



Important Instructions to examiners:

- 1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
- 2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
- 3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
- 4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
- 5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
- 6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
- 7) For programming language papers, credit may be given to any other program based on equivalent concept.

1. a) Attempt any six:

Marks 12

1) State syntax and use of getpixel.

(Syntax-1 Mark, Use-1 Mark)

Ans: Syntax: -

```
getpixel(int x, int y);
```

Where x, y gives pixel position on screen

Use:-

getpixel gets the color of a specified pixel;

getpixel gets the color of the pixel located at (x,y).

2) State various categories of line.

(All three categories-2 Marks)

Ans:

- i. Visible line
- ii. Invisible line
- iii. Partially Visible/Clipped line

3) Define fractal lines.

(Definition- 2 Marks)

Ans: Any object has either flat surfaces which can be drawn with the help of polygons or smooth curved surfaces which can be drawn by using curves. These objects are of rough, jagged, random edges. Some objects has a lot of specification which can be drawn with the help of fractals, where system draws jagged lines in given endpoints. Fractal uses geometry method, procedures to model objects.

**4) Write syntax of two graphic mode functions.**

(For each function-1 Mark, Any two functions are expected)

Ans:

- void initgraph (int *graphdriver, int graphmode, char *pathtodriver);
- void closegraph();
- void detectgraph(int *graphdriver, int graphmode);

5) Write homogeneous co-ordinate matrix for

- 2 D reflection

- 2D shear

(Reflection (Any one matrix) - 1 Mark, Shearing (any one Matrix)- 1 Mark for)

Ans:

i. 2D reflection

| | |
|---|--|
| About X Axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | About Y Axis $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |
| About Origin $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | |

ii. 2D Shear

| | |
|---------------|--|
| X Shearing | $\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |
| Y Shearing | $\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |



6) Define resolution.

(Definition- 2 Marks)

Ans: Total number of pixels displayed on the screen without overlapping.

OR

Resolution is defined as total number of pixels in X direction multiply with Total number of pixels in Y direction. It is denoted by

Resolution = Number of Pixel in x Direction X Number of pixels in y Direction.

7) What is beam penetration technique?

(For appropriate explanation-2 Marks)

Ans: It is used with random scan monitors. In this, normal CRT screen is coated with two phosphor layers. Red phosphor layer is deposited behind green phosphor layer. A slow electron beam excites only red phosphor layer and produces only red trace. A very fast beam penetrates through red layer and excites green layer also and gives green trace. A beam with intermediate speed gives red and green combination. The beam accelerating voltage controls the speed of beam and hence produces different screen colors.

8) Write two types of composite transformations.

(Any two types are expected- 1 Mark for each)

Ans: Following are type of composite transformation

- i. Translation + Scaling
- ii. Translation + Rotation
- iii. Rotation + Scaling
- iv. Translation + Reflection

**b) Attempt the following:****Marks 8****1) Compare properties of Bspline and Bezier curves.***(Each difference-1 Mark for, any four points are expected)***Ans:**

| Bezier curve | B-Spline Curve |
|---|--|
| Bezier Curve Offers only 4 control points | B-Spline offers more control points |
| Bezier Curve is less flexible | B-Spline Curve is more Flexible |
| Can represent simple curves | Cannot represent simple curves |
| High Degree of curves | Lower degree of curves |
| Simple to use and implement | Complex |
| Bezier is concave | B-spline can be concave and convex too |
| Extension is not possible | Extension is possible |

2) Explain syntax of flood-fill method.*(Flood fill syntax- 1 Mark, Explanation- 3 Marks)***Ans:** void floodfill(int x, int y, int border);

Floodfill function is used to fill an enclosed area. Current fill pattern and fill color is used to fill the area. (x, y) is any point on the screen if (x,y) lies inside the area then inside will be filled otherwise outside will be filled, border specifies the color of boundary of area. To change fill pattern and fill color use setfillstyle. Code given below draws a circle and then fills it.

Flood fill also called seed fill, is an algorithm that determines the area connected to a given node in a multi-dimensional array. It is used in the “bucket” fill tool of paint programs to determine which parts of bitmap to fill with color, and in puzzle games such as Puyo, Lumines, Magical Drop, and some implementations of Tetris (but not columns) for determining which pieces are cleared. The flood fill algorithm takes three parameters: a start node, a target color, and a replacement color. The algorithm looks for all nodes in the array which are connected to the start node by a path of the target color, and changes them to the replacement color. There are many



ways in which the flood-fill algorithm can be structured, but they all make use of a queue or stack data structure, explicitly or implicitly.

2. Attempt any four:**Marks 16****a) Explain steps for mid-point line clipping algorithm.**

(Steps/Algorithm-2 Marks, Diagram - 1 Mark, Explanation- 1 Mark)

Ans: Working of Mid Point Sub-Division Line Clipping Algorithm Midpoint sub-division line clipping algorithm find out the midpoint of a line segment and subdivide it into two equal parts. Then all possible cases are applied on line i.e. visible, invisible or partially visible. If line is visible then algorithm will display the line, if it is invisible then that segment of line is discarded. If line is partially visible then same sub-division process is continue till line segment become totally visible. This algorithm uses a binary search for the intersection point by always dividing the line at its mid point.

Step 1: Scan two end points for the line $P1(x1, y1)$ and $P2(x2, y2)$

Step 2: Scan corners for the window as $(Wx1, Wy1)$ and $(Wx2, Wy2)$

Step 3: Assign the region codes for endpoints P1 and P2 by initializing code with 0000.

Bit 1 - if $(x < Wx1)$ then bit 1 = 0 else bit 1 = 1

Bit 2 - if $(x > Wx2)$ then bit 2 = 0 else bit 2 = 1

Bit 3 - if $(y < Wy2)$ then bit 3 = 0 else bit 3 = 1

Bit 4 - if $(y > Wy1)$ then bit 4 = 0 else bit 4 = 1

Step 4: Check for visibility of line P1, P2

If region codes for both end points are zero then the line is visible, draw it and jump to step 6.

If region codes for end points are not zero and the logical Anding operation of them is also not zero then the line is invisible, reject it and jump to step 6.

If region codes for end points does not satisfies the condition in 4(i) and 4(ii) then line is partly visible.

Step 5: Divide the partially visible line segment in equal parts and repeat steps 3 through 5 for both subdivided line segments until you get completely visible and completely invisible line segments. Step 6: Exit

Formulas use in midpoint subdivision algorithm

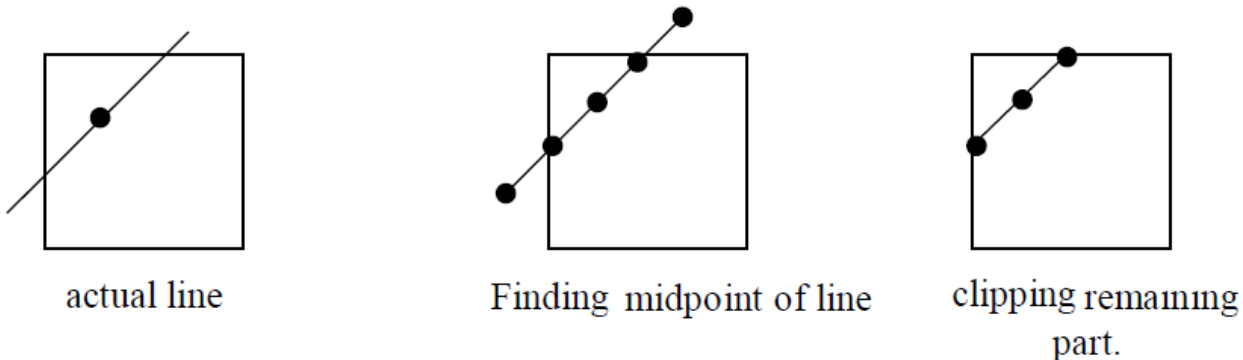
$X_m =$ To determine X value $\frac{X_2 + X_1}{2}$

2

Ym = To determine Y value. $\frac{Y_2 + Y_1}{2}$

2

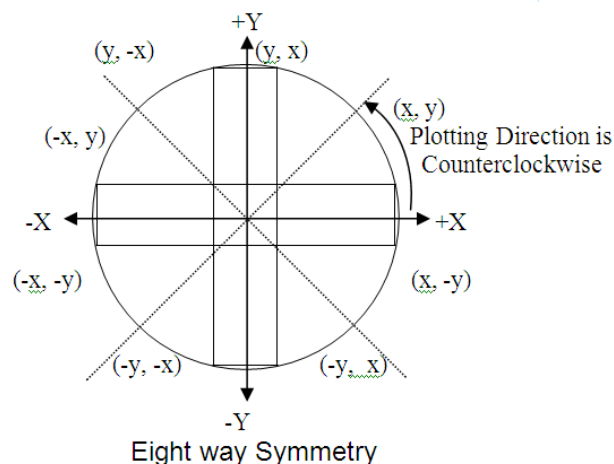
Line clipping operation in given window



b) Explain 8 – way symmetry concept of circle drawing.

(Explanation -3 Marks, Diagram: 1 Mark)

Ans: Eight way symmetry of a circle A circle is a symmetrical figure. It has eight-way symmetry as show in fig. below. Thus any circle drawing algorithm can have advantage of circle symmetry to plot eight points by calculating its co-ordinates of any one point. If any point is to be calculated in circle, seven other points can be drawn as reflection to it.



c) What are graphics standards? Explain any two graphics standards.

(Graphics Standard 2 Marks; Type 1 Mark for each type)

Ans: Graphics standards are provided for easy transfer of the graphics from one platform to another as there are many variations in display devices , software packages and even the graphical languages. Hence graphics standards are developed which provides the portability.



PHIGS: - PHIGS (Programmer's Hierarchical Interactive Graphics System) is an API standard for rendering 3D computer graphics, at one time considered to be the 3D graphics standard for the 1990s. Instead a combination of features and power led to the rise of OpenGL, which became the most popular professional 3D API of the 1990s. PHIGS is still widely used in the computer games and film industries. PHIGS was available as a standalone implementation (examples: Digital Equipment Corporation's DEC PHIGS, IBM's graPHIGS, Sun's SunPHIGS) and also used with the X Window system, supported via PEX, the "PHIGS Extension to X". PEX consisted of an extension to X, adding commands that would be forwarded from the X server to the PEX system for rendering. Workstations were placed in windows typically, but could also be forwarded to take over the whole screen, or to various printer-output devices.

PHIGS was designed in the 1980s, inheriting many of its ideas from the Graphical Kernel System of the late 1970s, and became an ANSI (ANSI X3.144-1988), FIPS (FIPS 153) and then ISO standard (ISO/IEC 9592 and ISO/IEC 9593) by 1989. Due to its early gestation, the standard supports only the most basic 3D graphics, including basic geometry and meshes, and only the basic Gouraud, "Dot", and Phong shading for rendering scenes. Features considered "standard" today, notably texture mapping, were not supported, nor were many machines of the era physically capable of it (at least in real-time).

IGES:- The Initial Graphics Exchange Specification (IGES) (pronounced *eye-jess*) is a file format which defines a vendor neutral data format that allows the digital exchange of information among Computer-aided design (CAD) systems. The official title of IGES is *Digital Representation for Communication of Product Definition Data*, first published in January, 1980 by the U.S. National Bureau of Standards as NBSIR 80-1978. Many documents (like early versions of the Defense Standards MIL-PRF-28000 and MIL-STD-1840) referred to it as ASME Y14.26M, the designation of the ANSI committee that approved IGES Version 1.0. Using IGES, a CAD user can exchange product data models in the form of diagrams, wireframe, freeform surface or solid modeling representations. Applications supported by IGES include traditional engineering drawings, models for analysis, and other manufacturing functions. An IGES file is composed of 80-character ASCII records, a record length derived from the punched card era. Text strings are represented in "Hollerith" format, the number of characters in the string, followed by the letter "H", followed by the text string, *e.g.*, "4HSL0T" (this is the text string format used in early versions of the FORTRAN language). Early IGES translators had problems with IBM mainframe computers because the mainframes used EBCDIC encoding for text, and some EBCDIC-ASCII



translators would either substitute the wrong character, or improperly set the Parity bit, causing a misread. Here is a very small IGES file from 1987, containing only two POINTS (Type 116), two CIRCULAR ARC (Type 100), and two LINE (Type 110) entities. It represents a slot, with the points at the centers of the two half-circles that form the ends of the slot, and the two lines that form the sides.

CGM:- The Computer Graphics Metafile (CGM) is the International Standard for storage and exchange of 2D graphical data. Although initially a vector format, it has been extended in 2 upwardly compatible extensions to include raster capabilities and provides a very useful format for combined raster and vector images. A metafile is a collection of elements. These elements may be the geometric components of the picture, such as polyline or polygon. They may be details of the appearance of these components, such as line color. They may be information to the interpreter about how to interpret a particular metafile or a particular picture. The CGM standard specifies which elements are allowed to occur in which positions in a metafile. CGM also has profile rules and a Model Profile to attempt to solve the problem of flavours of standards. 4 Internationally Standardised Profiles (ISPs) have been developed for CGM. CGM has been accepted as a MIME data type.

CORE:- The **3D Core Graphics System** (a.k.a. **Core**) was the very first graphical standard ever developed. A group of 25 experts of the ACM Special Interest Group SIGGRAPH developed this "conceptual framework". The specifications were published in 1977 and it became a foundation for many future developments in the field of computer graphics.

d) What is co-ordinate system? Explain polar co-ordinate system.

(Co-ordinate System- 2 Marks, Polar co-ordinate system- 2 Marks)

Ans: The idea of a coordinate system, or coordinate frame is pervasive in computer graphics. For example, it is usual to build a model in its own modeling frame, and later place this model into a scene in the world coordinate frame. It is common in the field of Computer Graphics to refer to the modeling frame as the object frame, and the world coordinate frame as the scene frame these terms are used interchangeably.

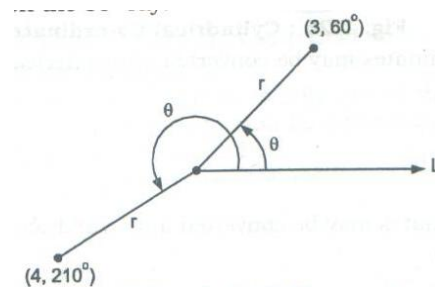
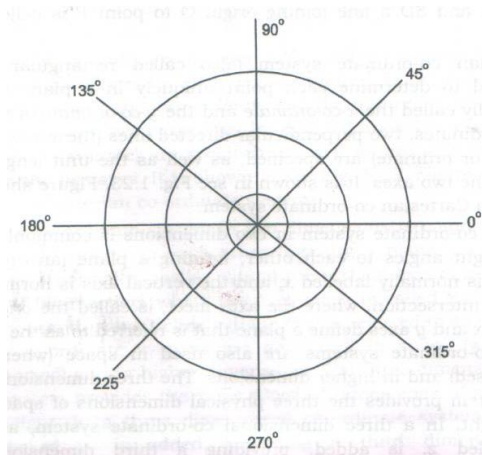
The polar Co- ordinate system, is a two –dimensional co-ordinate system in which each point on a plane is determined by an angle and a distance.

2D Cartesian (X,Y) co-ordinate corresponds to polar (r, Θ) co-ordinates I e point P is addressed by distance r of radius vector from origin and angle Θ which it makes in counter clockwise rotation from reference X axis.

Each point in the polar co-ordinate system can be described with the two parameters $\theta = \frac{\sin \theta}{\cos \theta}$

polar co-ordinates, which are usually called r (the radial co-ordinate) and θ (the angular co-ordinate, polar angle, sometimes represented as ψ or ϕ). The r co-ordinate represents the radial distance from the pole (pole is equivalent to the origin in the Cartesian system), and the θ co-ordinate represents the anticlockwise (counterclockwise) angle from the θ ray (sometimes called the polar axis), known as the positive X-axis on the Cartesian co-ordinate plane.

For example ,the polar co-ordinates (3 , 60°) would be plotted as a point 3 units from the pole on the 60° ray .



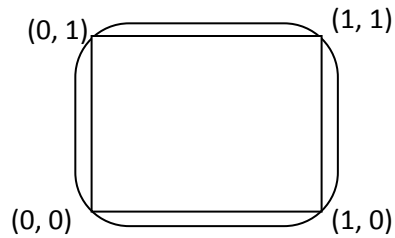
e) **Explain terms**

- **Normalization Co-ordinate**

- **World Co-ordinate**

(For Normalization co-ordinate System- 2 Marks, for World Co-ordinate- 2 Marks)

Ans: **Normalization Co-ordinate:** As different display devices have different screen size measured in pixels, we need to make our program display device independent . There for device independent co-ordinate system is used to display picture on screen. This device independent units system is called as “Normalized Device Co-ordinate System, the screen is measured as 1 unit in width and 1 unit height I.e 1*1 units. Lower left corner of screen is at the origin of co-ordinate system and upper right corner is the point (1, 1).The point (0.5, 0.5) is the center of the screen . It is shown in Fig.4.2.



Here some interpreter is used to convert these Normalized Device Co-ordinates to appropriate pixel values for respective display device .

Interpreter converts Normalized Device Co-ordinates to appropriate actual device co-ordinates (pixel values) by using following equation.

$$X = X_n + X_w$$

$$Y = Y_n + Y_H$$

X, Y = Actual device X and Y Co – ordinates

X_n, Y_n = Normalized device X and Y co-ordinates

Y_H = Height of actual screen in pixels..

World Co-ordinate:

Pictures are defined and stored into memory using any convenient Cartesian co-ordinate system is called as World Co-ordinate Systems (WCS).

When picture is to be displayed on display device it is measured in physical Device Co- ordinate System ((PDCS) corresponding to the display device .Therefore to display any image on screen , we need to map it from stored World Co- ordinate Systems to appropriate Physical Device Co- ordinate System (PDCS).

Therefore in general the process of mapping of picture from World Co- ordinate Systems to Physical Device Co- ordinate System is referred as “viewing Transformation”. Two dimensional viewing transformation is also referred as “Windowing Transformation”.

Viewing Transformation consist of two parts as

- 1) Normalization Transformation
- 2) Workstation Transformation

Where,

- (1) Normalisation Transformation : maps World Co-ordinate Systems to Normalized Device Co- ordinate Systems (NDCS).

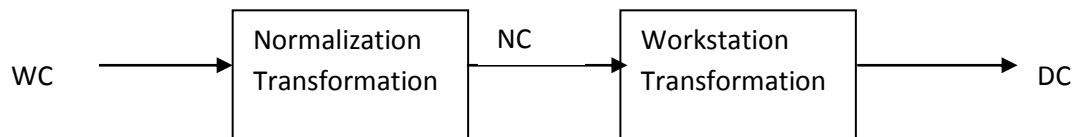


(2) Workstation Transformation maps this Normalize Device Co-ordinate Systems to physical Device Co- ordinate Systems

Note: Both above parts I.e Normalization Transformation

And Workstation Transformation are also referred as window –to –view port Transformation.

Pictorially Viewing Transformation is shown in following Fig . 4.1.



f) Translate a triangle ABC by 3 units in Y direction having co-ordinates A(5,5) B(10,5) and C(10,10). Find Co-ordinate after transformation.

(Computation of each point -1 Mark, showing all points- 1 Mark)

Ans: As a solution we shall find out new co –ordinates of triangle ABC.

We know equation for Translation ;

$$X' = X + t_x$$

$$Y' = Y + t_y$$

Let A'B'C' be the new coordinates of the triangle.

$$A'B'C' = T_x * ABC$$

Hence given $t_x = 0$, $t_y = 3$

$$\text{Formula} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}; \text{ putting } T_x = 0 \text{ and } T_y = 3 \text{ we will get;}$$

$$A'B'C' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 & 10 & 10 \\ 5 & 5 & 10 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 10 & 10 \\ 8 & 8 & 13 \\ 1 & 1 & 1 \end{bmatrix}$$

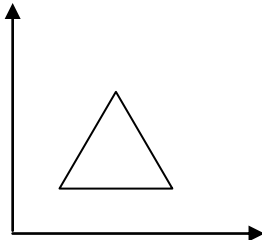
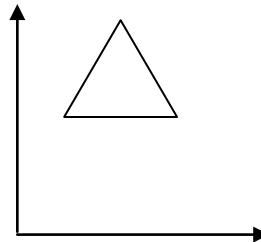
So the final co-ordinates for Triangle ABC will be:

$$A' = (X', Y') = (5, 8)$$

$$B' = (X', Y') = (10, 8)$$

$$C' = (X', Y') = (10, 13)$$

Output:-

**Original Object****Translated Object****3. Attempt any four:****Marks 16****a) Explain use and structure of display file.***(Use -2 Marks, Structure of Display file- 2 Marks)*

Ans: Use: Display file contains series of commands required to draw a required picture. It stores intensity of picture pattern.

Display File Structure commands are made up of two parts as:

- i. Operation code (OP-CODE)
- ii. Operands

Op code identifies what type of command it is and operands provides co-ordinate of point (x, y) to process command. Each command needs three arrays to store itself in display file. First array stores op code, second array stores x co-ordinate and third array stores y co-ordinate of a point. For example, to access a third command from display file third route from all three arrays must be accessed. Before starting to write a series of commands, we shall assign some op codes to commands. Here, we consider commands we shall assign some op codes to commands. Here, we consider only basic commands as LINE and MOVE. Let's define op code 1 for MOVE and 2 for LINE.

For example, to access a third command from display file third route from all three arrays must be accessed –

DF-OP[3], DF-X[3] and DF-Y[3]. We considered only the basic commands as LINE and MOVE.

i.e. DF-OP [] = 1 means MOVE

DF-OP [] = 2 means LINE

**b) Explain various methods of entering a polygon into the display file.**

(For Explanation- 2 Marks, Algorithm- 2 Marks)

Ans: If system supports polygon drawing primitives, it defines a set of different commands to draw polygon for e.g. triangle(), rectangle(), etc. Following command enters and draw rectangle on graphics device.

Rectangle (100, 100, 150, 50);

It consists of three columns where first column is opcode number, and second is X coordinate and third is Y coordinate. These three columns can be treated as a three different ways. Following is a stepwise procedure to enter polygon

Step1: Enter N as number of polygon edges.

Step2: Enter X & Y coordinates of vertices of polygon in array Ax & Ay respectively.

Step3: Initialize counter i=0

op [i] = N

x [i] = Ax [i]

y [i] = Ay [i]

i= i+1

It loads first row of display file i.e. polygon command with start point.

Step4:

op [i] = 2

x [i] = Ax [i]

y [i] = Ay [i]

i= i+1

Step5: Repeat step 4 'N – 1' times where step 4 enters a line to command

Step6:

op [i] = 2

x [i] = Ax [0]

y [i] = Ay [0]

This step enters last line command with start point as in point so to close polygon

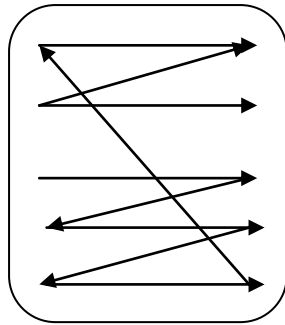
Step7: Stop



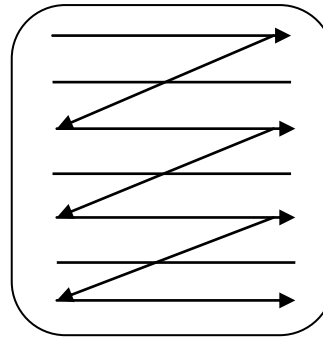
c) Describe Raster scan display methods with diagram.

(Diagram 1 Mark, Description- 3 Marks)

Ans:



Non – Interlace



Interlace

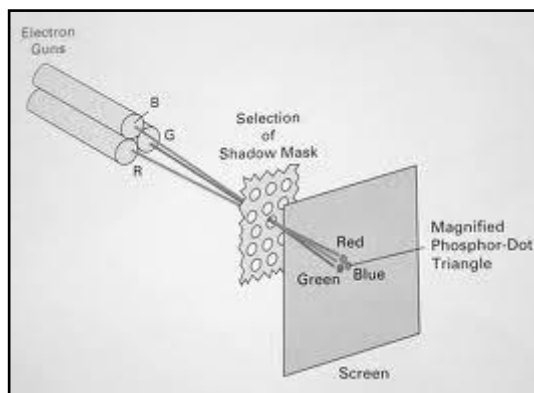
In this electron beam moves all over the screen one row at a time.

As electron beam moves across each row, beam intensity is turned ON and OFF to create picture pattern. Here, picture definition is stored in memory called as refresh buffer or frame buffer. This refresh buffer holds set of intensity values for all screen points. These stored intensity values are re-tried from buffer and displayed on screen one row at a time. When beam is moved from left to right, it is ON and when moved from right to left it is OFF and is termed as horizontal retrace when beam reaches bottom of screen it is made OFF and it return to top left corner of screen which is referred as vertical retrace. Raster scan may use non-interlace or interlace technique to display picture. In non-interlace, electron beam scans through each line one by one. In interlace technique, beam firstly scans through even scan lines and then odd scan lines thus scanning screen twice.

d) Explain how coloured pixel is displayed by shadow mask CRT.

(For explanation- 4 Marks, Diagram is optional)

Ans: Shadow Mask Technique:



It can be display wider range of colours than beam penetration technique. In this type of CRT a metal plate having small rounds holes in triangular pattern called shadow mask is inserted behind phosphor layer. Electron gun consists of three electron guns group in triangle for e.g. red, green & blue. These three beams encounter hole in mask passed through and strike the phosphor since they originate at three different points they strike the phosphor in three slightly different spots as red, blue, green.

These three spots placed which very closed to each other are activated a single phosphors spot is made visible on screen. By varying the intensity of three electron beams, we can retain different colors.

e) Explain flood fill algorithm and its two ways to fill the polygon.

(Explanation: 2 Marks for each type: 1 Mark)

Ans: Flood fill algorithms starts from any speed point inside polygon & set it to fill color. In Flood Fill method user starts from any point inside polygon & fills the color to the neighboring pixel till polygon boundary is reached. This method has one limitation that is the polygon has any point of different color then this method stops coloring the neighboring pixel.

Flood fill algorithm

Step 1: Read any seed pixel position (x, y)



Step 2: Check to see if this pixel (x, y) has old interior color. If old color, then set it to new fillcolor.

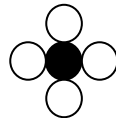
Step 3: Repeat step 2 for neighboring pixels of (x, y)

Step 4: If pixel (x, y) does not has old interior color then stop. (Getpixel, Putpixel and Flood fill statement can be used to increment the neighboring pixel).

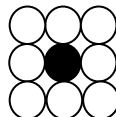
It has 2 ways to fill polygon:

1. 4-connected method
2. 8-connected method

4-connected method: In this 4 neighboring points of current test point are tested. These are pixel position that are right, left, above & below of current pixel as shown in fig.



8-connected method: In this 8 neighboring points of current test point are tested. These are pixel position that are right, left, above & below and 4-diagonal position of current pixel as shown in fig.



f) Write the need of homogeneous co-ordinate in transformations.

(Any relevant explanation shall be considered- 4 Marks)

Ans: Construction and design of sequence transformations involve transformations like scaling, rotation, reflection, and translation. The basic transformation which can be expressed in general form.

$$[P'] = [P] * [T_1] + [T_2]$$

For translation one can have;

$$[P'] = [P]. \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + [tx \quad ty]$$

Same procedure can be followed for scaling and rotation. In order to produce a sequence of transformation, including above equations, one must compute transformed coordinates one at a time. This sequential transformation is proved to be inefficient. A more general approach is

overcome these intermediate computations is required that gives the final coordinates directly from initial coordinates.

The objective can be achieved by introducing homogeneous coordinate representation, which is a technique based on projective geometry. The coordinate representation of a point $[x \ y]$ in the homogeneous representation is given by a triplet $[X_h \ Y_h \ h]$, where

$$\underline{X_h} \quad \text{and} \quad \underline{Y_h}$$

For 2D transformation system, the homogeneous parameter h can be any nonzero values. For simplicity and convenience, h is taken to be 1 which represents a point as $[X_h \ Y_h \ 1]$. The transformation matrix is now represented by order 3×3 instead of 2×2

4. Attempt any two:

Marks 16

a) Explain use of 4-bit region code in case of line clipping algo.

(Explanation- 4 Marks, Region code- 2 Marks, Line clipping diagram- 2 Marks)

Ans: Different cases which are to be consider before clipping line are,

- i) Completely inside.
- ii) Completely outside.
- iii) Partly inside.

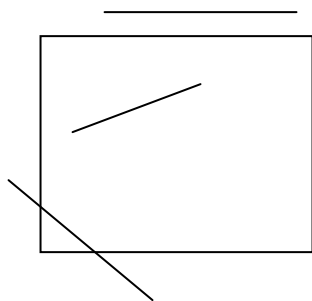


Fig (a) Before clipping

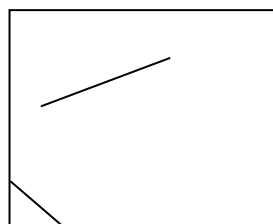


Fig (b) After clipping

As shown in fig three lines are given in which one line is inside window so that is displayed, second line is completely outside so it is discarded after clipping, and third line is clipped as it is partially in so that much part is displayed and remaining part is discarded.

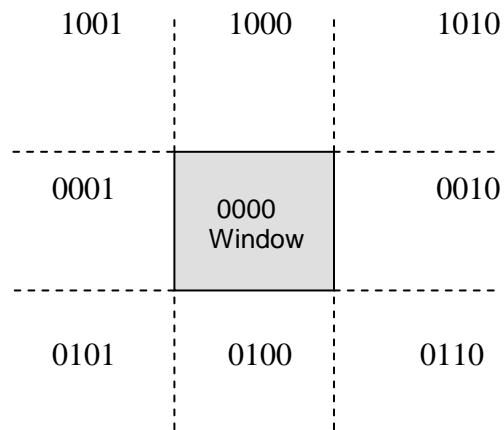


Fig Four bit codes for nine regions

Line clipping in given window:

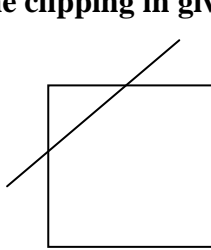


Fig (a) actual line

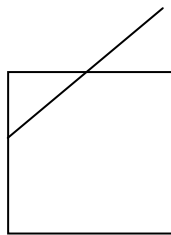


Fig (b) clipping
one part of line

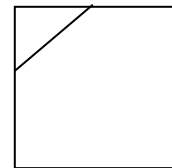


Fig (c) clipping
remaining part.

Assign the region codes for endpoints P1 and P2 by

Bit 1 - if ($x < Wx1$)

Bit 2 - if ($x < Wx2$)

Bit 3 - if ($x < Wy2$)

Bit 4 - if ($x < Wy1$)

Check for visibility of line P1, P2

- If region codes for both end points are zero then the line is visible, draw it and jump to step 9.
- If region codes for end points are not zero and the logical and operation of them is also not zero then the line is invisible, reject it and jump to step 9.
- If region codes for end points does not satisfies the condition in 4(i) and 4(ii) then line is partly visible.



| Line | Code for End point | | Logical And Operation | Result |
|--------|--------------------|------|-----------------------|----------------------|
| P1, P2 | 0000 | 0000 | 0000 | Completely Visible |
| P3,P4 | 0001 | 0001 | 0001 | Completely invisible |
| P5,P6 | 0001 | 0000 | 0000 | Partly Visible |
| P7,P8 | 0100 | 0010 | 0000 | Partly Visible |
| P9,P10 | 1000 | 0010 | 0000 | Partly Visible |

b) Explain 3 D transformation matrix for translation, scaling and rotation.

(For Translation- 2 Marks, For Scaling- 3 Marks, for Rotation-3 Marks)

Ans: Translation: -

In Euclidean geometry, a translation refers to moving every point at a constant distance in a specified direction. It is one of the rigid motions. A translation can also be interpreted as an addition of a constant vector to every point or shifting the origin of the coordinate system. If v is a fixed vector, then the translation T_v will work as $T_v(P) = P + v$

Translation is an affine transformation but not a linear transformation. Homogeneous coordinates are normally used to represent the translation operator by a matrix. Thus we write a 3D vector $w = (W_x \ W_y \ W_z)$ using four homogeneous coordinates co-ordinates as $w = (W_x \ W_y \ W_z \ 1)$

To translate an object by a vector $v = [T_x \ T_y \ T_z]$, each homogeneous vector $P = [x \ y \ z \ 1]$ would need to be multiplied by this translation matrix.

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

$$[T * P] = [X \ Y \ Z \ 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

$$= [x + T_x \ y + T_y \ z + T_z \ 1]$$

$$= P + v.$$

For Example:-

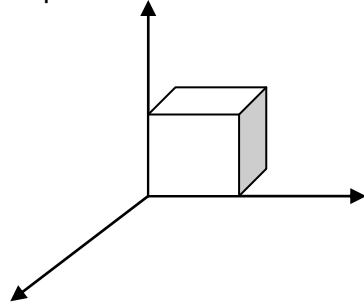


Fig (a) Before Translation

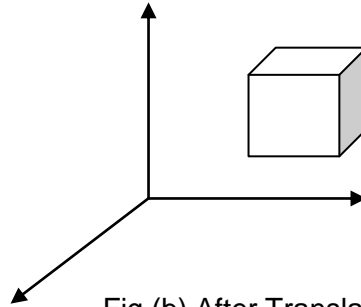


Fig (b) After Translation

Scaling Transformation:-

A scaling can be represented by a scaling matrix. To scale an object by a vector $V = [S_x, S_y, S_z]$, each point $P = [x \ y \ z]$ would need to be multiplied with this scaling matrix.

$$S_v = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$$

$$P * S_v = [X \ Y \ Z] * \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} = [S_x X \ S_y Y \ S_z Z]$$

Such a scaling changes the diameter of an object by a factor between the scale factors, the area by a factor between the smallest and the largest product of two scale factors, and the volume by the product of all three.

In homogeneous coordinates, since translation cannot be accomplished with a 3 X 3 matrix. To scale an object by a vector $V = [S_x \ S_y \ S_z]$ each homogeneous vector $P = [X \ Y \ Z \ 1]$ would need to be multiplied with the scaling matrix.



$$S_v = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P * S_v = [X \ Y \ Z \ 1] * \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [S_x X \ S_y Y \ S_z Z \ 1]$$

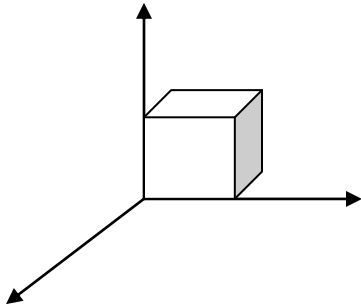


Fig (a) Before Scaling

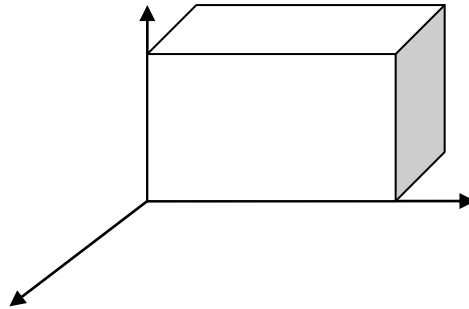


Fig (b) After Scaling

Rotation: In 3D rotation all transformations are carried out around the line in a space therefor to generate rotation transformation of an object should be designate and access of rotation about which object is to be rotated and amount of angular rotation. Equation for rotation is

$$P' = R \cdot P$$

Where R is transformation matrix for 3D rotation with homogeneous coordinates

1. Matrix for rotation about x-axis = $R_x(\theta)$

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

2. Matrix for rotation about y-axis = $R_y(\theta)$



$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Matrix for rotation about z-axis = $R_z(\theta)$

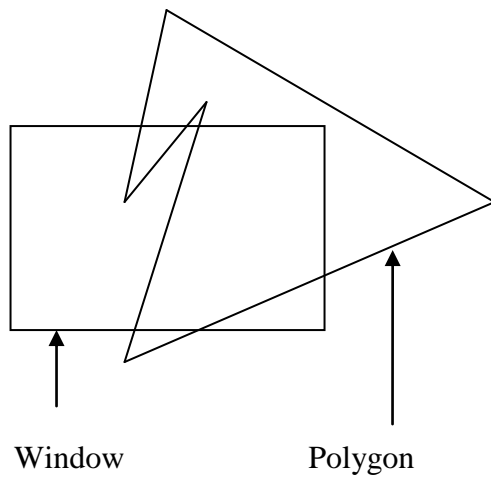
$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

c) Describe steps of polygon clipping and state its merits and demerits.

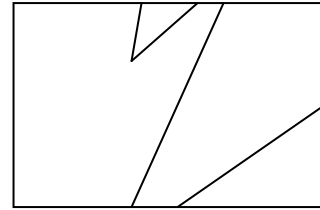
(For algorithm/steps- 6 Marks, for merit- 1 Mark, for demerit- 1 Mark)

Ans: Polygon Clipping:

The process which divides each elements of picture into its visible and invisible position, allowing invisible portion to be discarded is called as clipping. It a Polygon is process for line clipping using line clipping algorithm ,it may display series of unconnected line segments as shown in a Figure.



(a) Before Clipping



(b) After Clipping

Polygon clipping algorithm shows a steps

Sutherland-Hodgeman polygon clipping algorithm

Step 1: Read co-ordinates of all vertices of the polygon.

Step 2: Read co-ordinates of the clipping window.

Step 3: Consider the left edge of window.

Step 4: Compare vertices of each of polygon, individually with the clipping plane.

Step 5: Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary.

Step 6: Repeat the steps 4 and 5 for remaining edges of clipping window. Each time resultant list of vertices is successively passed to process next edge of clipping window.

Step 7: Stop

Merit: - any desired shape/object can be achieved.

Any small part of an object can be selected.

Demerit:-

Complexity for algorithm

Complex shape polygon clipping is not easily achieved.



5. Attempt any four: Marks 16

a) Write C code for line by bresenham, if line has slope less than one.

(Initiation of graphics and variables- 2 Marks, Equations- 2 Marks)

Ans: #include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"e:\\tc\\bgi");
int dx,dy,x,y,p,x1,y1,x2,y2;
clrscr();
printf("\n\n\tEnter the co-ordinates of first point : ");
scanf("%d %d",&x1,&y1);
printf("\n\n\tEnter the co-ordinates of second point : ");
scanf("%d %d",&x2,&y2);
dx = (x2 - x1);
dy = (y2 - y1);
p = 2 * (dy) - (dx);
if(x1<x2)
{
x=x1;
y=y1;
Xend=x2;
}
else
{
x=x2;
y=y2;
Xend=x1;
}
}



```
for(x=x1;x<=xend;x++)
{
    putpixel(x,y,WHITE);
    if(p <= 0)
    {
        x=x+1;
        y=y;
        p = p + 2 * (dy);
    }
    else
    {
        x=x+1;
        y=y+1;
        p = p + 2 * (dy - dx);
    }
}
getch();
closegraph();
}
```

b) Write four graphics hazards.

(Four points - 1 Mark each)

Ans:

1. The device independent property is not met by every application. The given application may not run equally well on wide variety of devices.
2. Improper implementations or poor definition of standard may lead to create problems for users, who know nothing of the standard.
3. Standards once defined may create difficulties in accommodating new needs. To avoid this problem to extent standards have to be made flexible enough to accommodate new needs and conditions occurring due to changing environment.



-
4. The methods embodied in standards may be viewed as the only suitable method for solving problems. This may lead to misunderstanding or even misuse of the standard. Thus suitability of particular standard for given application must be analysed carefully.

c) Describe DDA Arc generation algo.

(Algorithm- 3 Marks and explanation- 1 Mark)

Ans: DDA algorithm calculates coordinates of the points on the curve. The differential equations for the arc coordinates can be written in terms of angle parameter A as follows

1. $x = R \cos A + x_0$

2. $y = R \sin A + y_0$

After differentiating the equation becomes

$$dx = -R \sin A dA$$

$$dy = R \cos A dA$$

$$R \cos A = x - x_0 \text{ \& } R \sin A = y - y_0$$

Substituting these values

$$dx = -(y - y_0) dA$$

$$dy = (x - x_0) dA$$

Now changes in dx and dy gives the amount to add an old point on the arc to get new point

$$x_2 = x_1 + dx = x_1 - (y - y_0) dA$$

$$y_2 = y_1 + dy = (x - x_0) dA$$

Algorithm:-

Step 1

Read the center of a curvature say (x_0, y_0)

Step 2

Read the arc angle say Θ

Step 3

Read the starting point of the arc i.e. (x, y)

Step 4

Calculate $d\Theta$

$$d\Theta = \text{Min} (0.01, 1/(3.2 \times (|x - x_0| + |y - y_0|)))$$

Step 5

Initialise Angle = 0

Step 6



While (Angle < Θ)

Do

{ Plot (x, y)

$X = x - (y - y_0) \times d\Theta$

$Y = y + (x - x_0) \times d\Theta$

Angle = Angle + d Θ

}

Step 7

Stop

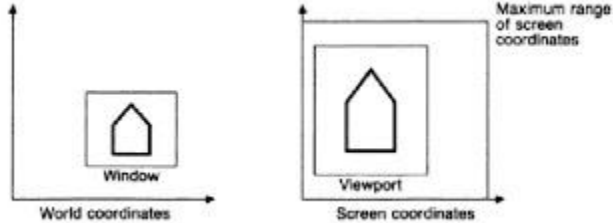
d) Explain any four design rules of GUI.

(Each principle-1 Mark (any four))

Ans: The user must be able to anticipate visual control's behavior from its visual property.

- The user must be able to anticipate behavior of your program using knowledge gained from other program.
- View every user warning and error dialog that your program generates as an opportunity to improve your interface.
- Provide adequate user feedback.
- Use sound, color, animation, and multimedia clip sparingly.
- Help users to customize and preserve their preferred work environment.
- Design your interface so that users can do their task while being minimally aware of the interface itself.

**e) Compare Window and Viewport.***(Any four points- 1 Mark each)***Ans:**

| window | Viewport |
|---|--|
| 1. Window is defined in world co-ordinate system. | 1. Viewport is defined in device co-ordinate system |
| 2. Window defines what is to be viewed. | 2. Viewport defines where it is to be viewed. |
| 3. When the window is moved, the display picture changes. | 3. When the viewport is moved, the display picture remains same only the area of display changes. |
| 4. Window can be defined with GWINDOW statement. | 4. Viewport can be defined using VPORT command. |
| 5. The window can be defined to select part of the picture or whole picture to display. | 5. The viewport can be defined to select the part of the device to display the picture or the whole device to display picture.  |

f) Rasterise a DDA line for co-ordinate(5,5) and (12,8).*(Initialization of values- 1 Mark, drawing table- 2 Marks, Diagram- 1 Mark)***Ans:** $X_1=5, y_1=5, x_2=12, y_2=8$

$$1. \quad Dx = x_2 - x_1 = 12 - 5 = 7$$

$$Dy = y_2 - y_1 = 8 - 5 = 3$$

$$2. \quad \text{If } (dx \geq dy)$$

$$\text{Length} = dx$$

Else

$$\text{Length} = dy$$

$$\text{So length} = 7$$

$$X_{\text{inc}} = dx / \text{length} = 7/7 = 1$$



$$Y_{inc} = dy / \text{length} = 3/7 = 0.4$$

3. For(i=0; i<=length; i++)

Do

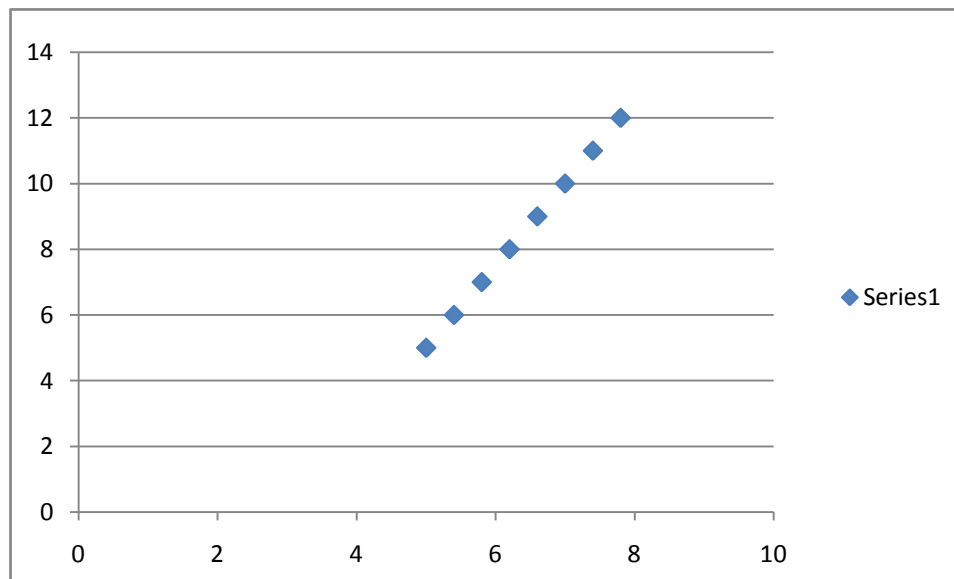
{

x=x+xinc;

Y=y+yinc;

}

| I | x | Y |
|---|----|-----|
| 0 | 5 | 5 |
| 1 | 6 | 5.4 |
| 2 | 7 | 5.8 |
| 3 | 8 | 6.2 |
| 4 | 9 | 6.6 |
| 5 | 10 | 7.0 |
| 6 | 11 | 7.4 |
| 7 | 12 | 7.8 |





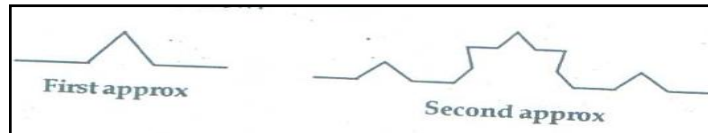
6. Attempt any four:

Marks 16

a) Explain Koch curve till 2nd appx. and state its applications.

(Koch curve explanation with diagram- 3 Marks and applications- 1 Mark)

Ans:



To geometrically construct a deterministic (nonrandom) self-similar fractal, we start with a given geometric shape, called the *initiator*. Subparts of the initiator are then replaced with a pattern, called the generator.

Each straight-line segment in the initiator is replaced with four equal-length line segments at each step. The scaling factor is $1/3$, so the fractal dimension is $D = \ln 4 / \ln 3 \approx 1.2619$. Also, the length of each line segment in the initiator increases by a factor of $4/3$ at each step, so that the length of the fractal curve tends to infinity as more detail is added to the curve.

Application of Koch curve:

- We can model various natural objects that have compound parts, such as island distributions along coastlines.
- Tree object and mountain object can be drawn.

b) Describe clipping concept with eg.

(Consider any 1 type of clipping with example Clipping definition- 1 Mark, Explanation- 2 Mark, Example- 1 Mark)

Ans:

Clipping is the process of dividing image into two parts visible and invisible and discard the invisible part from the image.

Point clipping:

If these conditions satisfy the point is inside otherwise the point is outside.

$X_{min} \leq x \leq x_{max}$

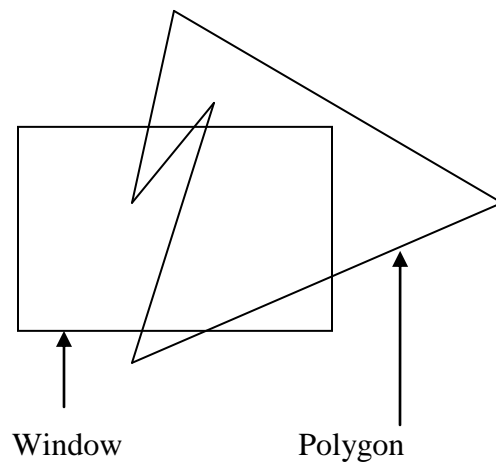
$Y_{min} \leq y \leq y_{max}$

Line clipping:

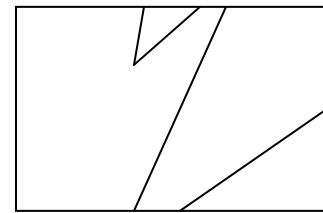
1. End-points pairs are check for trivial acceptance or trivial rejected using the outcode.
2. If not trivial-acceptance or trivial-rejected, divided into two segments at a clip edge.
3. Iteratively clipped by testing trivial-acceptance or trivial-rejected, and divided into two segments until completely inside or trivial-rejected

Polygon Clipping:

The process which divides each elements of picture into its visible and invisible position, allowing invisible portion to be discarded is called as clipping. It a Polygon is process for line clipping using line clipping algorithm ,it may display series of unconnected line segments as shown in a Figure.

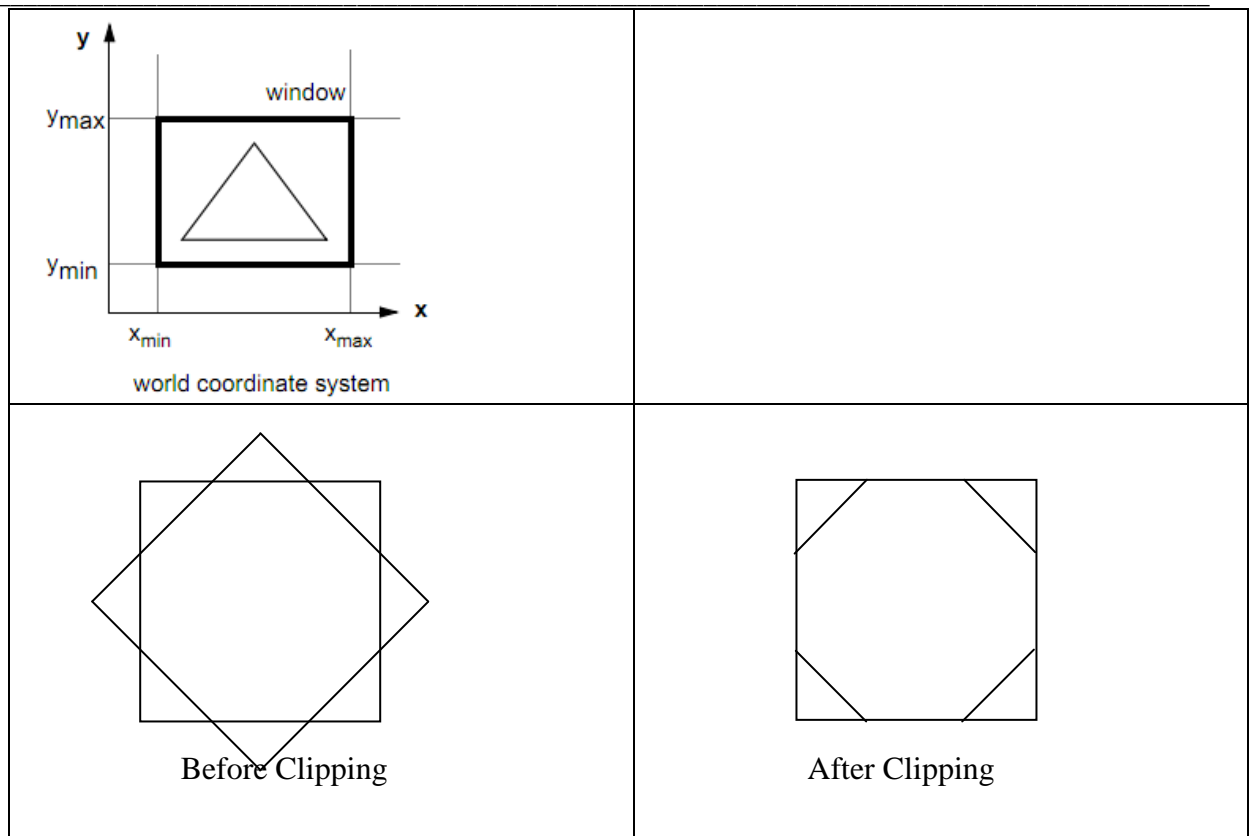


(a) Before Clipping



(b) After Clipping

Example:



c) **Explain four text mode functions.**

(Any four functions each- 1 Mark)

Ans :

1. **Window()**

This function takes four integer arguments that determine the left, top, right, and bottom co-ordinates. These co-ordinates essentially refer to rows and columns, which should be absolute values.

syntax: window(30,5,5,10);

2. **cputs()**

This function can be used to write a string in a window defined using window function.

Syntax

is

cputs(string);

3. **gotoxy()**

This function positions the cursor within the a text window . This function takes two integer parameters x and y co ordinates where the cursor should move. Syntax is:

gotoxy(column,row)

4. **outtext** function displays text at current position Declaration :- void outtext(char *string);



5. **outtextxy** function displays text at current position. Declaration :- void outtextxy(x,y,char *string);

6. **puts()**

This function can be used to write a string in a window defined using window function.

Syntax is

puts(string);

7. **gets()**

This function can be used to write a string in a window defined using window function.

Syntax is

gets(string);

d) **Write equation to find intersection of a line with left, right, top and bottom edge of a window.**

(Each edge equation- 1 Mark)

Ans: Formulas for clipping w.r.t. edge, in cases of:

Top edge: $x = x_0 + (x_1 - x_0) * (y_{max} - y_0) / (y_1 - y_0)$

Bottom edge: $x = x_0 + (x_1 - x_0) * (y_{min} - y_0) / (y_1 - y_0)$

Right edge: $y = y_0 + (y - y_0) * (x_{max} - x_0) / (x_1 - x_0)$

Left edge: $y = y_0 + (y - y_0) * (x_{min} - x_0) / (x_1 - x_0)$

e) **Write C code to display rectangle by functions.**

(Program - 4 Marks)

Ans: #include<stdio.h>

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\TC\\BGI");

rectangle(100,100,200,200);

getch();

closegraph();

```
return 0;
```

```
}
```

f) Draw and label DVST.

(Diagram - 2 Mark, Labeling- 2 Mark)

Ans:

