Biyani's Think Tank

Concept based notes

Computer Graphics

(MCA)

Gajendra Sharma
Assistant Professor
Department of IT
Biyani Girls College, Jaipur

Published by :

Think Tanks
Biyani Group of Colleges

Concept & Copyright :

©Biyani Shikshan Samiti
Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

Edition : 2012

Leaser Type Setted by :

Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director* (*Acad.*) Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this endeavour. They played an active role in coordinating the various stages of this endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Gajendra Sharma

Syllabus

MCA

Computer Graphics

Introduction: Elements of graphics workstation. Video Display Devices. Raster Scan Systems. Random Scan systems. Input devices. Graphics Software Coordinate Representations, Fundamental Problems in Geometry.

Algorithms: Line drawing algorithms- DDA Algorithm. Bresenham's Line Algorithm. Frame buffers. Circle and Eclipse generating algorithms. Midpoint Circle Algorithm. Sean-line polygon fill algorithm. Inside-Outside tests. Sean- Line fill of curved Boundary Areas. Boundary fill Algorithm. Flood fill Algorithm. Character generation. Attributes of lines, curves, filling, characters. etc.

Graphics Primitives: Primitive Operations, The display file interpreter-Normalized Device Coordinates. Display- File structure. Display – file algorithm. Display control and Polygonspolygon representation.

Attributes of output primitives: Line attributes - Line type. Line width. Pen and Brush options. Line Color. Color and gray scale levels. Color-tables. Gray scale. Area- Fill Attributes- Fill styles. Pattern fill. Soft fill. Character Attributes. Text attributes.

Geometric Transformations: Matrices. Scaling Transformations. Sin and Cos Rotation. Homogeneous Co-ordinates and Translation. Co-ordinate Translations. Rotation about an arbitrary point. Inverse Transformations, Transformations Routines.

2-D Viewing- The viewing pipeline. Viewing co-ordinate, Reference Frame. Windows to view ports . co-ordinate transformation 2-D Viewing functions. Clipping operations point clipping.Line clipping. Cohen- Sutherland. Line Clipping. Polygon clipping. Sutherland Hodge man clipping.

3-D concepts. Three dimensional Display Methods Parallel projection. Perspective projection. Visible line and surface identification. Surface rendering. Three Dimensional Object representations. Bezier curves and surfaces. B-Spline curves and surfaces.

Visibility , Image and object precision Z- buffer algorithm. Floating horizons. Computer Animation: Design of Animation Sequences. General Computer Animation Functions-Raster Animations. Key Frame Systems. Morphing Simulating Accelerations. Motion Specifications. Kinematics and Dynamics.

Chapter 1

Q1. Explain Video Display Devices.
Or
Explain how does the Cathode ray tube works with Detail.

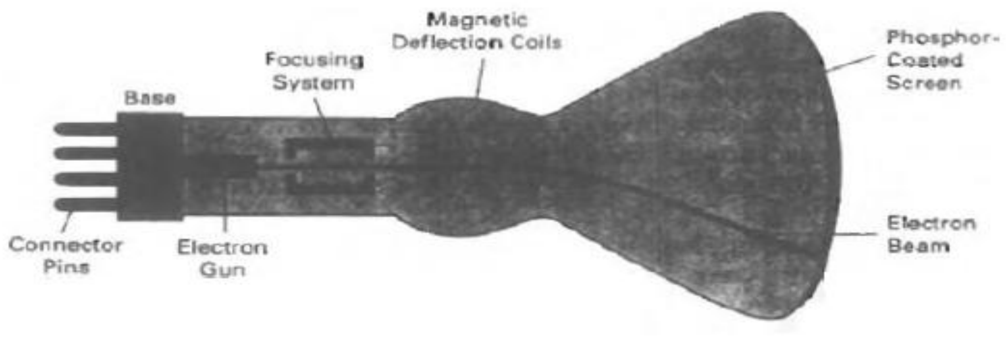
Ans

The Primary output device in Computer Graphics is a Monitor which operates on the standard cathode-ray tube(CRT) design and a few more technological hardware have also come into the concept.

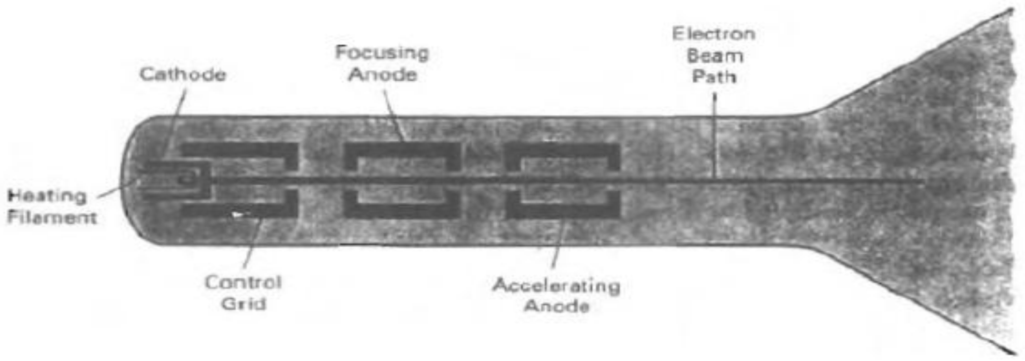
Computer graphics is a complex and diversified technology.

Refresh Cathode-Ray Tubes

The following figure illustrates the basic operation of how does a CRT work. An electron beam comes from the electron gun, passes through focus and deflection systems that send the beam towards directed positions on the phosphor-coated screen. The phosphor in return emits a small spot of light at each position where ever the electron beam makes contact. As the light which is emitted by the phosphor fades very easily, some mechanism is required for managing the picture on the screen. One method to make the phosphor glowing is to keep on redrawing the picture in a repeated manner by quickly projecting the electron beam over the same points again and again.

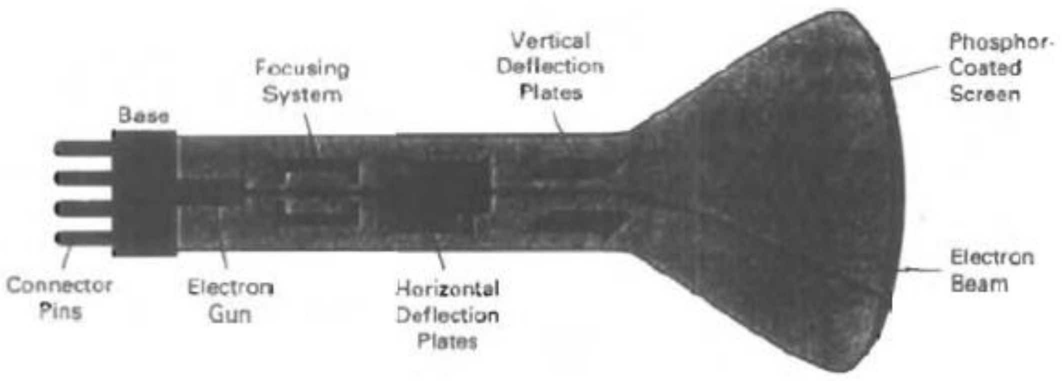


The basic component of an electron gun in a CRT



The heated metal cathode and a control grid are the key components of an electron gun in a CRT. Through the coil of wire, called the filament, inside the cylindrical cathode structure, heat is supplied to the cathode by directing a current which makes electrons to be 'boiled off" the hot cathode surface. The free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage, in the vacuum inside the CRT envelope. The accelerating voltage can be generated with a positively charged metal coating on the inside of the CRT envelope near the phosphor screen, or an accelerating anode can be used, as in Figure**.** Most of the times the electron gun is meant to contain the accelerating anode and focusing system within the same unit. Intensity of the electron beam is maintained by keeping voltage levels on the control grid, which is a metallic cylinder and that fits over the shape of the cathode. A high negative voltage applied to the control grid shuts off the beam as it repels electrons and stops them from passing through the small hole at the end of the control grid structure. A smaller negative voltage on the control grid alltogetherly decreases the total number of electrons passing through it. Since the amount of light emitted by the phosphor coating depends on the number of electrons which strike the screen, the brightness of a display can be controlled by changing the voltage on the control grid. In the electron beam, electrons spread all over the screen as a result of repulsion among them. To make the electron beam strike at one point, focusing anode is present in the CRT. Hence our focusing mechanism makes the electron beam to strike the phosphor screen at a small spot and focusing is following by usage of magnetic and electric field. Magnetic deflected is carried out by using 2 pairs of magnetic coils within the CRT. One pair is on the top and down position and the other one pair is on the opposite sides of CRT as it is shown in the Figure Magnetics field thus produced by each pair creates a transverse deflection force, perpendicular to the way of magnetic field and to the direction in which the electron bean is travelling. Moreover Horizontal deflection of electron bean is accomplished by one pair of coils and vertical deflection is carried out by the others.

Electric deflection is carried out by using two pairs of deflecting plates inside CRT, the two pairs are mounted vertically and horizontally.



Horizontal deflecting plates provide vertical deflection to the electron beam and vertical deflecting plates provide horizontal deflection to the electron beam. Important terminologies in CRT are as follows:

Dot Pitch: It denotes the distance which marks separation between two phosphor dots of the same color. A dot pitch equal to or less than .25mm is of use comfortably, whereas we should avoid monitors with a dot pitch equal to or greater than 0.28 mm.

Refresh rate: It denotes the number of images which are displayed every second, or we can say that it is the number of times the images is remapped per second. And It is also known as vertical scan rate and is expressed in Hertz (Hz).

Resolution : It denotes the number of pixels per surface unit and can be abbreviated as DPI or dots per inches and is calculated both vertically and horizontally. A resolution of 200dpi means that 200 columns and 200 rows of pixels per square

Size : It is calculated by taking the dimension of the diagonal of the screen and is expressed

Aspect Ratio : It is termed as the ratio of vertical points to horizontal points.

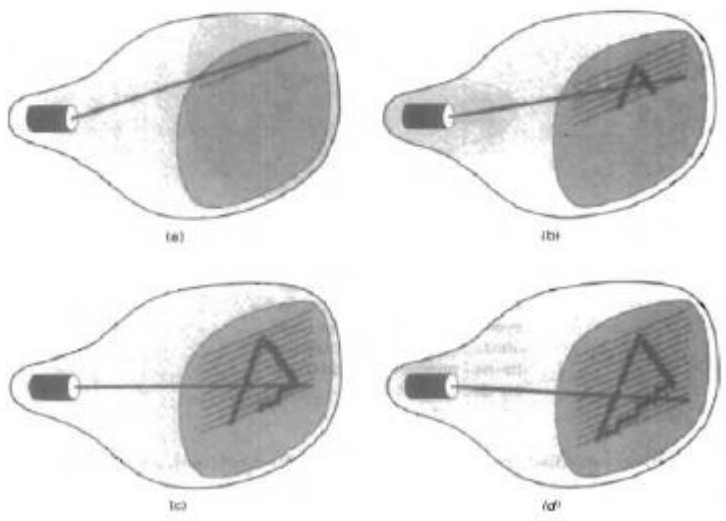
Q2. Explain Refresh and Raster scan Display System.

Ans

Our home television sets use Raster scan technologies. In this sort of Display Mechanism, an electron beam scans every row of the screen display row by row starting from top to the bottom. Each screen point represents the intensity value either 0 or 1 and the intensity value is kept in refresh buffer or frame buffer. Thus, each pixel value or screen point keeps on changing from 0 to 1 or from 1 to 0 depending on its intensity value in refresh buffer. And this is the way the screen is painted one row at a time. And this is shown in the Figure.

The range of the intensity depends upon the system capabilities. We can plot only two different colors or intensities if it is a black and white system. In this case one bit per pixel is enough, 1 for white intensity and bit value '0' for black intensity. More bits can be used to display color and intensity for colors. So, bitmap is the term used for frame buffer for black and white systems and Pixmap is the term which is used for Frame buffer which stores multiple bits per pixel.

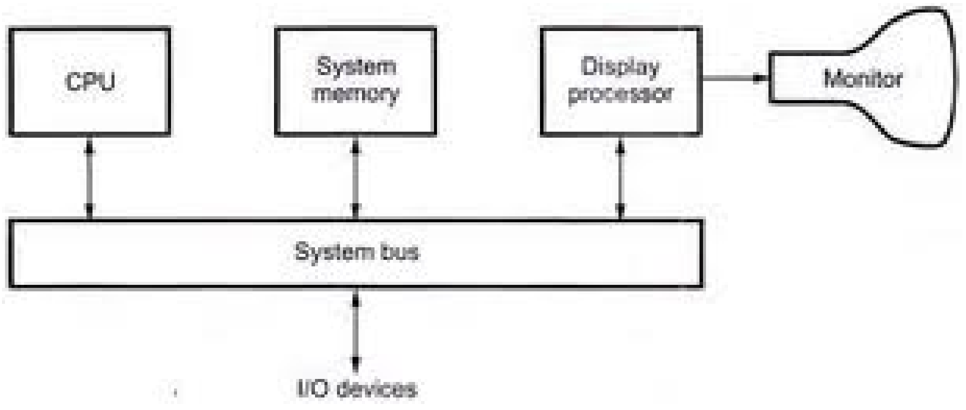
For raster system the refresh rate is generally 60 to 80 frames per second, it can be higher for some systems. After it scans one row and it returns to the left of the screen for scanning next row, it is called horizontal retrace. After it has refreshed each scan line, it moves to the top left corner of the display and again starts the refreshing process and this is called vertical retrace.



Raster scan systems are much more capable than the random systems. As it stores the intensity values for each screen position, it is capable of displaying the color variations and shade which is not possible with random systems. But raster system has lower resolution as compared to random system. This is because, random system follows the line path to be drawn and line drawing commands are stored in refresh buffer. For raster system, intensity values are stored for each screen

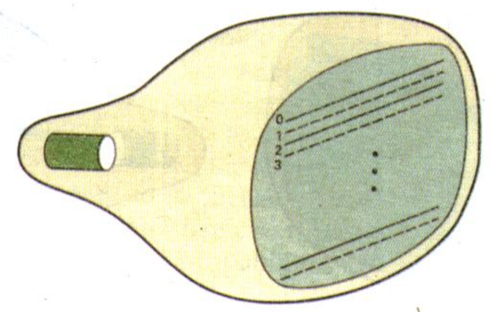
Q.3 Explain Random Scan Display

Ans The arrangement of a simple random scan system is shown in the following figure. System stores and application program in the system memory along with a graphics package. With the help of graphics package the Graphics command in the application program are converted into a display file stored in the system memory. And this file helps the system to refresh the screen. When operated as a **random-scan** display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random-scan monitor draw a picture one line at a time and for this reason are also referred to as **vector** displays (or **stroke-writing** of **calligraphic** displays).



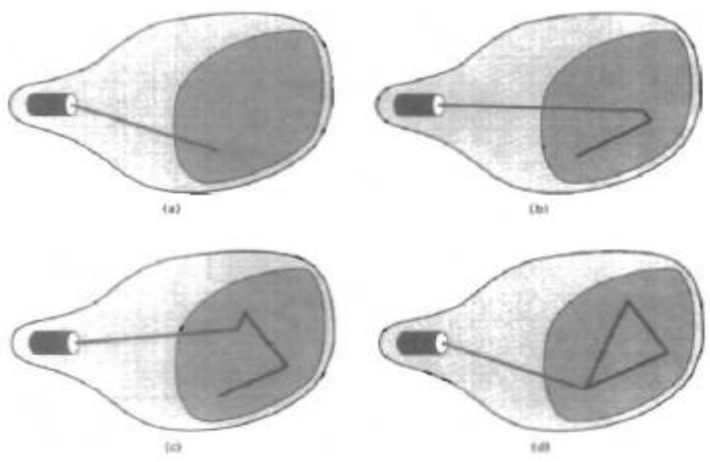
The component lines of a picture can be drawn and refreshed by a random-scan system in any specified order Figure. A pen plotter in a similar way and is an example of a random-scan, hard-copy device.

Refresh rate on a random-scan system depends on the number of lines to be displayed. Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the refresh display file. Sometimes the refresh display file is called the display list, display program,



or simply the refresh buffer. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all line drawing commands have been processed, the system cycle back to the first line command in the list. Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second.

High-quality vector systems are capable of handling approximately 100,000 “short” lines at this refresh rate. When a small set of lines is to be displayed, each refresh cycle is delayed to avoid refresh rates greater than 60 frames per second. Otherwise, faster refreshing of the set of lines could burn out the phosphor. Random-scan systems are designed for line-drawing applications and can-not display realistic shaded scenes.



Since picture definition is stored as a set of line-drawing instruction and not as a set of intensity values for all screen points, vector displays generally have higher resolution then raster system. Also, vector displays produce smooth line drawings because the CRT beam directly follows the line path. A raster system, in contrast, produces jagged lines that are plotted as discrete point sets.

Q4 **Explain the differences between Raster Scan and Random Scan Display?**

Ans. :

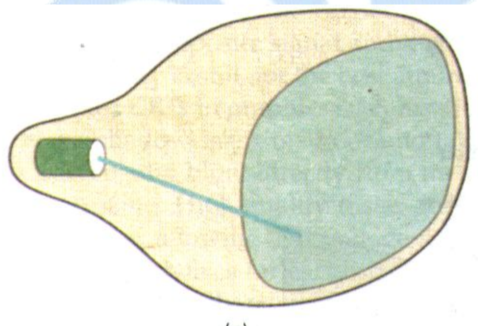
Raster Scan	**Random Scan**
Refresh Process for Raster Scan display occurs at the rate of 60 to 80 frames per second.	Refresh rate depends on number of lines to be displayed. Refresh cycle is displayed to avoid refresh rate greater than 60 frames per second for small set of lines.
This provides higher resolution.	This provides lower resolution.
The electron beam is swept across the screen, one row at a time from top to bottom.	The electron beam is projected0 only to the parts of the screen where a picture is to be drawn.
This display porous produces smooth line drawings as the CRT beam directly follows the line path.	In this display it produces jagged lines that are potted as discrete point sets.

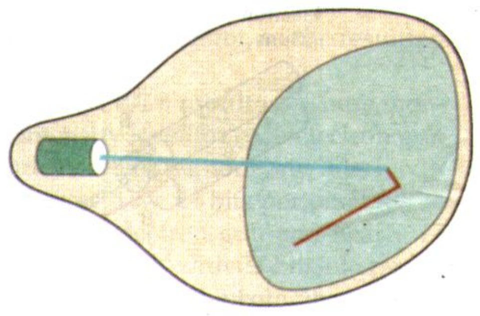
illuminated spots are created as a pattern.	Picture is drawn one line at a time.

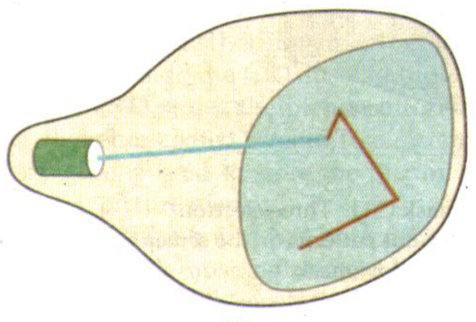
Q.5 Write short note on Color CRT Monitor. Explain Shadow Mask Method.

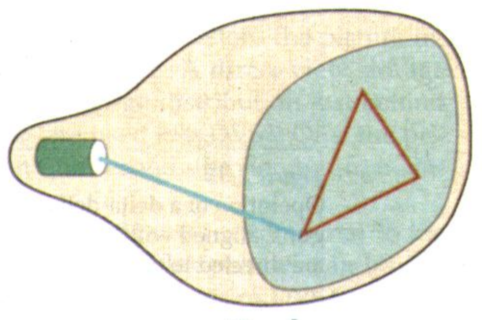
Ans.: A CRT monitor displays color picture by using a combination of phosphor that emit different-colored light. By combining the emitted light from the different phosphor, a range of colors can be generated. The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadow-mask method.

The **beam-penetration** method for displaying color pictures has been used with random-scan monitors. Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen, and the displayed color depends on how far the electron beam penetrates into the phosphor layers. A beam of slow electrons excites only the outer red layer. A beam of very fast electron penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow. The speed of the electrons, and hence the screen color at any point, is controlled by the beam-acceleration voltage. Beam penetration has been an inexpensive way to produce color in random-scan monitor, but only four colors are possible, and the quality of picture is not as good as with other methods.

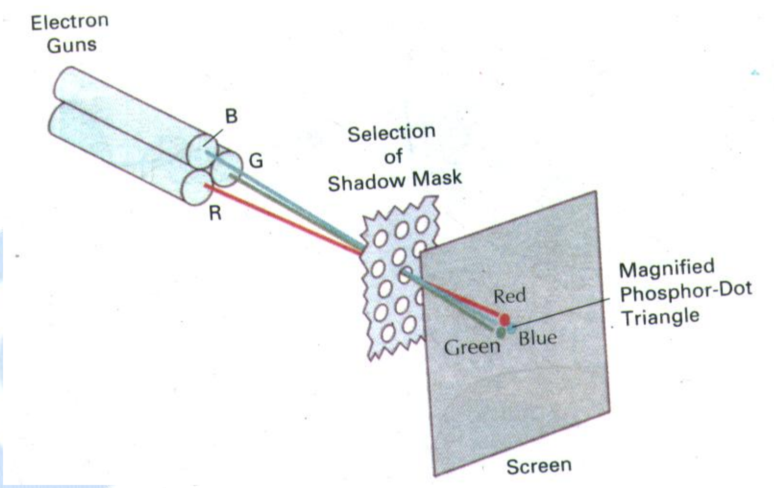

(a)


(b)


(c)


(d)

Shadow-mask methods are commonly used in raster-scan system (including color TV) because they produce a much wider range of colors than the beam penetration method. A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type ofCRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen. Figure 2-10 illustrates the delta-delta shadow-mask method, commonly used in color CRT system. The three beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the shadow mask. Another configuration for the three electron guns is an in-line arrangement in which the three electron guns, and the corresponding red-green-blue color dots on the screen, are aligned along one scan line instead of in a triangular pattern. This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color CRTs.



We obtain color variations in a shadow-mask CRT by varying the intensity levels of the three electron beams. By turning off the red and green guns, we get only the color coming from the blue phosphor. Other combinations of beam intensities produce a small light spot for each pixel position, since our eyes tend to merge the three colors into one composite. The color we see depends on the amount of excitation of the red, green, and blue phosphors. A white (or gray) area is the result of activating all three dots with equal intensity. Yellow is produced with the green and red dots only, magenta is produced with the blue and red dots, any cyan shows up when blue and green are activated equally. In some low-cost systems, the electron beam can only be set to on or off, limiting displays to eight colors. More sophisticated systems can set intermediate intensity level for the electron beam, allowing several million different colors to be generated.

Color graphics systems can be designed to be used with several types of CRT display devices. Some inexpensive home-computer system and video games are designed for use with a color TV set and an RF (radio-frequency) modulator. The purpose of the RF modulator is to simulate the signal from a broad-cast TV station. This means that the color and intensity information of the picture must be combined and superimposed on the broadcast-frequency carrier signal that the TV needs to have as input. Then the circuitry in the TV takes this signal from the RF modulator, extracts the picture information, and paints it on the screen. As we might expect, this extra handling of the picture information by the RF modulator and TV circuitry decreased the quality of displayed images.

Chapter 2

Output Primitives

Q.1 Explain in details the Line Drawing Algorithms using DDA Algorithm.

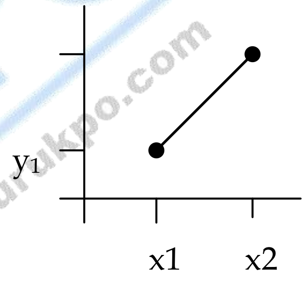
Ans.: Straight Line Equation in the form of Slope intercept is as follows:

$$y = m x + b \qquad \text{------------1}$$

where m represents the slope of the line and b as the y intercept which it makes with the Y axis. The two end point of a line segment are denoted by the positions (x_1, y_1) and (x_2, y_2) as shown in the following diagram. Using this Equation we can determine values for the slope m and y intercept b using the following calculations.

$$M = \frac{y_2 - y_1}{x_2 - x_1} \qquad \text{----2}$$

$$b = y_1 - mx_1 \qquad \text{-----3}$$



Value of y is calculated

$$\Delta y - m \,.\, \Delta x \qquad \_\,\_\,\_\,(4)$$

Similarly we can obtain Δ x interval **Figure. (1) Line Path between endpoint**

$$\Delta x = \frac{\Delta y}{m}$$

position (x_1, y_1) & (x_2, y_2)

For lines with slope magnitude $|m|>1$, Δy can be set proportional to a small deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx.

For lines with m = 1 Δx = Δy.

DDA Algorithm : Also Called as Digital Differential Analyzer (DDA) performs scan. It is a Scan conversion line algorithm. It is also called an incremental algorithm, as it increments the value of x or y by 1 depending on the slope value. It tries to decrease the computation burden and increase the sped of computing. Conversion line Algorithm based on calculating either Δy or Δx using equation (4) & (5).

We sample the line at unit intervals in one coordinate and determine corresponding integer values nearest. The line paths for the other coordinate. Now consider first a line with positive slope, as shown in Figure.(1). If the slope is less than one or equal to 1. We sample at unit x intervals ($\Delta x = 1$) compute each successive y values as :

$$y_{k+1} = y_k + m \qquad \_\_\_ (6)$$

Value k takes integer values starting form 1, for the first point & gets incremented by 1 until the final end point is reached.

For lines with positive slope greater than 1, we reverse the role of x and y. That is we sample at unit y intervals ($\Delta y = 1$) and calculate each succeeding x value as :

$$x_{k+1} = x_k + \frac{1}{m} \qquad \_\_\_ (7)$$

Equation (6) and (7) are based on assumption that lines are to be processed form left end point to the right end point.

If this processing is reversed the sign is changed

$$\Delta x = -1 \qquad \& \qquad \Delta y = -1$$

$$y_{k+1} = y_k - m \qquad \_\_\_ (8)$$

$$x_{k+1} = x_k - \frac{1}{m} \qquad \_\_\_ (9)$$

Equations (6) to (9) are used to calculate pixel position along a line with negative slope.

When the start endpoint is at the right we set $\Delta x = -1$ and obtain y position from equation (7) similarly when Absolute value of Negative slope is greater than 1, we use $\Delta y = -1$ & eq.(9) or we use $\Delta y = 1$ & eq.(7).

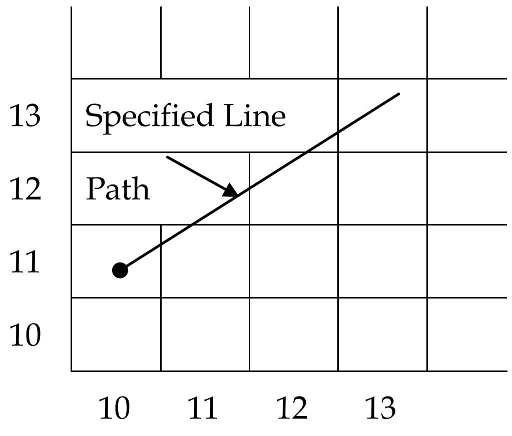
Q.2 Explain in details the Line Drawing Algorithms using Bresenham's Line Drawing Algorithm.

Ans

This is very efficient and faster line drawing algorithm. It scan converts lines and uses only incremental integer calculations. Thus we can use this algorithm for drawing circles and similar other curves also. An accurate and efficient raster line generating Algorithm, developed by Bresenham, scan concerts line using only incremental integer calculations that can be adapted to display circles and other curves. The vertical axes show scan-line position, & the horizontal axes identify pixel columns as shown in Figure. (5) & (6) . This algorithm follows the closeness theory to implement line plotting.

As we did for DDA algorithm, here also we start with I octant where slope , m<1. Since m<1, we move in x-direction by sampling at unit x intervals.

Thus we start plotting from initial, say(x0-y0) and take steps in success x-columns an plot the point whose y-values is closest to the ideal line path.



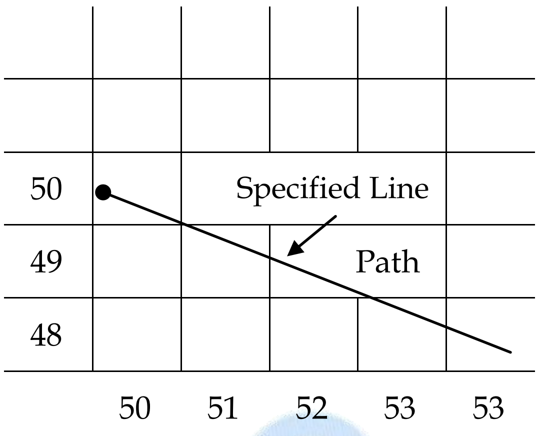


Figure.5 **Figure.6**

We first consider the scan conversion process for lines with positive slope less than 1 to illustrate Bresenham's approach. Pixel position along a line path are then determined by sampling at unit x intervals starting form left and point (x_0 , y_0) of a given line, we step at each successive column (x position) & plot the pixel whose scan line y is closest to the line path. Now assuming we have to determine that the pixel at (x_k , y_k) is to be displayed, we next need to divide which pixel to plot in column x_{k+1}. Preference would be at the position (x_{k+1} , y_k) and (x_{k+1} , y_{k+1}). At sampling position x_{k+1}, we label vertical pixel separations from the mathematical line path as d_1 and d_2 Figure.(8).

The y coordinate on the mathematical line at pixel column position x_{k+1} is calculated as :

$$y = m(x_k + 1) +b \qquad \_\_\_(10)$$

Then $d_1 = y - y_k = m (x_k + 1) +b - y_k$

$$d_2 = (y_k + 1) -y = y_k + 1 - m (x_k + 1) - b$$

The difference can be define between these two separations as

$$d_1 - d_2 = 2m (x_k+1) - 2y_k + 2b - 1 \qquad \_\_\_ (11)$$

The deciding Parameter P_k for the K^{th} step in the line algorithm can be obtained by making some rearrangements in eq.(11) so that it involves sort of integer calculation. We accomplish this by substituting $m = \Delta y/\Delta x$. where Δy & Δx are the vertical & horizontal separation of the endpoint positions & defining. The sign of P_k remains same as that of the sign of $d_1 - d_2$.

$$P_k = \Delta x (d_1 - d_2) = 2\Delta y. x_k - 2\Delta x\, y_k + c \qquad \_\_\_ (12)$$

Since $\Delta x > 0$ for our example Parameter C is constant & has the value $2\Delta y + \Delta x\,(2b-1)$, which is independent of pixel position.

If the pixel position at y_k is closer to line path than the pixel at y_{k+1} (that is $d_1 < d_2$), then decision Parameter P_k is Negative. In that case we plot the lower pixel otherwise we plot the upper pixel. Coordinate changes along the line owner in unit steps in either the x or directions. Therefore we can obtain the values of successive decision Parameter using incremental integer calculations. At step k = 1, the decision Parameter is evaluated form eq.(12) as :

$$P_{k+1} = 2\Delta y \,.\, x_{k+1} - 2\Delta x \,.\, y_{k+1} + C$$

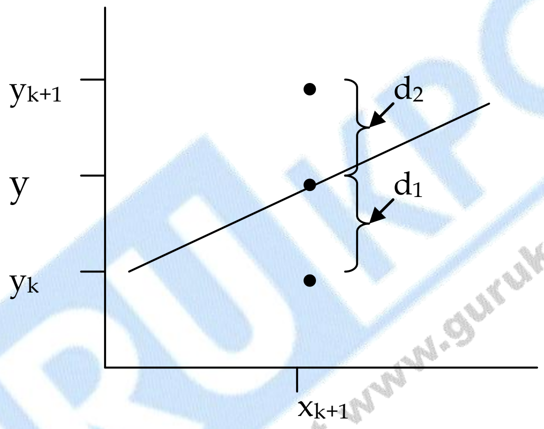


Figure.8

Subtracting eq.(12) from the preceding equation we have

$$P_{k+1} - P_k = 2\Delta y\,(x_{k+1} - x_k) - 2\Delta x\,(y_{k+1} - y_k)$$

But $x_{k+1} = x_k + 1$

So that, $P_{k+1} = P_k + 2\Delta y - 2\Delta x\,(y_{k+1} - y_k)$ ___ (13)

The term $y_{k+1} - y_k$ either results into 0 or 1, depending on sign of Parameter P_k.This recursive calculation of decision Parameter is performed each integer x position, starting at left coordinate endpoint of the line. The first parameter P_0 is evaluated from equation (12) at starting pixel position (x_0, y_0) and with m evaluated as $\Delta y/\Delta x$.

$P_0 = 2\Delta y - \Delta x$ ___ (14)

The following lines express how does Bresenham's Line Drawing Algorithm work for $|m|<1$:

Take input for two endpoints of a line & store the left end point in (x_0, y_0). Load (x_0, y_0) into frame buffer that is plot the first point. Calculate constants Δx, Δy, $2\Delta y$ and $2\Delta y - 2\Delta x$ and obtain the starting value for the decision parameter as : $P_0 = 2\Delta y - \Delta x$. At each x_k along the line starting at k = 0, perform the following test if $P_k < 0$ the next point to plot is (x_{k+1}, y_k) and $P_{k+1} = P_k + 2\Delta y$ otherwise the next point to plot is (x_{k+1}, y_{k+1}) and $P_{k+1} = P_k + 2\Delta y - 2\Delta x$. Repeat step 4 Δx times.

Q.3 Digitize the line with end points (20, 10) & (30, 18) using Bresenham's Line Drawing Algorithm.

Ans.: slope of line, $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{18 - 10}{30 - 20} = \frac{8}{10} = 0.8$

$\Delta x = 10$, $\Delta y = 8$

Initial decision parameter has the value

$P_0 = 2\Delta y - \Delta x = 2x8 - 10 = 6$

Since $P_0 > 0$, so next point is $(x_k + 1, y_k + 1)$ (21, 11)

Now k = 0, $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_1 = P_0 + 2\Delta y - 2\Delta x$

$= 6 + (-4)$

$= 2$

Since $P_1 > 0$, $\therefore$ Next point is (22, 12)

Now k = 1, $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_2 = 2 + (-4)$

$= -2$

Since $P_2 < 0$, $\therefore$ Next point is (23, 12)

Now k = 2 $P_{k+1} = P_k + 2\Delta y$

$P_2 = -2 + 16$

$= 14$

Since $P_3 > 0$, $\therefore$ Next point is (24, 13)

Now k = 3 $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_4 = 14 - 4$

$= 10$

Since $P_4 > 0$, $\therefore$ Next point is (25, 14)

Now k = 4 $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_5 = 10 - 4$

$= 6$

Since $P_5 > 0$, $\therefore$ Next point is (26, 15)

Now k = 5 $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_6 = 6 - 4$

$= 2$

Since $P_6 > 0$, $\therefore$ Next point is (27, 16)

Now k = 6 $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_7 = 2 + (-4)$

$= -2$

Since $P_7 < 0$, $\therefore$ Next point is (28, 16)

Now k = 7 $P_{k+1} = P_k + 2\Delta y$

$P_8 = -2 + 16$

$= 14$

Since $P_8 > 0$, $\therefore$ Next point is (29, 17)

Now k = 8 $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_9 = 14 - 4$

$= 10$

Since $P_9 > 0$, $\therefore$ Next point is (30, 18)

K	P_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)
1	2	(22, 12)
2	-2	(23, 12)
3	14	(24, 13)
4	10	(25, 14)
5	6	(26, 15)
6	2	(27, 16)
7	-2	(28, 16)
8	14	(29, 17)
9	10	(30, 18)

Plot the graph with these following points.

Q.4 Explain how does the Mid Point Circle Algorithm work.

Ans.: The equation of a circle can be given as follows, where (x_c, y_c) represents the centre coordinates.

$$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0$$

In the following way the calculation is made for the position of points along the circlular path by moving in the x direction from $(x_c - r)$ to $(x_c + r)$ and determining the corresponding y values as :

$$y = y_c \pm \sqrt{(x_c - x)^2 - r^2}$$

As it requires heavy computation this method is not the best method to calculate the circle point coordinates. Moreover spacing between the points is not uniform. Another method that can be used by calculating the polar coordinates r and θ where

$$x = x_c + r \cos \theta$$

$$y = y_c + r \sin \theta$$

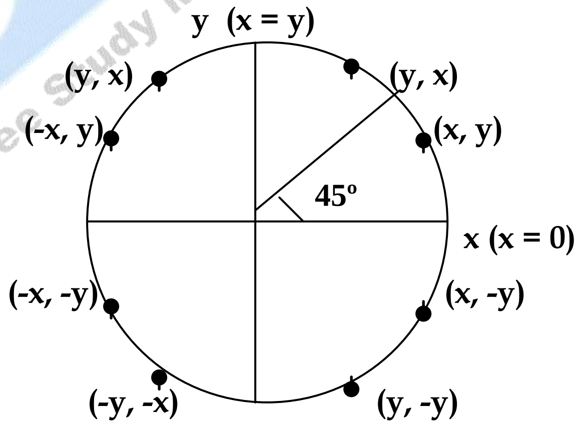
It requires heavy computation but this method results in equal spacing between the points. The efficient method is incremental calculation of decision parameter.

Mid Point Algorithm :

We assume that we are working in II octant of the circle .The concept behind Mid point circle is that, a midpoint M lies between two points and we have to decide if M lies between two points and we have to decide if M lies inside or outside the circle. This would tell the next point to be plotted along the circumference of the circle. We move in unit steps in the x-direction and calculate the closed pixel position along the circle path at each step.

For a given radius r & screen center position (x_c, y_c). We first set our Algorithm to calculate the position of points along the coordinate position (x_0, y_0). These calculated positions are then placed at this proper screen position by adding x_c to x and y_c to y. For a circle from x = 0 to x = y in first quadrant, the slope varies from 0 to 1.

We move in the positive x direction and determine the decision parameter to find out the possible two y values along the circle path. And the Points calculation in other 7 octants is done using the symmetry pattern.

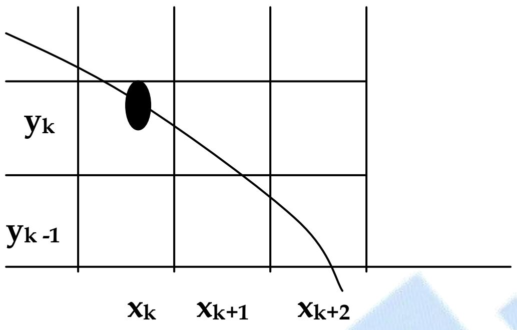


The following function is used for the implementation of this method :

$$f_{circle}(x, y) = x^2 + y^2 - r^2 \qquad \text{\_\_\_ (1)}$$

Any point (x, y) on the boundary of the circle with radius r satisfies the equation of $f_{circle}(x, y) = 0$. The relative position of any point (x, y) can be determined by checking the sign of circle function.

$$f_{circle}(x, y) \begin{cases} < 0 \text{ if } (x, y) \text{ denotes it inside circle boundary.} \\ = 0 \text{ if } (x, y) \text{ denotes it on circle boundary.\_ \_ (2)} \\ > 0 \text{ if } (x, y) \text{ denotes it outside circle boundary.} \end{cases}$$



Taking an assumption that we have just plotted a pixel at (x_k, y_k). We next need to determine whether the pixel (x_{k+1}, y_k) or (x_{k+1}, y_{k-1}) is closer. Our decision parameter is the circle function evaluated at the mid point between these two pixels.

$P_k = f_{circle}(x_k + 1, y_k - ½)$

Or $P_k = (x_k + 1)^2 + (y_k - ½)^2 - r^2$ _ _ _ (3)

This denotes that If $P_k < 0$, Mid point is inside the circle boundary and the pixel on the scan line y_k is closer to the circle boundary. Otherwise, Mid point is on or outside the circle boundary and the point on the scan line y_k - 1 is closer. Successive decision parameters are obtained by incremental calculations. Again the next deciding parameter is calculate the position at next sampling position by taking the next position.

$x_{k+1} + 1 = x_k + 2$

$P_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - ½)$

Or $P_{k+1} = [(x_k + 1) + 1]^2 + (y_{k+1} - ½)^2 - r^2$

Or $P_{k+1} = P_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$ _ _ _ (4)

Successive increment for P_k is $2x_{k+1} + 1$(If $P_k < 0$) otherwise $(2x_{k+1} + 1 - 2y_{k+1})$ where

$2x_{k+1} = 2x_k + 2$ & $2y_{k+1} = 2y_k - 2$

Initial decision parameter P_0 is obtained as $(0, r) = (x_0, y_0)$

$P_0 = f_{circle}(x, y) = f_{circle}(1, r - ½) = 1 + (r - ½)^2 - r^2$

Or $P_0 = \frac{5}{4} - r$

If r is a integer then $P_0 = 1 - r$

Algorithm for this can be defined in the following steps for calculating the Mid Point:

(1) Input radius r and circle center (x_c, y_c) and obtain the first point on circumference of a circle centered on origin (x_0, y_0) = (0, r)

(2) Calculate the initial value of the decision parameter as : $P_0 = \frac{5}{4} - r$

(3) At each x_k position, starting at k = 0 if $P_k < 0$ the next point along the circle is (x_{k+1}, y_k) and $P_{k+1} = P_k + 2x_{k+1} + 1$, otherwise the next point along the circle is ($x_k + 1$, $y_k - 1$) and $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$ where $2x_{k+1} = 2x_k + 2$ & $2y_{k+1} = 2y_k - 2$.

(1) Determine symmetry points in other seven octants.

(2) Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) & plot coordinate values $x = x + x_c$ & $y = y + y_c$.

(3) Repeat step (3) through (5) until $x \geq y$.

Q.5 Demonstrate the Mid Point Circle Algorithm with circle radius, r = 10.

Ans.: $P_0 = 1 - r = 1 - 10 = -9$

Now the initial point (x_0, y_0) = (0, 10) and initial calculating terms for calculating decision parameter are

$2x_0 = 0$, $2y_0 = 20$	Since $P_k < 0$, ∴ Next point is (1, 10)
$P_1 = -9 + 3 = -6$	Now $P_1 < 0$, ∴ Next point is (2, 10)
$P_2 = -6 + 5 = -1$	Now $P_2 < 0$, ∴ Next point is (3, 10)
$P_3 = -1 + 7 = 6$	Now $P_3 > 0$, ∴ Next point is (4, 9)
$P_4 = 6 + 9 - 18 = -3$	Now $P_4 < 0$, ∴ Next point is (5, 9)
$P_5 = -3 + 11 = 8$	Now $P_5 > 0$, ∴ Next point is (6, 8)
$P_6 = 8 + 13 - 16 = 5$	Now $P_6 > 0$, ∴ Next point is (7, 7)

K	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	(1, 10)	2	20
1	(2, 10)	4	20

2	(3, 10)	6	20
3	(4, 9)	8	18
4	(5, 9)	10	18
5	(6, 8)	12	16
6	(7, 7)	14	14

Plot the graph with these points.

Q.6 Give the properties of Ellipse & explain the Algorithm.

Ans.: Ellipse is an elongated form of circle or in other words, circle is an special case of ellipse where the two radii of the circle are equal. cvcAn ellipse explained as a set of points such that the sum of distances from two fixed points (foci) is same for all points given a point P = (x, y), distances are d_1 & d_2, equation is :

$$d_1 + d_2 = \text{constant} \qquad \text{\_\_\_ (1)}$$

In terms of local coordinates

$F_1 = (x_1, y_1)$ & $F_2\ (x_2, y_2)$

Equation is :

$$\sqrt{(x - x_1)^2 + (y - y_1)^2} + \sqrt{(x - x_2)^2 + (y - y_2)^2} = 0 \qquad \text{\_\_\_ (2)}$$

This can also be written in the form :

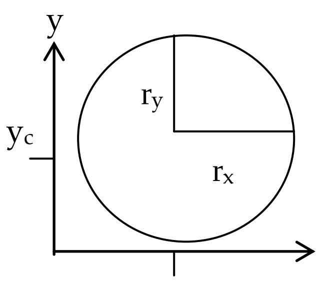
$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0 \qquad \text{\_\_\_ (3)}$$

(More A, B, C, D, E, & F are evaluated in terms of focal coordinates & major minor axis).

- Major axis – which extends form 1 point to other through foci.
- Minor axis – which spans the shouter dimension bisecting major axis at ellipse center.

An interactive method for specifying an ellipse in an arbitrary orientation is to input two foci & a point on ellipse boundary & evaluate constant in equation (1) & so on.

Equation can be simplified if ellipse is in “standard position” with major & minor axis oriented parallel to x and y axis.



x_c x

$$\left(\frac{x - x_c}{rx}\right)^2 + \left(\frac{y - y_c}{ry}\right)^2 = 1 \qquad \_\_\_(4)$$

Now using polar coordinates r & θ

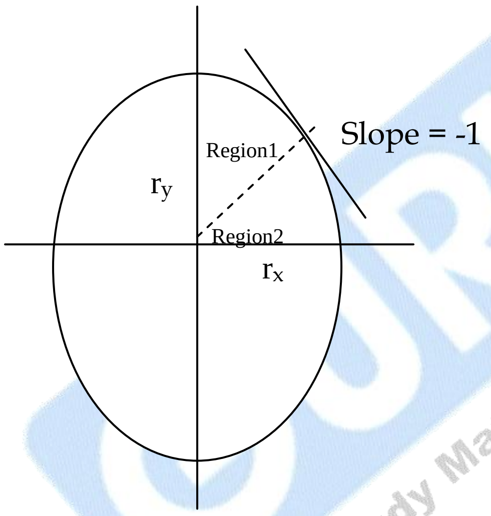
Parameter equations are :

$x = x_c + r_x \cos\theta$

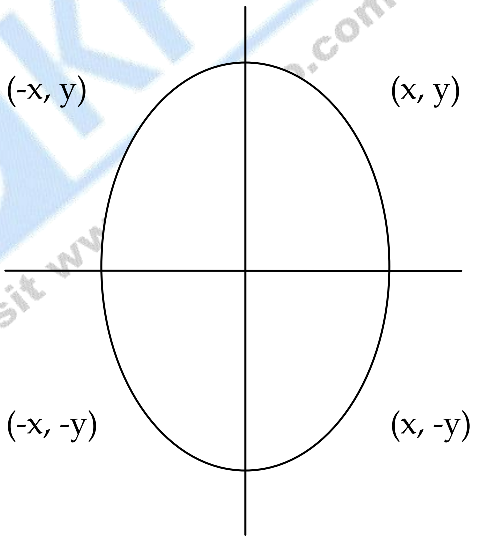
$y = y_c + r_y \sin\theta$

Mid Point Ellipse Algorithm

Assuming that we are calculating points in the first quadrant. As it is quite difficult to calculate points in first quadrant in one go, we need to divide it in 2 regions or parts Here approach is similar to that used in circle r_x and r_y and (x_c, y_c) obtain points (x, y) for ellipse centered on origin. Mid point ellipse algorithms process the quadrants in 2 parts. Figure shows the division of quadrant with $r_x < r_y$.



Figure



Figure

In region 2, we take unit steps in y-direction and magnitude of slope is greater than 1. Region 1 & 2 can be processed in different ways: Region 1 is processed starting at $(0, r_y)$ we step clockwise along the path in unit steps in x- direction & then unit steps in y- direction.

The function for Ellipse can be written as follows:

$f_{ellipse}(x, y) = (0, r_y)$

$f_{ellipse}(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 \qquad \_\_\_ (5)$

And it has the following Attributes:

$$f_{ellipse}\begin{cases} < 0 & (x, y) \text{ is inside boundary.} \\ = 0 & (x, y) \text{ is on the boundary.} \\ > 0 & (x, y) \text{ is outside the boundary.} \end{cases}$$

And this is how the ellipse function $f_{ellipse}(x, y)$ serves as a decision parameter in the mid point Algorithm. Starting at $(0, r_y)$ we step in x direction until we reach boundary between Region 1 & 2 slope is calculated at each step as :

$$\frac{dy}{dx} = \frac{-2r_y^2 x}{2r_x^2 x} \qquad \text{- \{from eq.(5)\}}$$

At boundary $\frac{dy}{dx} = -1$ So, $2r_y^2 x = 2r_x^2 y$

$\therefore$ We move out of Region 1 when $2r_y^2 x \geq 2r_x^2 y$ _ _ _ (7)

Figure (1) shows mid point between 2-candidate pixel at (x_{k+1}), we have selected pixel at (x_k, y_k) we need to determine the next pixel.

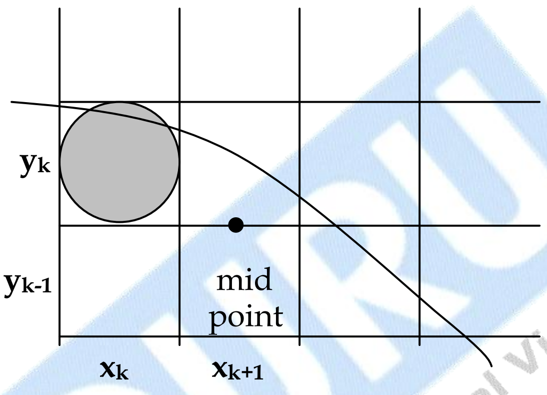


Figure.:1

$P1_k = f_{ellipse}(x_k + 1, y_k - ½) = r_y^2(x_k + 1)^2 + r_x^2 (y_k - ½)^2 - r_x^2 r_y^2$ _ _ _ (8)

Next symmetric point $(x_{k+1} + 1, y_{k+1} - ½)$

$P1_{k+1} = r_y^2 [(x_k + 1) + 1]^2 + r^2_x [(y_{k+1} - ½)^2 (y_k - ½)^2]$ _ _ _ (9)

Where y_{k+1} is either y_k or $y_k - 1$ depending on sign of P_{1k}

$$\text{Increments} = \begin{cases} 2r_y^2 x_{k+1} + r_y^2 & \text{if } P_k < 0 \\ 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2 & \text{if } P_k \geq 0 \end{cases}$$

With initial position $(0, r_y)$ the two terms evaluate to

$2r_y^2 x = 0$, $2r_x^2 y = 2r_x^2 r_y$

Now when x & y are incremented the updated values are

$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2$, $2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$

And these values are compared at each step & we move out of Region 1 when condition (7) is satisfied initial decision parameter for region 1 is calculated as :

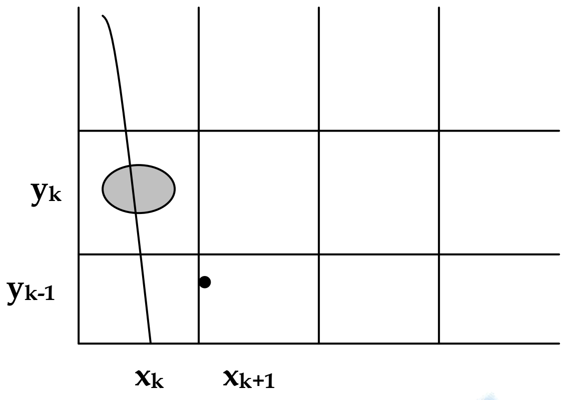
$P1_0 = f_{ellipse}(x_0, y_0) = (1, r_y - ½) = r_y^2 + r_x^2(r_y - ½)^2 - r_x^2 r_y^2$

$P1_0 = r_y^2 - r_x^2 r_y + ¼ r_x^2$ ___ (10)

Over Region 2, we step in (-)ve y-direction & mid point selected is between horizontal pixel positions

$P2_k = f_{ellipse}(x, y) = (x_k + ½, y_k - 1)$

$P2_k = r_y^2(x_k + ½)^2 + r_x^2(y_k - 1)^2 - r_x^2r_y^2$ ___ (11)



If $P2_k > 0$ then we select pixel at x_{k+1}

Initial decision parameter for region (2) is calculated by taking (x_0, y_0) as last point in Region (1)

$P2_{k+1} = f_{ellipse}(x_{k+1} + ½, y_{k+1} -1) = r_y^2(x_{k+1} + ½)^2 + r_x^2[(y_k - 1) -1]^2 - r_x^2r_y^2$

$P2_{k+1} = P_{2k} - 2r_x^2(y_k - 1) + r_x^2 + r_y^2[(x_{k+1} + ½)^2 - (x_k + ½)^2]$ ___ (12)

At initial position (x_0, y_0)

$P2_0 = f_{ellipse}(x_0 +½, y_0 - 1) = r_y^2(x_0 + ½)^2 + r_x^2(y_0 - 1)^2 - r_x^2r_y^2$ ___ (13)

Mid Point Ellipse Algorithm :

(1) Take Input of r_x, r_y and ellipse centre (x_c, y_c) obtain the first point $(x_0, y_0) = (0, r_y)$

(2) Calculate the initial value of the decision parameter in region 1 as

$P1_0 = r_y^2 - r_x^2r_y + ¼ r_x^2$

(3) At each x_k position in region 1, starting at K = 0. If $P_k < 0$, then next point along ellipse is (x_{k+1}, y_k) and $P1_{k+1} = P1_k + 2r_y^2x_{k+1} + r_y^2$ otherwise next point along the circle is $(x_k +1, y_k -1)$ and $P1_{k+1} = P1_k + 2r_y^2x_{k+1} - 2r_x^2y_{k+1} + r_y^2$ with $2r_y^2x_{k+1} = 2r_y^2x_k + 2r_y^2$, $2r_x^2y_{k+1} = 2r_x^2y_k - 2r_x^2$ and continue until $2r_y^2x \geq 2r_x^2y$.

(4) Calculate the initial value of the decision parameter in region (2) using last point (x_0, y_0) calculated in region 1 as $P2_0 = r_y^2 (x_0 + ½)^2 + r_x^2 (y_0 - 1)^2 - r_x^2r_y^2$.

(5) At each y_k position in region (2), starting at k = 0 perform following test : If $P2_k > 0$ next point is $(x_k, y_k - 1)$ and $P2_{k+1} = P2_k - 2r_x^2y_{k+1} +$

r_x^2 otherwise next point is ($x_k + 1, y_k - 1$) and $P2_{k+1} = P2_k + 2r_y^2x_{k+1} - 2r_x^2y_{k+1} + r_x^2$.

(6) Determine the symmetry points in other three quadrants.

(7) More each calculated pixel position (x, y). Center on (x_c, y_c), plot coordinate values $x = x + x_c$ & $y = y + y_c$.

(8) Repeat the steps for region 1 until $2r_y^2x \geq 2r_x^2y$.

Illustrate the Mid Point Ellipse Algorithm by ellipse parameter $r_x = 8$ $r_y = 6$

Ans.: $r_x^2 = 64$, $r_y^2 = 36$

$2r_x^2 = 128$, $2r_y^2 = 72$

$P1_0 = 36 - (64 \times 6) + \frac{1}{4} \times 64 = 36 - 384 + 16 = -332$

$P1_0 < 0$ $\therefore$ Next point is (1, 6)

$P1_1 = -332 + 72 \times 1 + 36 = -332 + 108 = -224$

$P1_1 < 0$ $\therefore$ Next point is (2, 6)

$P1_2 = -224 + 72 \times 2 + 36 = -224 + 144 + 36 = -44$

$P1_2 < 0$ $\therefore$ Next point is (3, 6)

$P1_3 = -44 + 72 \times 3 + 36 = -44 + 216 + 36 = 208$

$P1_3 > 0$ $\therefore$ Next point is (4, 5)

$P1_4 = 208 + 72 \times 4 - 128 \times 5 + 36 = 208 + 288 - 640 + 36 = -108$

$P1_4 < 0$ $\therefore$ Next point is (5, 5)

$P1_5 = -108 + 72 \times 5 + 36 = 288$

$P1_5 > 0$ $\therefore$ Next point is (6, 4)

$P1_6 = 288 + 72 \times 6 - 128 \times 4 + 36 = 244$

$P1_6 > 0$ $\therefore$ Next point is (7, 3)

$P1_7 = 244 + 72 \times 7 - 128 \times 3 + 36 = 1168$

K12	$P1_k$	(x_{k+1}, y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
1	– 332	(1, 6)	72	768
1	– 224	(2, 6)	144	768
2	– 44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	– 108	(5, 5)	360	640

5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

Now we move to region 2 since $2r_x^2 > 2r_x^2y$

Now For region the initial point is (x0, y0 = (7, 3) the initial decision parameter is

$P2_0 = f(7 + ½, 3 – 1) = 36\ (7 + ½)^2 + 64\ (2)^2 – 64 \times 36 = 2025 + 256 – 2304 =– 23$

The remaining positions are then calculated as :

K	**$P1_k$**	**(x_{k+1}, y_{k+1})**	**$2r_y^2x_{k+1}$**	**$2r_x^2y_{k+1}$**
0	– 23	(8, 2)	576	256
1	– 215	(8, 1)	576	128
2	– 279	(8, 0)	-	-

Now Plot these point on the graph.

Q.7 Explain Why is DDA Algorithm not good & efficient Algorithm?

Ans.: (1) DDA traces out the successive x & y values by simultaneously increasing x & y by small steps proportional to their first derivative. In our example the x increment is 1 but y increment is $\frac{dy}{dx}$ = m. since the real values have limited Precision, the accumulation of round off error in "m" causes the accumulative error. Build up which drifts the pixel positions from the true line path in most lines.

(2) Moreover the round off operations & floating point incrementation is still time consuming.

Q.8 What do you understand by Area Filling? Discuss any one Algorithm.

Ans.: A standard out put primitive in general graphics package is a solid-color or patterned polygon area. There are two basic approaches to area filling on raster system :

(1) To fill an area is to determine the overlap intervals for scan lines that cross the area.

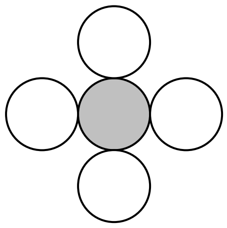
(2) Another is t start from a given interior position & point outward from this point until we specify the boundary conditions.

Now scan line approach is typically used in general graphics package to still polygons, circles, ellipses and simple curses.

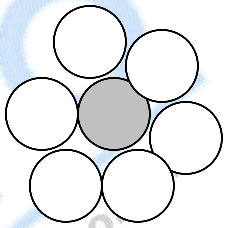
Fill methods starting from an interior point are useful with more complex boundaries and in interactive painting systems.

Boundary Fill Algorithm : Another approach to area filling is to start at a point inside a region and paint the interior outward toward the boundary. If the boundary is specified in a single color, the fill Algorithm precedes outward pixel by pixel until the boundary color is encountered. This method is called Boundary Fill Algorithm.

Boundary Fill Algorithm procedure accepts as input the coordinates of an interior point (x, y), a fill color and a boundary color. Starting with (x, y), neighboring boundary points are also tested. This process continues till all the pixels up to the boundary color for the area is tested.



Diagram



1	Fill method applied to 4-connuted area.	**2**	Fill method applied to 8-connuted area.

Recursive boundary fill Algorithm may not fill regions correctly if some interior pixels are already displayed in the fill color. This occurs because the Algorithm checks next point both for boundary color and for fill color.

The pixel positions are stored in form of a stack in the Algorithm.

Flood Fill Algorithm: Sometimes we want to fill in (or recolor) an area that is not defined within a single color boundary suppose we take an area which is bordered by several different color. We can paint such area by replacing a specified interior color instead of searching for a boundary color value. This approach is called flood fill Algorithm.

We start from interior paint (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color. If the area we want to paint has move than one interior color, we can first reassign pixel value so that all interior points have the same color.

Chapter 3

Q1. **Explain Translation in detail**

Ans A translation is applied on an object by changing its position along a straight line path from one coordinate location to another. A translation is implemented to an object by moving it along a straight-line path from one coordinate place to another. We render a two-dimensional point by adding translation distances, f, and t,, to the original coordinate point (x, y) to move the point to a new position (x ' , y') (Figure. 5-1)

$$x' = x + tx, \qquad y' = y + ty,$$

The translat~ond istance pair (t,, t,) is called a translation vector or shift vector.

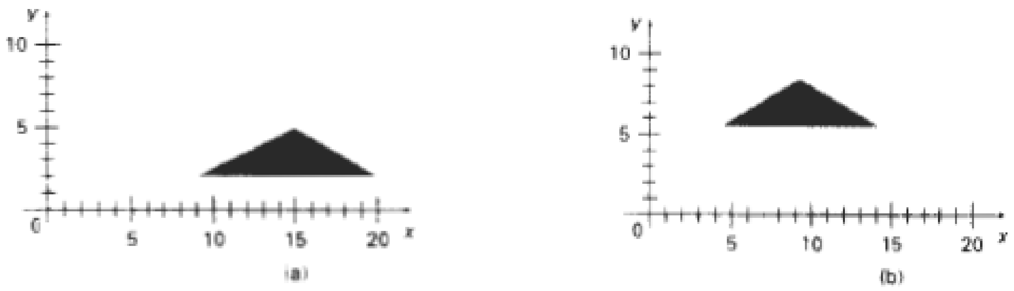
We can express the translation equations 5-1 as a single matrix equation by usng column vectors to represent coordinate positions and the translation vector:

$$\mathbf{P} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

This allows us to write the two-dimensional translation equations in the matrix form:

$$P` = P + T$$

At times matrix-transformation equations are articulated in terms of coordinate row vectors in place of column vectors. In this case, we would mark the matrix representations as P = [x y] and T = [tx, ty]. Since the column-vector representation intended for a dot is typical mathematical notation, and since a lot of graphics packages also use the column-vector illustration, we will follow this rule. Translation is a rigid-body transformation that moves objects with no deformation. That is, every dot on the object is translated by the identical amount. A straight Line segment is translated by applying the transformation equation 5-3 to each of the line endpoints and redrawing the line amid the new endpoint positions. Polygons are translated by adding the translation vector to the coordinate position of each vertex and regenerating the polygon by means of the new set of vertex coordinates and the present feature adjustments.. Figure 5-2 shows the application of a particular translation vector to move an object starting one position to another. Similar techniques are used to transform curved objects. To change the position of a circle or ellipse, we translate the center coordinates and redraw the figure in the new location. We translate other curves (for example, splines) by displacing the coordinate positions defining the objects, then we restructure the curve paths using the translated coordinate points.



Moving a polygon from the first position to the second one.

Q2. Explain Rotation in detail.

Ans 2-dimensional rotation is implemented on an object by moving it along a circular path in the **xy** plane. To carry out a rotation, we specify a rotation angle 0 and the position ***(x,y)*** of the rotation point (pivot point) about which the specified object is to be rotated (Figure. **5-3).** Positive values for the rotation angle represent anti clockwise rotations about the pivot point, as in Figure. 5-3, and negative values carry out object rotation in the clockwise direction. This transformation can also be expressed in terms of rotation about a rotation axis which is vertical to the **xy** plane and passes through the pivot point.

We first establish the transformation equations for rotation of a point position P when the pivot point is at the coordinate origin. The angular and coordinate associations of the original and transformed point positions are shown in Figure. 5-4. **In** this figure, **r** is the constant distance of the point from the origin, angle **φ** is the original angular position of the point from the horizontal, and t3 is the rotation angle. Using standard trigonometric identities, we can state the transformed coordinates in terms of angles 0 and ***θ***

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

The original coordinates of the point in polar coordinates are

$$x = r\cos\phi, \qquad y = r\sin\phi$$

Replacing expressions 5-5 into 5-4, we obtain the transformation equations for rotating a point at position ***(x,* y)** through an angle 9 about the origin:

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

With the column-vector representations 5-2 for coordinate positions, we can write the rotation equations in the matrix form:

$$\mathbf{P}' = \mathbf{R}\cdot\mathbf{P}$$

where the rotation matrix is

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

When coordinate positions are expressed as row vectors instead of column vectors, the matrix product in rotation equation 5-7 is transposed so that the transformed row coordinate vector [x' y'] is determined as

$$\mathbf{P}'^T = (\mathbf{R} \cdot \mathbf{P})^T$$
$$= \mathbf{P}^T \cdot \mathbf{R}^T$$

where **PT** = **[x y],** and the transpose RT of matrix R is obtained by interchanging rows and columns. For a rotation matrix, the transpose **is** obtained by simply changing the sign of the sine terms.

Rotation of a point about an arbitrary pivot position is illustrated in Figure. 5-5. Using trigonometric associations in this figure, we can simplify Eqs. 5-6 to obtain the transformation equations for carrying out rotation of a point about any specified mutation position **(x_r,y_r):**

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

These general rotation equations differ from Eqs. 5-6 by the inclusion of additive terms, as well as the multiplicative factors on the coordinate values. Thus, the above matrix expression could be modified to include : pivot coordinates by matrix addition of a column vector whose elements contain the additive (translational) terms In Eqs. 5-9. There are better ways, however, to formulate such matrix equations, and we discuss in Section **5-2** a more consistent scheme for representing the transformation equations

As with translations, rotations are rigid-body transformations that move objects without deformation. Every point on an object is rotated through the same anglc. A straight line segment is rotated by applying the rotation equations 5-9 to each ot tht' line endpoints and redrawing the line between the new endpoint positions. Polygons are rotated by displacing each vertex through the specified rotation angle and regenerating the polygon using the new vertices. Curved lines arc rotatcd by repositioning the defining p ~ r ~atnsd redrawing the curves. A circle k>r ,117 ellipse, for instance, can be rotated about a noncentral axis by movins the center position through the arc that subtcncs thc sprcified rotation angle. An ellipse can be rotated about its center coordinates by rotating the major and minor **axes**

Chapter 4
Two Dimensional Concepts Clipping Algorithm

Q.1 Explain Clipping. Also Explain How can a point be clipped?

Ans. : Clipping is defined as the identification of the objects of the view which are outside the clipping region and which can be removed or clipped from the viewing window. Any procedure that identifies those portion of a picture that are either inside or outside of a specified region of space is referred to as a clipping Algorithm or clipping.

The region against which the object needs to be clipped is known as clip window. We will assume here that clip window is a rectangular window but it might be a polygon shaped as well and can have boundaries in the curved form as well. Hence the objects which are not inside and are outside the rectangular clip window are hence discarded. The various clipping algorithms which are involved in the process of clipping are as follows:

- Point Clipping
- Line Clipping
- Polygon clipping
- Text Clipping
- Curve Clipping

Here are a few examples of the application of the clipping concepts which are as follows:

(1) Creating objects using solid-modeling procedures.

(2) Drawing and painting operations.

(3) Identifying visible surface in three dimensional views.

(4) Antialising line segments or object boundaries.

(5) Extracting parts of defined scene for viewing.

(6) Displaying multi window environment.

Q2. Explain Point Clipping.

Ans Assuming that a point P(x,y) is to be displayed on the screen we need to determine if this point lies within the clip window or not. Assuring that clip window is a rectangle in standard Position, we save a point P = (x, y) for display if following inequalities are satisfied. And we have to compare the point coordinates with window coordinates.

$$xw_{min} \le x \le xw_{max}$$

$$yw_{min} \le y \le yw_{max}$$

then the point (x,y) lies within the view window and can be displayed, otherwise it needs to be discarded. Where the edges of clip window (xw_{min}, xw_{max}, yw_{min}, yw_{max}) can be either coordinate window boundaries or view port boundaries. If any one of these inequalities is not satisfied the point is clipped.

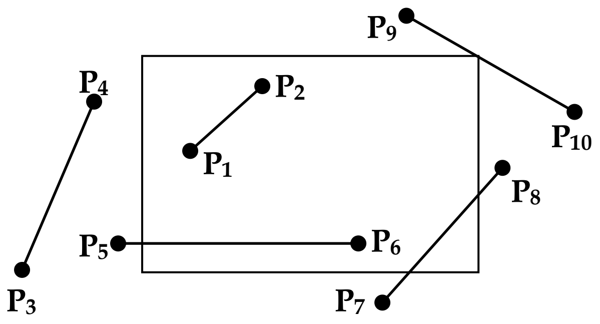
Application of Point Clipping: Point clipping can be applied to scenes involving explosions or sea foam that are modeled with particles (points) distributed in some region of the scene.

Q. 3 Explain Line Clipping? Also Explain Cohen Sutherland Method of Line Clipping.

Ans. A line consists of a sequence of number of points arranged between the two end points. Here we just have to consider only the endpoints for clipping purpose, and we would not consider the points between the endpoints. A line clipping procedure involves several parts First, we can test a given line segment to determine whether it lies completely inside the clipping window.

If the two endpoints of a line fall within the clip window, it is accepted. And if in case one of the line end point falls inside and the other end point goes out of the clip window, we need to make calculation of the intersection of the line with the edges of the rectangular window. If it does not, we try to determine whether it lies completely outside the window.

Finally if we can not identify a line as completely inside or completely outside we must perform intersection calculation with one or more clipping boundaries. We process lines through inside-outside tests by checking the line end points.



- A line with both end points outside any one of the clip boundaries (line P_3, P_4 in Figure.) is outside the window.
- A line with both end points inside all clipping boundaries such as line form P_1 to P_2 is saved.
- And all the other lines that cross one or more clipping boundaries and may require calculation of multiple intersection point.
- For a line segment with end points (x_1, y_1) and (x_2, y_2) and one or both end points outside clipping rectangle, the parametric representation.Could be used to determine values of parameter u for intersections with the clipping boundary coordinates.

$$x = x_1 + u(x_2 - x_1)$$

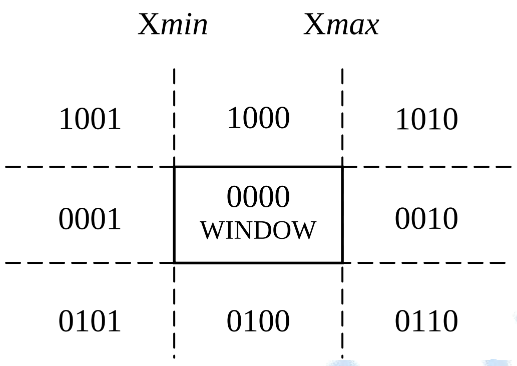
$$y = y_1 + u(y_2 - y_1), \quad 0 \le u \le 1$$

If the value of u for an intersection with a rectangle boundary edge is outside the range 0 to 1, the line does not enter the interior of the window at that boundary. If the value of u is within the range from 0 to 1, the line segment does indeed cross into the clipping area.

Cohen - Sutherland line Clipping

This algorithm also reduces calculations by the identification of the lines which can be trivially discarded or accepted. And this can be done by making comparison with the endpoints with the window coordinates (Xmin, Ymin) and (Xmax, Ymax). This is one of the oldest and most popular line clipping procedures. Generally, the method speeds up the processing of line segments by performing initial test that reduces the number of intersections that must be calculated.

Every line end point in a picture is assigned a four digit binary code called, a region code, that identifies the location of the points relative to the boundaries of the clipping rectangle. For this purpose we assign 4-digit binary code to each endpoint of the line. As shown in the following diagram, we have extended the window to get a plane of nine regions.



Code for inside the window region is 0000. Each regions is defined or represented by 1 bit. First bit from left i.e., MSB is for the region above the top edge. If this bit is 1, it means point is above the top edge or $y>y_{max.}$ Each bit position in the region code is used to indicate one of the four relative coordinate positions of the point with respect to the clip window: to the left, right, top and bottom.

Second bit from left is for the region below the bottom

By numbering the bit position in the region code as 1 through 4 right to left, the coordinate regions can be correlated with the bit positions as :

bit 1 : left ; bit 2 : right ; bit 3 : below ; bit 4 : above

A value of 1 in any bit position indicates that point is in that relative position otherwise the bit position is set to 0.

- Now here bit values in the region are determined by comparing end point coordinate values (x, y) to the clip boundaries.

  Bit 1 is set to 1 if $x < xw_{min}$

  Bit 2 is set to 1 it $xw_{max} < x$

Now Bit 1 sign bit of $x - xw_{min}$

Bit 2 sign bit of $xw_{max} - x$

Bit 3 sign bit of $y - yw_{min}$

Bit 4 sing bit of $yw_{max} - y$

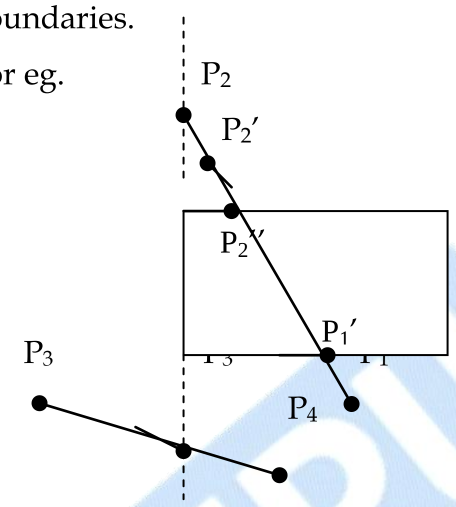
(1) Any lines that are completely inside the clip window have a region code 0000 for both end points few points to be kept in mind while checking.

(2) Any lines that have 1 in same bit position for both end points are considered to be completely outside.

(3) Now here we use AND operation with both region codes and if result is not 0000 then line is completely outside.

Now for lines that cannot be identified as completely inside or completely outside the window by this test are checked by intersection with window boundaries.

For eg.



We check P_1 against left right we find it is below window. We find intersection point P_1' and then discard line section P_1 to P_1' & Now we have P_1 to P_2. Now we take P_2 we find it in left position outside window then we take an intersection point P_2''. But we find it outside window then we again calculate final intersection point P_2''. Now we discard line P_2 to P_2''. We finally get a line P_2'' to P_1' inside window similarly check for line P_3 to P_4.

For end points (x_1, y_1) (x_2, y_2) y–coordinate with vertical boundary can be calculated as

$$y = y_1 + m(x - x_1) \text{ where x is set to } xw_{min} \text{ to } xw_{max}$$

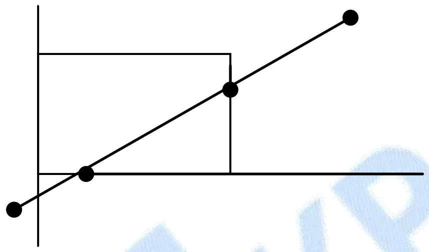
Q.4 Discuss the Cyrus Beck Algorithm for Clipping in a Polygon Window.

Ans. Cyrus Beck Technique : Cyrus Beck Technique can be used to clip a 2–D line against a rectangle or 3–D line against an arbitrary convex polyhedron in 3-d space.

Liang Barsky Later developed a more efficient parametric line clipping Algorithm. Now here we follow Cyrus Beck development to introduce parametric clipping now in parametric representation of line Algorithm has a parameter t representation of the line segment for the point at which that segment intersects the infinite line on which the clip edge lies, Because all clip edges are in general intersected by the line, four values of t are calculated. Then a series of comparison are used to check which out of four values of (t) correspond to actual intersection, only then are the (x, y) values of two or one actual intersection calculated.

Advantage of this on Cohen Sutherland :

(1) It saves time because it avoids the repetitive looping needed to clip to multiple clip rectangle edge.



(2) Calculation in 3D space move complicated than 1-D

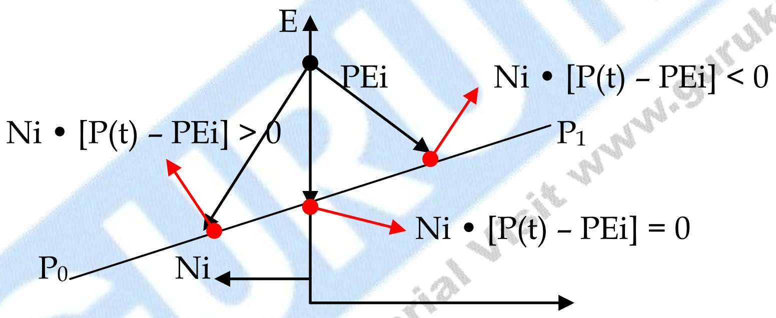


Figure.1

Cyrus Beck Algorithm is based on the following formulation the intersection between 2-lines as in Figure.1 & shows a single edge Ei of clip rectangle and that edge's outward normal Ni (i.e. outward of clip rectangle).

Either the edge or the line segment has to be extended in order to find intersection point.

Now before this line is represented parametrically as :

$$P(t) = P_0 + (P_1 - P_0)t$$

Where $t = 0$ at P_0

$t = 1$ at P_1

Now pick an arbitrary point PEi on edge Ei Now consider three vectors P(t) – PEi to three designated points on P_0 to P_1.

Now we will determine the endpoints of the line on the inside or outside half plane of the edge.

Now we can determine in which region the points lie by looking at the dot product $Ni \bullet [P(t) - PEi]$.

$$Ni \bullet [P(t) - PEi] \begin{cases} \rightarrow \text{Negative if point is inside half plane} \\ \rightarrow \text{Zero if point is on the line containing the edge} \\ \rightarrow \text{Positive if point lies outside half plane.} \end{cases}$$

Now solve for value of t at the intersection P_0P_1 with edge.

$$Ni \bullet [P(t) - PEi] = 0$$

Substitute for value P(t)

$$Ni \bullet [P_0 + (P_1 - P_0)\, t - PEi] = 0$$

Now distribute the product

$$Ni \bullet [P_0 - PEi] + Ni\, [P_1 - P_0]\, t = 0$$

Let $D = (P_1 - P_0)$ be the from P_0 to P_1 and solve for t

$$t = \frac{Ni \bullet [P_0 - PEi]}{-Ni \bullet D} \qquad \_\_\_\,(1)$$

This gives a valid value of t only if denominator of the expression is non-zero.

<u>Note</u> : Cyrus Beck use inward Normal Ni, but we prefer to use outward normal for consistency with plane normal in 3d which is outward. Our formulation differs only in testing of a sign.

(1) For this condition of denominator to be non-zero. We check the following :

$Ni \neq 0$ (i. e Normal should not be Zero)

$D \neq 0$ (i.e $P_1 \neq P_0$)

Now hence $Ni \bullet D \neq 0$ (i.e. edge Ei and line between P_0 to P_1 are not parallel. If they were parallel then there cannot be any intersection point.

Now eq.(1) can be used to find the intersection between the line & the edge.

Now similarly determine normal & an arbitrary PEi say an end of edge for each clip edge and then find four arbitrary points for the "t".

(2) Next step after finding out the four values for "t" is to find out which value corresponds to internal intersection of line segment.

(i) Now 1st only value of t outside interval [0, 1] can be discarded since it lies outside P_0P_1.

(ii) Next is to determine whether the intersection lies on the clip boundary.

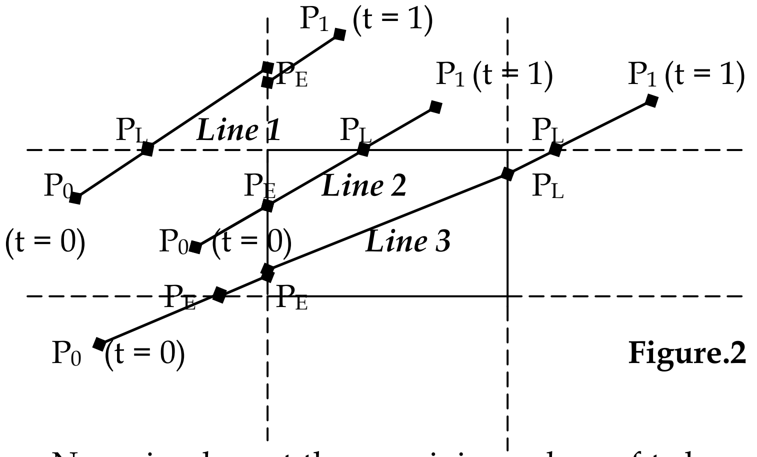


Figure.2

Now simply sort the remaining values of t choose the intermediate value of t for intersection points as shown in Figure.(2) for case of line 1.

Now the question arises how line 1 is different from line 2 & line 3.

In line 2 no portion of the line lies on the clip boundary and so the intermediate value of t correspond to points not on the clip boundary.

In line 3 we have to see which points are on the clip boundary.

Now here intersections are classified as :

PE $\longrightarrow$ Potential entering

PL $\longrightarrow$ Potential leaving

(i) Now if moving from P_0 to P_1 and the line causes to cross a particular edge t enter the edge inside half plane the intersection is PE.

(ii) Now if it causes to leave the inside plane then it is denoted as P_L.

Formally it can be checked by calculating angle between P0, P1 and Ni.

$Ni \bullet D < 0$ $\longrightarrow$ PE (angle is greater than 90^o)

$Ni \bullet D > 0$ $\longrightarrow$ PL (angle is less than 90^o)

(3) Final steps are to select a pair (PE, PL) that defines the clipped line.

Now we suggest that PE intersection with largest t value which we call t_E and the PL intersection with smallest t value t_L. Now the intersection line segment is then defined by range (t_E, t_L). But this was in case of an infinite line. But we want the range for P_0 to P_1 line. Now we set this as :

$t = 0$ is a upper bound for t_E

$t = 1$ is a upper bound for t_L

But if $t_E > t_L$

Now this is case for line 2, no portion of P_0P_1 is in clip rectangle so whole line is discarded. Now t_E and t_L that corresponds to actual intersection is used to find value of x & y coordinates.

Table 1 : For Calculation of Parametric Values.

Clip edge i	**Normal Ni**	**PEi**	**P_0 – PEi**	$t = \frac{Ni.(P_0 - PEi)}{-Ni.D}$
Left : $x = x_{min}$	(– 1, 0)	(x_{min}, y)	$(x_0 - x_{min}, y_0 - y)$	$\frac{-(x_0 - x_{min})}{(x - x_0)}$
right : $x = x_{max}$	(1, 0)	(x_{max}, y)	$(x_0 - x_{max}, y_0 - y)$	$\frac{(x_0 - x_{max})}{-(x_1 - x_0)}$
bottom : $y = y_{min}$	(0, – 1)	(x, y_{min})	$(x_0 - x, y_0 - y_{min})$	$\frac{-(y_0 - y_{min})}{(y_1 - y_0)}$
top : $y = y_{max}$	(0, 1)	(x, y_{max})	$(x_0 - x, y_0 - y_{max})$	$\frac{(y_0 - y_{max})}{-(y_1 - y_0)}$

□ □ □

Chapter 5

Three dimensional concepts

Q.1 What are Bezier Curves and Surfaces. Write the properties of Bezier Curves.

Ans.: This spline approximation method was developed by French Engineer Pierre Bezier for use in the design of Renault automobile bodies Bezier splines have a number of properties that make them highly useful and convenient for curves and surface design. They are easy to implement. Bezier splines are widely available in CAD systems.

Bezier Curves : A Bezier curve can be fitted to any number of control points. The number of control point to be approximated and their relative position determine the degree of Bezier polynomial. As with interpolation splines, a Bezier curve can be specified with boundary conditions, with characterizing matrix or with blending functions.

Suppose we have (n + 1) control point positions: P_k – (x_k, y_k, z_k,) with K varying from 0 to n. These coordinate points can be blended to produce the following position vector P(u), which describes the path of an approximating Bezier Polynomial function between P_0 & P_n.

$$P(u) = \sum_{K=0}^{n} P_k \, BEZ_{k,n}(u) \qquad 0 \le u \le 1 \qquad \_\_\_ (1)$$

The Bezier blending functions $BEZ_{k,n}$ (u) are the Bernstein polynomial.

$$BEZ_{k,n}(u) = C(n,k)\, u^k (1-u)^{n-k} \qquad \_\_\_ (2)$$

Where $C_{(n,k)}$ are the binomial coefficients

$$C_{(n,k)} = \frac{\underline{n|}}{\underline{k|}\,\underline{n-k|}} \qquad \_\_\_ (3)$$

Equivalently we can define Bezier blending function with recursive calculation :

$$BEZ_{k,n}(u) = (1-u)\, BEZ_{k,n-1}(u) + u\, BEZ_{k-1,n-1}(u) \qquad n > k \ge 1 \qquad \_\_\_ (4)$$

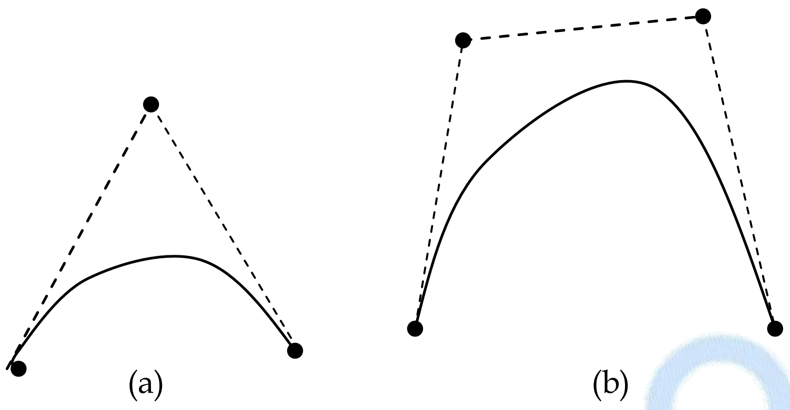
Three parametric equations for individual curve coordinates :

$$x(u) = \sum_{K=0}^{n} x_k \, BEZ_{k,n}(u) \qquad y(u) = \sum_{K=0}^{n} y_k \, BEZ_{k,n}(u) \qquad z(u) = \sum_{K=0}^{n} z_k \, BEZ_{k,n}(u)$$

A Bezier Curve is a polynomial of degree one less than the number of control points used.

Three points generates Parabola.

Four points generates Cubic Curve.



Bezier Curves generated from three or four control points. Dashed lines connect the control point positions.

Properties of Bezier Curve :

(1) It always passes through the first and last control points. That is boundary condition at two ends of the curve are :

For two end points $\rightarrow P(0) = P_0$, $\rightarrow P(1) = P_n$ ___ (6)

Now value for the first derivative of a Bezier Curve at the end points can be calculated from control point coordinates as :

$P'(0) = -nP_0 + nP_1$ [First derivative for the

$P'(1) = -nP_{n-1} + nP_n$ first & last 2 end points] ___ (7)

$\therefore$ Slope of the curve at the beginning of curve is calculated by joining the beginning 2- points.

Slope of the curve at the end of curve is calculated by joining the 2 – end points.

Similarly second derivative of Bezier Curve at the end point are : -

$P''(0) = n(n-1)[(P_2 - P_1) - (P_1 - P_0)]$

$P''(1) = n(n-1)[(P_{n-2} - P_{n-1}) - (P_{n-1} - P_n)]$ ___ (8)

(2) Another property of Bezier Curve is, it lies within the convex hull. Therefore this follows the properties of Bezier blending function. That is they are all positive & their sum is always 1.

$\therefore \quad \sum^{n} BEZ_{k,n}(u) = 1$ ___ (9)

K = 0

So that any curve position is simply weighted sum of the control points positions.

The convex hull also ensures that polynomial smoothly follows the control points without erratic oscillations.

Bezier Surfaces : Two sets of orthogonal Bezier Curves can be used to design an object surface by specifying by an input mesh of control points.

Blending function are :

$$P(u, v) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,k}\, BEZ_{j,m}(v)\, BEZ_{k,n}(u) \qquad \_\,\_\,\_\ (10)$$

With $P_{j,k}$ specifying location of (m + 1) by (n + 1) control points.

Bezier surfaces have same properties as Bezier Curves. Zero order continuity is obtained by matching control points at the boundary.

First order continuity is obtained by choosing control points along a straight line across the boundary & by maintaining a constant ratio of collinear line segments for each set of specified control points across section boundaries.

NOTE : In context to Cubic Bezier Curve. Derive Bezier Matrix.

$$\begin{bmatrix} 1 & 0 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Write the same answer as the above one.

Q.2 What are B-Spline Line, Curves and Surfaces? Write the properties of B-Spline Curves?

Ans.: These are most widely used class of approximating splines B-splines have two advantage over Bezier splines.

(1) The degree of a B – spline polynomial can be set independently of the number of control points (with certain limitations).

(2) B – spline allow local control over the shape of a spline curve or surface.

The trade of off is that B – splines are move complex than Bezier splines.

B – spline Curves : Blending function for B – spline curve is :

n

$$P(u) = \sum_{K=0} P_k B_{k,d}(u) \qquad u_{min} \leq u \leq u_{max}$$
$$2 \leq d \leq n+1$$

Where the P_k are an input set of (n + 1) control points. There are several differences between this B-spline formulation and that for Bezier splines. The range of parameter u now depends on how we choose the B – spline parameters. And the B – spline blending functions $B_{k,d}$ are polynomials of degree d – 1, where parameter d can be chosen to be any integer. Value in the range from 2 up to the number of control points , n + 1, Local control for B – splines is achieved by defining the blending functions over subintervals of the total range of u.

Blending function for B – spline curves are defined by Cox – de Boor recursion formulas :

$$B_{k,d}(u) \; \begin{cases} 1 & \text{if } u_k \leq u \leq u_{k+1} \\ = & \\ 0 & \text{otherwise} \end{cases}$$

$$B_{k,d} \; u = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1} \; u + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k=1,d-1} \; u$$

Where each blending function is defined over d subintervals of the total range of u. values for u_{min} and u_{max} then depends on the number of control points we select, we can increase the number of values in the knot vector to aid in curve design.

Properties of B – spline Curve :

(1) The polynomial curve has degree (d – 1) and C^{d-2} continuity over the range of u.

(2) For (n + 1) control points, the curve is described with (n + 1) blending function.

(3) Each blending function $B_{k,d}$ is defined over d subintervals of the total range of u, starting at knot value u_k.

(4) The range of parameter u is divided into (n + d) subintervals by the (n + d + 1) values specified in knot vector.

(5) With knot values labeled as { u_0 , u_1 , - - - , u_{n+d}} the resulting B – spline over is defined only in the interval from knot value u_{d-1} up to the knot value u_{n+1}.

(6) Each section of the spline curve (between two successive knot values) is influenced by d – control points.

(7) Any one control points can affect the shape of almost d curve section.

For any vale of u in the interval from knot value u_{d-1} to u_{n-1} the sum over all basis function is 1.

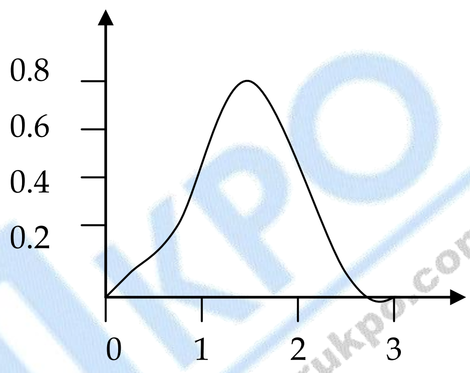
$$\sum_{K=0}^{n} B_{k,d}(u) = 1$$

We need to specify the knot values to obtain the blending function using recurrence relation.

Classification of B – splines according to the knot vectors :

Uniform, Periodic B – splines : When spacing between knot values is constant. The resulting curve is called a uniform B – spline.

For e.g. : { - 1.5, -1.0, -0.5, 0.0}



Periodic B - spline blending functions for n = d = 3 and a uniform, integer knot vector.

Uniform B – splines have periodic blending functions shifted version of previous function :

$$B_{k,d}(u) = B_{k+1,d}(u + \Delta u) = B_{k+2,d}(u + 2\Delta u)$$

Cubic Periodic B – spline : Periodic splines are particularly useful for generating certain closed Curves. If any three consecutive control points are identical, the curve passes through that coordinate position.

The boundary condition for periodic cubic B – spline with four consecutive control points labeled P_0, P_1, P_2, P_3 are.

$$P(0) = \frac{1}{6}(P_0 + 4P_1 + P_2)$$

$$P(1) = \frac{1}{6}(P_1 + 4P_2 + P_3)$$

$$P'(0) = \frac{1}{2}(P_2 - P_0)$$

$$P'(1) = \frac{1}{2}(P_3 - P_1)$$

Matrix formulation for periodic cubic polynomial is :-

$$P(u) = [u^3\ u^2\ u\ 1] \cdot M_B \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

where B – spline Matrix is

$$M_B = \frac{1}{6} \begin{bmatrix} 1 & 0 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Open Uniform B – spline : This class of B – spline is a cross between uniform B – spline and non uniform B – spline. For open uniform B – spline or simply open B – spline, the Knot spacing is uniform except at the ends where Knot values are repeated d – times.

For eg.: [0, 0, 1, 2, 3, 3] for d = 2 and n = 3

[0, 0, 0, 0, 1, 2, 2, 2, 2] for d = 4 and n = 4

Non uniform B – splines : With non uniform B – spline, we can choose multiple Knot values and unequal spacing between the Knot values.

For eg.: [0, 1, 2, 3, 3, 4]

[0, 2, 2, 3, 3, 6]

Non uniform B – spline provides increased flexibility in controlling a curve shape.

We can obtain the blending function for a non uniform B – spline using methods similar to those discussed for uniform and open B – spline.

B – spline Surfaces : Cartesian product of B – spline blending function in the form

$$P(u, v) = \sum_{K_1=0}^{n_1} \sum_{K_2=0}^{n_2} P_{K_1, K_2}\, B_{K_1, d_1}(u)\, B_{K_2, d_2}(v)$$

Where vector values for $P_{k1,\ k2}$ specify position of the $(n_1 + 1)$ by $(n_2 + 1)$ control points.

Q. 3 What is Hermite Interpolation?

Ans. A hermite spline is an interpolating piecewise cubic polynomial with a specified tangent at each control point. Hermite splines can be adjusted locally because each curve section is only dependent on its end point constraints. If P(u) represents a parametric cubic point function for the curve section between control points P_k and P_{k+1}.

Boundary conditions that define this Hermite curve section are :

$$\left.\begin{aligned} P(0) &= P_k \\ P(1) &= P_{k+1} \\ P'(0) &= DP_k \\ P'(1) &= DP_{k+1} \end{aligned}\right\} \qquad \text{\_\_\_ (1)}$$

With DP_k and DP_{k+1} specifying values for the parametric derivatives (slope of the curve) at control points P_k and P_{k+1} respectively.

Vector equivalent equation :

$$P(u) = au^3 + bu^2 + cu + d \qquad 0 \le u \le 1 \qquad \text{\_\_\_ (2)}$$

Where x component of P is $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$ and similarly for y and z components. The Matrix is

$$P(u) = [u^3 \;\; u^2 \;\; u \;\; 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad \text{\_\_\_ (3)}$$

Derivative of the point function as :

$$P'(u) = [3u^2 \;\; 2u \;\; 1 \;\; 0] \begin{bmatrix} a \\ b \\ C \\ d \end{bmatrix} \qquad \text{\_\_\_ (4)}$$

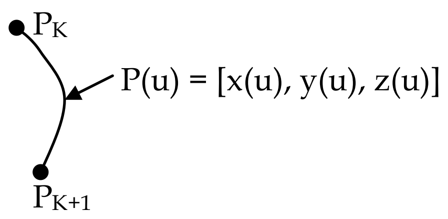


Figure.(1) Parametric point function P(u) for a Hermite curve section between control point P_k and P_{k+1}.

Now we express the hermite boundary condition in matrix form.

$$\begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \_\_\_ (5)$$

Solving this equation for the polynomial coefficients : -

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} P_k \\ P_{k+1} \\ D_{Pk} \\ DP_{k+1} \end{bmatrix} = M_H \begin{bmatrix} P_k \\ P_{k+1} \\ D_{Pk} \\ DP_{k+1} \end{bmatrix} \quad \_\_\_ (6)$$

Where M_H, the Hermite Matrix is the inverse of the boundary constraint Matrix.

Equation (3) can thus be written in terms of boundary condition as :

$$P(u) = [u^3 \quad u^2 \quad u \quad 1]\, M_H \begin{bmatrix} P_k \\ P_{k+1} \\ DP_k \\ DP_{k+1} \end{bmatrix}$$

Now we obtain the blending function by carrying out Matrix Multiplication :

$P(u) = P_k\,(2u^3 - 3u^2 + 1) + P_{k+1}\,(-2u^3 + 3u^2) + DP_k\,(u^3 - 2u^2 + u) + DP_{k+1}\,(u^3 - u^2)$

$= P_k\, H_0\,(u) + P_{k+1}\, H_1\,(u) + DP_k\, H_2\,(u) + DP_{k+1}\, H_3\,(u)$

$H_k(u)$ are refered to as blending function for K = 0, 1, 2, 3.

Q.4 Differentiate between Hermite Curve and B - spline Curve?

Ans.:

S.No.	**Hermite Curve**	**B – spline Curve**
1.	This spline is interpolation spline with a specified tangent at each control point.	This is a class of approximating spline.
2.	It does not require any input	It requires many I/P values

	values for curve slope or other geometric information, in addition to control point coordinates.	like Knot vector, range of Parameter u etc.

Q.5 Explain Zero Order, First Order, and Second Order Continuity in Curve Blending?

Ans.: To ensure a smooth transition form one section of a piecewise curve to the next. We can suppose various continuity conditions, at the connection point. Each section of a spline is described with a set of parametric coordinate function of the form :

$$x = x(u),\ y = y(u),\ z = z(u), \qquad u_1 \le u \le u2 \qquad \_\_\_ (1)$$

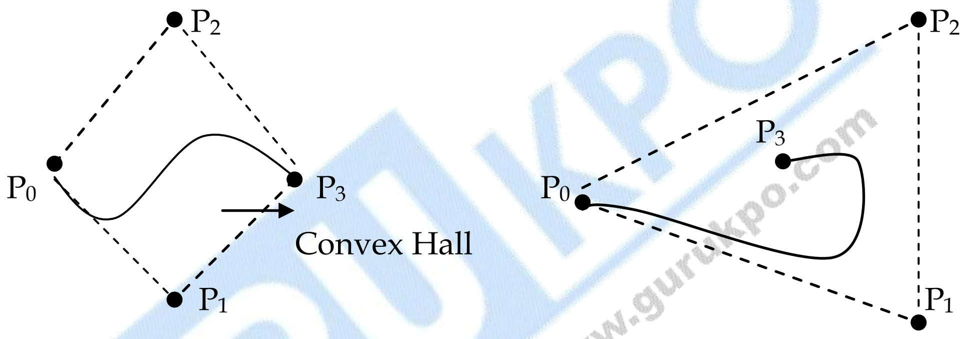


Figure.(1) convex hull shapes for two sets of control points.

Parametric Continuity : We set parametric continuity by matching the parametric derivative of adjoining section at their common boundary.

Zero – Order Parametric Continuity : Described as C^0 continuity means simply that curves meet. That is the values of x, y, z evaluated at u_2 for first curve section is equal to values of x, y, z evaluated at u_1 for the next curve section.

First – Order Parametric Continuity : Refered as C^1 continuity means that the first Parametric derivatives (tangent lines) of the coordinate functions in eq.(1) for two successive curve sections are equal at their joining point.

Second - Order Parametric Continuity : Described as C^2 continuity means that both first and second parametric derivatives of the two curve sections are the same at the intersection.

With second order continuity, the rates of change of the tangent vector for connecting sections are equal at their intersection. Thus the tangent line transitions smoothly from one section of the curve to the Next like in Figure.(2).

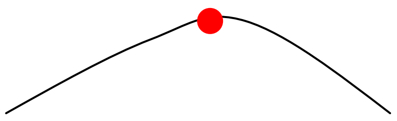


Figure.(2) : Showing second order continuity.

With First order continuity, the change of tangent vectors for two sections can be quite different, so that the general shape of the two adjacent section can change abruptly.

Applications :

(1) First order continuity is often sufficient for digitizing drawings and some design applications.

(2) Second order continuity is useful for setting up animation paths for camera motion and for many precision CAD Requirements.

Geometric Continuity Conditions : An alternate method for joining two successive curve sections is to specify conditions for geometric continuity. In this case, we only require parametric derivatives of the two sections to be **proportional** to each other at their common boundaries instead of **equal** to each other.

Zero Order Continuity : Described as G^0 continuity, is same as zero order parametric continuity. That is two curve section must have the same coordinate position at the boundary point.

First Order Geometric Continuity: - Described as G^1 continuity means those parametric first derivatives are proportional at the intersection of two successive sections.

Suppose if we denote the parametric position on the curve as P(u) the direction of the tangent vector P'(u), but not necessarily its magnitude will be same of two successive curve sections at their joining point under G^1 Continuity.

Second Order Geometric Continuity : Described as G^2 continuity means that both the first and second parametric derivative of the two curve sections are proportional at their boundary under G^2 continuity, curative of two curve section will match at the joining position.

A curve generated with geometric continuity conditions is similar to one generated with parametric continuity but with slight difference in curve shape.

Chapter 6

Computer Animations

Q1. Explain the Animation Sequence or rules for Animation

Ans

Animation sequence in general is designed in the following steps.

1. Storyboard layout
2. Object definitions.
3. Key-frame specifications
4. Generation of in-between frames.

This approach of carrying out animations is applied to any other applications as well, although some applications are exceptional cases and do not follow this sequence.

For frame-by-frame animation, every frame of the display or scene is generated separately and stored. Later, the frame recording can be done and they might be displayed consecutively in terms of movie.

The outline of the action is storyboard. This explains the motion sequence. The storyboard consists of a set of rough structures or it could be a list of the basic ideas for the motion.

For each participant in the action, an object definition is given. Objects are described in terms of basic shapes the examples of which are splines or polygons. The related movement associated with the objects are specified along with the shapes.

A key frame in animation can be defined as a detailed drawing of the scene at a certain time in the animation sequence. Each object is positioned according to the time for that frame, within each key frame. Some key frames are selected at extreme positions and the others are placed so that the time interval between two consecutive key frames is not large. Greater number of key frames are specified for smooth motions than for slow and varying motion.

And the intermediate frames between the key frames are In-betweens. And the Media that we use determines the number of In-betweens which are required to display the animation. A Film needs 24 frames per second, and graphics terminals are refreshed at the rate of 30 to 60 frames per second. Depending on the speed specified for the motion, some key frames are duplicated. For a one minutes film sequence with no duplication, we would require 288 key frames. We place the key frames a bit distant if the motion is not too complicated.

A number of other tasks may be carried out depending upon the application requirement for example synchronization of a sound track.

Q2. Explain the General Computer-Animation Functions

Ans In the development of an animation sequence some steps well suit to computer solution and these operations include camera motions, object manipulations and rendering and the generation of in-betweens. Animation packages provide special functions for carrying out the animation and processing individual objects.

Animation package also provides function to store and manage the object database. And the shapes of the objects and the parameters which are associated to them are stored and updated in the database. Other functions are implemented and carry out motion generation and object rendering.

Q 3 Explain Raster Animation

Ans Raster systems have the capabilities of generating real time animation in limited applications using raster operations. A simple method to carry out translation in the xy plain is to transfer a rectangular block of pixel values from one location to another. For the rotation of a block of pixels, we need to determine percentage of area covered has to be done for that pixel that overlaps the rotated block. Raster operation sequence can be executed to carry out real time animation of either 2 D or 3 D. objects as long as the plane (projection). In this case no viewing or visible surface algorithms need to be involked.

Using coor-table transformations we can animate objects along two dimensional motion paths. The object here at successive positions is predefined along with the motion path. And the color table entries are updated with the successive blocks of pixel values. We set the pixels at the first position of the object to “on” values, and we set the pixels at the other object positions to the background color. The animation is attained by varying the color table values so that the object is “on” at successively along the animation path.

Q 4 Explain Key-Frame Systems

Ans We generate each set of in-betweens from the specification of two (or more) key frames. Here, motion paths can be defined with a kinematic description in terms of set of spline curves or the motion paths.

For more complex scenes, frames can be separated into individual components or objects cols cells. We can interpolate the positions of individual objects for any given animation paths. Shapes of complex objects may change over time examples of which are clothes, evolving shapes, exploding or disintegrating objects and this also includes transformation of one object into another.

Multiple Choice Questions

1) The smallest addressable screen element. Is called?

A. Pixel
B. Graph
C. voltage level
D. color information

2) Pixel on the graphics display represents?

A. mathematical point
B. a region which theoretically can contain an infinite number of points
C. voltage values
D. picture

3) The process of determining the appropriate pixels for representing picture or graphics object is known as?

A. rasterization
B. animation
C. both a and b
D. representation

4) The relationship among the data and objects which are stored in the database called application database, and referred by the?

A. Application programs
B. application model
C. graphics display
D. both a and b

5) Selective or part erasing of screen is not possible in?

A. raster scan display
B. vector scan display
C. DVST
D. both a and b

6) Display processor is also called a?

A. graphics controller
B. display coprocessor
C. both a and b
D. out put device for graphics

7) Graphics software acts as a very powerful tool to create?

A. Images
B. animated pictures
C. both a and b
D. system interface

8) The purpose of display processor is __from the graphics routine task?

A. to free the CPU
B. To free the secondary memory
C. to free the main memory
D. Both a & c

9) random-scan monitors typically offer __color choices?

A. Only a few
B. wide range
C. just 1 or 2
D. moderate range of

10) RGB system needs __of storage for the frame buffer?

A. 100 megabytes
B. 10 megabytes
C. 3 megabytes
D. 2 Gb

11) The SRGP package provides the __to wide the variety of display devices?

A. interface
B. connection
C. link
D. way

12) Length of shift register in bits is equal to __?

A. one word
B. A single scan lines
C. number of scan lines
D. One bit

13) Display controller is not required for __?

A. display system with only frame buffer
B. Display system with display controller
C. both a and b
D. display system with color

14) The basic principle of Bresenham`s line algorithm is__?

A. to select the optimum raster locations to represent a straight line
B. to select either Δx or Δy, whichever is larger, is chosen as one raster unit
C. we find on which side of the line the midpoint lies
D. both a and b

15) The midpoint circle drawing algorithm also uses the __of the circle to generate?

A. two-way symmetry
B. four-way symmetry
C. eight-way symmetry
D. both a & b

16) a polygon in which the line segment joining any 2 points within the polygon lies completely inside the polygon is called__?

A. convex polygon
B. concave polygon
C. both a and b
D. both a and b

17) A polygon in which the line segment joining any 2 points within the polygon may not lie completely inside the polygon is called __?

A. convex polygon
B. concave polygon
C. both a and b
D. Hexagon

18) Rectangular patterns are sometimes referred as__?

A. tiling patterns
B. Filling patterns
C. coloring patterns
D. both a and b

19) Replicating pixels is one of the methods of__?

A. Thick primitives
B. Thin primitives
C. stylish primitives
D. both a and b

20) Line produced by moving pen is __ at the end points than the line produced by the pixel replication?

A. thin

B. straight
C. thicker
D. both a and b

21) The process of selecting and viewing the picture with different views is called__?
A. Windowing
B. clipping
C. projecting
D. both a and b

22) The process which divides each element of the picture into its visible and invisible portions, allowing the invisible portion to be discarded is called__?
A. clipping
B. Windowing
C. both a and b
D. Projecting

23) The region against which an object is to be clipped is called__?
A. clip window or clipping window
B. Drawing window
C. image window
D. both b and c

24) The region code 0000 represents the__?
A. viewing window
B. left clipping window
C. right clipping window
D. bottom clipping window

25) A method used to test lines for total clipping is equivalent to the__?
A. logical XOR operator
B. logical OR operator
C. logical AND operator
D. both a and b

26) A process of changing the position of an object in a straight line path from one coordinate location to another is called__?
A. Translation
B. rotation
C. motion
D. both b and c

27) Data hazards occur when__?

A. Greater performance loss
B. Pipeline changes the order of read/write accesses to operands
C. some functional unit is not fully pipelined
D. machine size is limited

28) A two dimensional rotation is applied to an object by repositioning it along a?

A. circular path in the x-y plane
B. straight path in the x-y plane
C. diagonals path in the x-y plane
D. upward in the x-y plane

29) A scaling transformation changes the__of an object?

A. size
B. location
C. shape
D. both a and b

30) In two dimensional viewing we have?

A. 3D window and 2D viewport
B. 3D window and 3D viewport
C. 2D window and 2D viewport
D. 2D window and 2D viewport

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
A	B	A	A	C	B	C	A	A	C	A	C	A	A	C
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
A	B	A	A	C	A	A	A	A	C	A	C	A	A	A

Glossary

2D Graphics

Displayed representation of a scene or an object along two axes of reference: height and width (x and y).

3D Graphics

Displayed representation of a scene or an object that appears to have three axes of reference: height, width, and depth (x, y, and z).

3D Pipeline

The process of 3D graphics can be divided into three-stages: tessellation, geometry, and rendering. In the tessellation stage, a described model of an object is created, and the object is then converted to a set of polygons. The geometry stage includes transformation, lighting, and setup. The rendering stage, which is critical for 3D image quality, creates a two dimensional display from the polygons created in the geometry stage.

Anti-aliasing

Anti-aliasing is sub pixel interpolation, a technique that makes edges appear to have better resolution.

Bitmap

A Bitmap is a pixel by pixel image.

Blending

Blending is the combining of two or more objects by adding them on a pixel-by-pixel basis.

Depth Cueing

Depth cueing is the lowering of intensity as objects move away from the viewpoint.

Dithering

Dithering is a technique for archiving 24-bit quality in 8 or 16-bit frame buffers. Dithering uses two colors to create the appearance of a third, giving a smooth appearance to an otherwise abrupt transition.

Flat Shading

The flat shading method is also called constant shading. For rendering, it assigns a uniform color throughout an entire polygon. This shading results in the lowest quality, an object surface with a faceted appearance and a visible underlying geometry that looks 'blocky'.

Hidden Surface Removal

Hidden Surface Removal or visible surface determination entails displaying only those surfaces that are visible to a viewer because objects are a collection of surfaces or solids.

Interpolation

Interpolation is a mathematical way of regenerating missing or needed information. For example, an image needs to be scaled up by a factor of two, from 100 pixels to 200 pixels. The missing pixels are generated by interpolating between the two pixels that are on either side of the pixel that needs to be generated. After all of the 'missing' pixels have been interpolated, 200 pixels exist where only 100 existed before, and the image is twice as big as it used to be.

Lighting

There are many techniques for creating realistic graphical effects to simulate a real-life 3-D object on a 2-D display. One technique is lighting. Lighting creates a real-world environment by means of rendering the different grades of darkness and brightness of an object's appearance to make the object look solid.

Line Buffer

A line buffer is a memory buffer used to hold one line of video. If the horizontal resolution of the screen is 640 pixels and RGB is used as the color space, the line buffer would have to be 640 locations long by 3 bytes wide. This amounts to one location for each pixel and each color plane. Line buffers are typically used in filtering algorithms.

Projection

The process of reducing three dimensions to two dimensions for display is called Projection. It is the mapping of the visible part of a three dimensional object onto a two dimension screen.

Rasterization

Translating an image into pixels.

Rendering

The process of creating life-like images on a screen using mathematical models and formulas to add shading, color, and lamination to a 2D or 3D wireframe.

Transformation

Change of coordinates; a series of mathematical operations that act on output primitives and geometric attributes to convert them from modeling coordinates to device coordinates.

Z-buffer

A part of off-screen memory that holds the distance from the viewpoint for each pixel, the Z-value. When objects are rendered into a 2D frame buffer, the rendering engine must remove hidden surfaces.

Z-buffering

A process of removing hidden surfaces using the depth value stored in the Z-buffer. Before bringing in a new frame, the rendering engine clears the buffer, setting all Z-values to 'infinity'. When rendering objects, the engine assigns a Z-value to each pixel: the closer the pixel to the viewer, the smaller the Z value. When a new pixel is rendered, its depth is compared with the stored depth in the Z-buffer. The new pixel is written into the frame buffer only if its depth value is less than the stored one.

Z-sorting

A process of removing hidden surfaces by sorting polygons in back-to-front order prior to rendering. Thus, when the polygons are rendered, the forward-most surfaces are rendered last. The rendering results are correct unless objects are close to or intersect each other. The advantage is not requiring memory for storing depth values. The disadvantage is the cost in more CPU cycles and limitations when objects penetrate each other.

Filtering

This is a broad word which can mean the removal of coffee grinds from the coffee. However, within the narrow usage of this book, a filtering operation is the same as a convolution operation (see "convolution"). Anti-aliasing is usually done by filtering.

flat projection

A method of projecting a 3D scene onto a 2D image such that the resulting object sizes are not dependent on their position. Flat projection can be useful when a constant scale is needed throughout an image, such as in some mechanical drawings.

Frame

One complete video image. When interlacing is used, a frame is composed of two fields, each containing only half the scan lines.

GIF

A file format for storing images. GIF stands for Graphics Interchange format, and is owned by Compuserve, Inc.

key frame

A selected frame of an animation at which all the scene state is defined. In the key frame animation method, the scene state at key frames is interpolated to create the scene state at the in-between frames.

key frame animation

An animation control method that works by specifying the complete scene state at selected, or key, frames. The scene state for the remaining frames is interpolated from the state at the key frames.

Raster Scan

The name for the pattern the electron beam sweeps out on a CRT face. The image is made of closely spaced scan lines, or horizontal sweeps.

Refresh Rate

The rate at which parts of the image on a CRT are re-painted, or refreshed. The horizontal refresh rate is the rate at which individual scan lines are drawn. The vertical refresh rate is the rate at which fields are drawn in interlaced mode, or whole frames are drawn in non-interlaced mode.

Refresh Rate, Horizontal

The rate at which scan lines are drawn when the image on a CRT is re-drawn, or refreshed.

Refresh Rate, Vertical

The rate at which fields are re-drawn on a CRT when in interlaced mode, or the rate at which the whole image is re-drawn when in non-interlaced mode.

Scan Line

One line in a raster scan. Also used to mean one horizontal row of pixels.

Bibliography

1. J. Foley, A. Van Dam, S. Feiner, J. Hughes: Computer Graphics- Principles and Practice, Pearson
2. Hearn and Baker: Computer Graphics, PHI.