm.

The Cohen-Sutherland line clipping Algorithm clips a line segment against a rectangular window by using region codes to determine visibility; It's efficient for 2D graphics, reducing unnecessary computations.

Development of the Algorithm

- Objective: clip a line segment with end points (x_1, y_1) and (x_2, y_2) to lie within a rectangular window defined by (x_{min}, y_{min}) and (x_{max}, y_{max}).

Region codes: Divide the 2D space into nine regions around the window using a 4-bit code for each point.

- Bit 1 (left): 1 if $x < x_{min}$, else 0
- Bit 2 (right): 1 if $x > x_{max}$, else 0
- Bit 3 (below): 1 if $y < y_{min}$, else 0
- Bit 4 (above): 1 if $y > y_{max}$, else 0

- Example: inside window = 0000, left of window = 1000, above and left = 1001.

Clipping logic:

- Trivially Accept: if both endpoints have code 0000, the line is fully inside.
- Trivially Rejected: if the bitwise AND of the codes is non-zero, the line is fully outside (both points on same side of a boundary).
- Partial clipping: if neither accept nor reject, clip the line at window boundaries and recompute codes.

Algorithm

Procedure Cohen Sutherland Clip ($x_1, y_1, x_2, y_2, x_{min}, x_{max}, y_{max}$):

1) assign region codes to (x_1, y_1) and (x_2, y_2):
   - code1 = compute Region code (x_1, y_1)
   - code2 = compute Region code (x_2, y_2)
2) while true:
   - if (code1 == 0000 AND code2 == 0000):
     - Accept line (x_1, y_1) to (x_2, y_2)
     - Return
   - if (code1 AND code2 ≠ 0000):
     - Reject Line
     - Return
   - Choose an end point outside (e.g., code1 ≠ 0000):
     - if bit 1 (left): intersect with $x = x_{min}$, compute y
     - if bit 2 (right): intersect with $x = x_{max}$, compute y
     - if bit 3 (below): intersect with $y = y_{min}$, compute x
     - if bit 4 (above): intersect with $y = y_{max}$, compute x
     - update end point (x_1, y_1) with intersection point
     - Recompute code1
   - Repeat for other endpoint if needed
3) Output clipped line segment

Intersection calculation

- for a line from (x_1, y_1) to (x_2, y_2), slope $m = \frac{y_2 - y_1}{x_2 - x_1}$
- Example: clip at $x = x_{min}$:
  $y = y_1 + m(x_{min} - x_1)$.
- Similarly for other boundaries (e.g., $y = y_{max}$: $x = x_1 + \frac{y_{max} - y_1}{m}$).

Explanation

- The algorithm iteratively clips the line against window boundaries, recomputing region codes until the line is fully inside or rejected.
- **Advantages:** fast, handles all cases efficiently, widely used in graphics pipelines.
- **Limitations:** limited to rectangular windows, requires floating-point calculations for intersections.

Q23 (10×1)

Name three basic geometric transformations. Describe every such transformation for 3D objects.

Geometric transformations are operations that alter the position, orientation, or size of 3D objects in computer graphics. They are essential for modeling, animation, and rendering in virtual environments. The three basic geometric transformations for 3D objects are translation, rotation, and scaling.

Each is described below.

1) Translation

Translation moves a 3D object from one position to another in 3D space without changing its size or orientation.

Description

- In 3D space, a point on an object is represented by coordinates (x, y, z).
- Translation shifts the objects by adding displacements values (T_x, T_y, T_z) along the x, y, z axes, respectively.
- The new position of a point (x, y, z) after translation is: $(x', y', z') =$

$$= (x + T_x, y + T_y, z + T_z)$$

Example: If an object (e.g: a cube) is translated by (2, 3, -1), every point of the cube moves 2 units along the X-axis, 3 units along the Y-axis, and 1 unit backward along the z-axis.

- Effect: The object retain it's shape size and orientation but appears at a new location in the 3D space.

- Matrix Representation: Translation is often represented using a 4×4 homogeneous transformation matrix.

2) Rotation

Rotation changes the orientation of a 3D object by rotating it around a specific axis (X, Y, or Z) or a custom axis.

Description

- In 3D space, rotation can occur around the x-axis y-axis, or z-axis, defined by an angle θ (in radians or degrees).

Each rotation transforms the co-ordinates (x, y, z) of the objects points based on trigonometric functions (sine and cosine).

- Rotation around the z axis (example):

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

- Similar formulas can apply for rotation around the x-axis or y-axis, affecting different co-ordinate pairs.

- Example: Rotating a 3D model (e.g. a sphere) by 90° around the z-axis reorients it so that points originally along the x-axis lie along the y-axis.

Effect: The objects shape and size remain unchanged, but it's orientation in 3D space changes.

- Matrix Representation: for rotation around the z-axis by angle θ: similar matrix exists for x-axis and y-axis rotation.

Note: for rotation around arbitrary axis, a more complex transformation (e.g., using quaternions or a composite matrix) is used, but basic rotations are typically axis-aligned.

3) Scaling

Scaling changes the size of a 3D object by stretching or shrinking it along the x, y, and z axes.

Descriptions

- Scaling multiplies the coordinates (x, y, z) of each points by scaling factors (S_x, S_y, S_z) along the x, y, z axes, respectively. and new coordinate $(x', y', z') = (S_x \cdot x, S_y \cdot y, S_z \cdot z)$

Types of scaling:

- Uniform scaling: $S_x = S_y = S_z$, which enlarges or shrinks the object proportionally (e.g., double the size).
- Non-uniform scaling: Different scaling factors (e.g., $S_x = 2$, $S_y = 1$, $S_z = 0.5$) stretch the object unevenly, altering its proportions.

Example: scaling a 3D cube by 2,2,2 doubles it's size in all directions, while (2,1,1) stretches it along the x-axis.

Effect: The object's size changes, and it's shape may change if scaling is non uniform. The position of the object's origin remains fixed unless combined with translation.

Matrix Representation

This matrix scales the object vertices relative to the origin.

(b) Write Bresenham's circle drawing algorithm and use it to compute 4 points on any quadrant of the circle $x^2 + y^2 = 16$

⇒ Bresenham's circle drawing algorithm is used to plot a circle on a raster grid by selecting the closed pixels to the ideal circle's path. It optimized for digital displays, minimizing computations by using integer arithmetic.

Algorithm overview

- The algorithm exploits the circle's symmetry (8-way symmetry) to compute points in one octant (e.g., the first octant of the first quadrant where $x \geq y \geq 0$) and maps them to other octants.
- For a circle centered at the origin with radius r, the equation is $x^2 + y^2 = r^2$

Algorithm steps

1) Input: Radius r of the circle (for $x^2 + y^2 = 16$, $r = \sqrt{16} = 4$)

2) Initialize
   - Start at point $(x, y) = (0, r)$, i.e., $(0, 4)$ for $r = 4$
   - Set initial decision parameter $d = 3 - 2r$

3) Iterate:
   - while $x \leq y$:
     - Plot the point (x, y) and it's symmetric points in all 8 octants (x, y), (y, x), $(-x, y)$, $(-y, x)$, $(x, -y)$, $(y, -x)$ $(-x, -y)$, $(-y, -x)$

- Update the decision parameter:
  - if $d < 0$:
    - Choose the next pixel by incrementing x only: $x = x + 1$
    - Update $d = d + 4x + 6$
  - Else ($d \geq 0$):
    - Choose the next pixel by incrementing x and decrementing y: $x = x + 1$, $y = y - 1$
    - Update $d = d + 4 \cdot (x - y) + 10$
- Continue until $x > y$, as the first octant is complete

Pseudocode

```
bresenham circle (r):
    x = 0, y = r
    d = 3 - 2 * r
    while (x <= y):
        plot(x, y), plot(y, x), plot(-x, y), plot(-y, x)
        plot(x, -y), plot(y, -x), plot(-x, -y), plot(-y, -x)
        if (d < 0):
            d = d + 4 * x + 6
        else:
            d = d + 4 * (x - y) + 10
            y = y - 1
        x++
```

Compute 4 points for, $x^2+y^2=16$

~~Soln for 16~~

So, $r = \sqrt{16} = 4$

⇒ Initialize $x, y := 0, 4$

& $d = 3 - 2*r = -5$

Iterate:

Point 1 : $x, y = 0, 4$

Plot $0, 4$

$d = -5 < 0$

$d = -5 + 4 \cdot 0 + 6 = 1$

$x = 0+1 = 1, y = 4$

Point-2 $x, y := 1, 4$

Plot $(1, 4)$

$d = 1 \geq 0$

$d = 1 + 4 \times (1-4) + 10$

$= -1$

$x = 2$

$y = 3$

Point-3 $x, y = 2, 3$

Plot $(2, 3)$

$d = -1 \leq 0$

$d = 1 + 4 \times 2 + 6$

$d = 13$

$x = 3, y = 3$

Point-4. x, y = 3, 3

Plot (3, 3)

d = 13 >= 0, so

d = 13 + 4 · (3 - 3) + 10 = 23

x = 4, y = 2

so the 4 points for, $x^2 + y^2 = 16$

are; (0, 4), (1, 4), (2, 3), (3, 3)

2022-SEE-4 / 10×1

(3) (a) Explain the working principle of a CRT monitor using:

1) Beam penetration method.

2) shadow-mask method.

(b) Develop Cohen-Sutherland line clipping algorithm.

2021-SEE-4 / 10×1

(3) write Midpoint circle generation algorithm.

Overview

- The algorithm is based on Bresenham's circle algorithm but uses the midpoint between two possible pixels to decide which to plot.
- It exploits the circle's 8-way symmetry, computing points in the first octant (where $0 \le y \le x \le r$) and mapping them to other octants.

- for a circle centered at (0,0) with radius r, the equation is $x^2+y^2=r^2$
- The algorithm evaluates the circle's implicit equation $f(x,y) = x^2+y^2-r^2$ at the midpoint between candidate pixels

Algorithm Steps

1) Input: Radius of the circle r
2) Initialize:
   - Start point $(x,y) = (0,r)$
   - Initial decision parameter: $d = 1-r$
3) Iterate:
   - while $(x \leq y)$:
     - Plot the point (x,y) and its symmetric points in all 8 octants: (x,y), (y,x), $(-x,y)$, $(-y,x)$, $(x,-y)$, $(y,-x)$, $(-x,-y)$, $(-y,-x)$
     - Update the decision parameter
     - if $d<0$ (midpoint is inside the circle):
       - Choose the east pixel:
         $x = x+1$
       - Update $d = d+2x+3$.
     - Else ($d \geq 0$) midpoint is outside or on the circle:
       - Choose the southeast pixel:
         $x = x+1$, $y = y-1$
       - Update $d = d+2x+2y+5$.
     - Continue until $x > y$.

Pseudocode:

```
midpoint circle (r):
    x = 0, y = r
    d = 1 - r
    while (x <= y):
        Plot(x,y), Plot(y,x), Plot(-x,y),
        Plot(-y,x), Plot(x,-y) Plot(-y,-x)
        Plot(-x,-y), Plot(-y,-x)
        if (d < 0):
            d = (d + 2*x + 3)
        else:
            d = d + 2*(x-y) + 5
            y = y - 1
        x = x + 1
```

(a) Draw a circle with center (0,0) and radius 9

→ Step 1: ~~x, y = 0, 0 & r = 9~~

$x^2 + y^2 = 81$ $[0 \le x \le y \le 9]$

Step 2:

x, y = 0, 9

d = 1 - 9 = -8

Step-3

Point 1 (x, y) = (0, 9)

Plot all 8 for 0, 9

d = -8 < 0, so

d = -5

x = 1, y = 9

Point 2

Plot(1,9)

d = -5 < 0

d = 0

x = 2, y = 9

Point-3

Plot(2,9)

d = 0 ≥ 0

d = -9

x = 3, y = 8

Point-4

Plot(3,8)

d = -9 < 0

d = 0

x = 4 y = 8

point-5

Plot(4,8)

d = 0 ≥ 0

d = 3

x = 5, y = 7

Point-6

Plot(5,7)

d = -3 < 0

d = 10

x = 6, y = 7

Point-7

Plot(6,7)

d = 10 ≥ 0

x = 7, y = 6

Point-8

Plot(7,6)

since x = 7 > y = 6

Stop the iteration

first octant complete

All points of first quadrant are

(0,9) (1,9) (2,9) (3,8)

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

(9,0) (9,1) (9,2) (8,3)

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

3) Briefly discuss about anti-aliasing technique.

→ Anti-aliasing is a technique used to reduce the jagged or "staircase" appearance (aliasing) of line and curves, such as circles, on a raster displays.

- Aliasing occurs because pixels are discrete, causing smooth curves to appear as stepped edges.

Techniques

1) Supersampling

- Render the image at a higher resolution (e.g., 4x), then average pixel values to produce the final image.
- Improves smoothness but is computationally expensive

2) Area sampling

- Calculate the fraction of each pixel covered by the circle and set its intensity proportionally.
- Produces smoother edges by varying pixel brightness.

4) Wus Anti Aliasing.

- for lines or curves, adjust the intensity of pixels based on their distance from the ideal path
- for a circle, pixels near the boundary are shaded with partial intensity.

4) Filtering:

- Apply a low-pass filter (e.g., Gaussian blur) to blend pixel intensities, reducing sharp transitions.

(b) What is the significance of Geometric transformations?

⇒

1) object manipulation

- Transformations like translation, rotation and scaling allow objects to be moved, re-oriented or resized in a virtual scene. This is essential for modeling, animation, and designing complex scene (e.g., moving a car in a game or rotating a 3D model).

2. Scene composition

- Transformations enable the placement of multiple objects relative to each other in a scene. For example, combining transformation to position a character's limbs in an animation or align objects in a 3D environment.

3. Animation and simulation

- Transformations are used to create smooth animations by incrementally changing an object's position (translation),

orientation (rotation), or size (scaling) over time, for instance, rotating a planet around its axis simulates day-night cycles.

4. Efficiency and Matrix Representation

- Transformations are implemented using matrix operations, allowing efficient computation in graphics pipelines (e.g., GPUs). Homogeneous coordinates simplify the combinations of multiple transformations into a single matrix, streamlining rendering processes.

These capabilities make ~~graphics~~ geometric transformations indispensable for creating realistic and interactive graphics in applications like games, simulations, CAD software, and virtual reality.

2022-23 10X1

@ Discuss the advantages and disadvantages of using raster graphics versus vector graphics

→ Raster Graphics

Definition: Raster graphics represent images as a grid of pixels; each with a specific color or intensity (e.g., bitmap image like PNG, JPEG)

Advantages

- Rich Detail and Realism: Raster graphics excel at rendering complex images with continuous tone variations, such as photographs or detailed textures, due to their pixel-based structures.
- Wide Compatibility: Supported by most display devices and software, making them ideal for digital screens, web graphics and image editing tools.

Disadvantages

- Scalability issues: Scaling raster images (enlarging or reducing) causes pixelation or loss of quality, as pixels are fixed in resolution.
- Large file sizes: High-resolution images require significant storage, especially for detailed or large images.

Vector Graphics

Definition: Vector graphics represent images using mathematical equations (lines, curves, shapes) defined by points, paths, and attributes. (e.g., SVG, EPS files).

Advantages

- Scalability: Vector graphics can be scaled infinitely without loss of quality, as they are resolution-independent, making them ideal for logos, icons and print media.
- Small file size: Store only geometric descriptions, resulting in smaller files compared to high-resolution raster images.

Disadvantages

- Limited Realism: Struggle to represent complex photorealistic images with continuous tones, as they are better suited for simple or stylized designs.
- Complex Rendering: Require more computational effort to render curves and shapes, especially in real-time applications like games.

(a) write a short note on shadow masking method.

⇒ Definition

The shadow masking method, also known as shadow masking, is a technique used in computer graphics to render shadow in 3D scenes, particularly in real-time rendering (e.g., games). It projects an objects shadow from a light source's perspective to determine shadow areas.

Process

1) Shadow map creation
   - Render the scene from the light source's perspective, storing depth information in

a shadow map (a texture that records the distance from the light to the nearest surface).

2) Scene Rendering
- Render the scene from the camera's perspective, for each pixel, compare its depth (distance from the light) with the shadow map's depth value.
- if the pixel's depth is greater than the shadow map's depth, the pixel is in shadow (occluded by another object); otherwise, it is lit.

3) Shadow Applications
- Apply shading to shadowed pixels (e.g., darken them) to create the shadow effect.

Advantages
- Efficiency: Suitable for realtime applications due to its relatively fast computation using graphics hardware.
- Variety: Works with complex scenes and dynamic objects, as shadow map can be updated per frame.

Disadvantages
- Aliasing Artifacts: Shadows may have jagged edges or 'shadow acne' (self-shadowing errors) due to limited shadow map resolution.
- Soft Shadows Limitation: Basic shadow mapping produces hard shadows, requiring additional techniques (e.g., percentage-closer filtering) for soft shadows.

Applications

- Widely used in video games, simulations, and 3D rendering software to add realism by simulating how light is blocked by objects.

(b) Develop Liang-Barsky line clipping algorithm?

⇒ The Liang-Barsky line clipping algorithm efficiently clips a 2D line segment against a rectangular clipping window by parameterizing the line and computing intersections. Below, we develop the algorithm concisely, including its logic, key equations, and pseudocode.

Purpose

- Objective: Clip a line from (x_1, y_1) to (x_2, y_2) against a window defined by x_{min}, x_{max}, y_{min} and y_{max}, outputting the visible portion.

Advantages: Faster than Cohan-Sutherland by directly calculating intersection parameters using a parametric approach

Development of the Algorithm

1) Parametric Equation

- Line from (x_1, y_1) to (x_2, y_2):

$$x = x_1 + t(x_2 - x_1) = x_1 + t\Delta x$$

$$y = y_1 + t(y_2 - y_1) = y_1 + t\Delta y$$

where $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$, $t = 0$ at start, $t = 1$ at end.

2) clipping constraints:

- The line must satisfy:

$x_{min} \le x_1 + t\Delta x \le x_{max}$, $y_{min} \le y_1 + t\Delta y \le y_{max}$

- Rewrite as four inequalities:
  - Left: $t \ge \frac{x_{min} - x_1}{-\Delta x}$
  - Right: $t \le \frac{x_{max} - x_1}{\Delta x}$
  - Bottom: $t \ge \frac{y_{min} - y_1}{-\Delta y}$
  - Top: $t \le \frac{y_{max} - y_1}{\Delta y}$

2) Parameter calculation

- Define:
  - $p_1 = -\Delta x$, $q_1 = x_1 - x_{min}$
  - $p_2 = \Delta x$, $q_2 = x_{max} - x_1$
  - $p_3 = -\Delta y$, $q_3 = y_1 - y_{min}$
  - $p_4 = \Delta y$, $q_4 = y_{max} - y_1$
- Compute $t_k = \frac{q_k}{p_k}$ (if $p_k \neq 0$):
  - $p_k < 0$: Entering parameter (update $t_0 = \max(t_0, t_k)$).
  - $p_k > 0$: Leaving parameter (update $t_1 = \min(t_1, t_k)$).
  - $p_k = 0$: if $q_k < 0$, reject line; else, ignore boundary

Logic

- The algorithm checks each boundary constraint, updating the valid t range. It accepts the line segment if a valid portion lies within [0, 1] and computes the clipped endpoints directly.

Pseudocode

```
liangBarskyClip(x1, y1, x2, y2, xmin, xmax, ymin, ymax):

    dx = x2 - x1
    dy = y2 - y1
    p = [-dx, dx, -dy, dy]
    q = [x1 - xmin, xmax - x1, y1 - ymin, ymax - y1]
    t0, t1 = 0.0, 1.0

    for k in range(4):
        if p[k] == 0:
            if q[k] < 0:
                return False, None, None, None, None
        else:
            t = q[k] / p[k]
            if p[k] < 0:
                t0 = max(t0, t)
            else:
                t1 = min(t1, t)
    if (t0 > t1):
        return False, None, None, None, None

    x1_clip = x1 + t0 * dx
    y1_clip = y1 + t0 * dy
```

$x_{2\_clip} = x_1 + t_1 * dx$

$y_{2\_clip} = y_1 + t_1 * dy$

return True, $x_{1\_clip}, y_{1\_clip}, x_{2\_clip}, y_{2\_clip}$

Summery

Steps

a) Parameterize the line segment using t.

b) Formulate clipping constraints as inequalities

c) Compute entering and leaving parameters (t_k) for each boundary.

d) determine the valid t interval $[t_0, t_1]$

e) if $t_0 \leq t_1$, calculate endpoints; otherwise, reject the lines.

Advantages

- more efficient than cohen-sutherland, as it computes intersections directly
- Handles edges cases (e.g., parallel lines) robustly.

Applications: used in 2D graphics, systems, CAD tools, and rendering pipelines to clip lines against viewports or windows.

2023-24

(b) Write Bresenham's circle drawing algorithm.

> Bresenham's circle drawing algorithm is an efficient method for drawing circles on a raster grid by determining which pixels to illuminate based on a circle's equation. It minimizes computational complexity by using integer arithmetic and exploiting the circle symmetry to calculate points in all eight octants from computations in one octant.

Algorithm Description

1) Initialization, for a circle centered at the origin with radius r, start at point (0,r) i.e, $(x,y) = (0,r)$.

- Define the initial decision parameter $p_i = 3 - 2r$, which is derived from the midpoint circle drawing algorithm to decide whether to move east or southeast in the first octant.

2) Iteration

- while $x \leq y$ compute the next pixel in the first octant (where $0 \leq x \leq y \leq r$).

- for each point (x, y):
  - if $p < 0$, choose the east pixel $(x+1, y)$, and update $p = p + 4x + 6$
  - if $p \geq 0$, choose the southeast pixel $(x+1, y-1)$, and update $p = p + 4(x-y) + 10$
  - Increment x. If southeast pixel was chosen, decrement y.
- use symmetry to generate points in all eight octants:
  - From (x, y), the points are:

octants > (x, y); $(-y, x)$, $(y, -x)$, $(x, -y)$
$(-x, -y)$, $(-y, -x)$, (y, x), $(-x, y)$

3) Termination:
- Stop when $x > y$, as all points in the first octant have been computed, and symmetry covers the rest.

pseudocode

```
bresenhamcircle(radius):
    x = 0, y = radius
    p = 3 - 2 * radius
    points = []

    while x <= y:
        add points (x,y), (y,x), (y,-x), (x,-y)
                   (-x,-y), (-y,-x), (-y,x) (-x,y)

        if p < 0:
            p = p + 4 * x + 6

        else:
            p = p + 4 * (x-y) + 10

            y = y - 1

        x = x + 1

    return points
```