UNIT 1 INTRODUCTION TO COMPUTER GRAPHICS

Structures **Page Nos.**

1.0 INTRODUCTION

Early man used drawings to communicate even before he learnt to talk, write, or count. Incidentally, these ancient hieroglyphics (picture-writings) communicate as well today, as they must have done thousands of years ago, this fully supports the saying that "A picture is worth a thousand words" and in the era of computers we can add on to it or we may as well revise the saying to "*A computer is worth a million pictures!" ;* so, you can estimate the power of a computer as a communication system.

Now, with the advances in computer hardware and software, graphics has come a full circle and, more and more people are teaching and learning, communicating and sharing their ideas through the medium of graphics. By **graphics**, we mean any sketch, drawing, special artwork or other material that pictorially depict an object or a process or otherwise conveys information, as a supplement to or instead of written descriptions, and the utilisation of computers to accomplish such tasks leads to a new discipline of computer graphics. Traditionally, graphics has referred to engineering drawings of buildings, bridges, machine parts etc. and scientific drawings such as x-y curves, network and process flowcharts. In recent decades, graphics has ventured into industrial design, advertising and other artistic endeavours. During the last few years, even newspapers and periodicals aimed at the common man have begun to utilise graphics to present quantitative news such as selection results and production statistics. Computer graphics can do all this and more. In fact, the power and easy availability of computer graphics have increased the use of pictures to replace and augment words to describe, educate, or inform a wide variety of audiences, on a wide variety of subjects.

In this unit, we shall concentrate on the graphic capabilities and potential of the digital computer plus we will discuss the meaning of the term graphics and its types, in addition to which, we will also discuss the hardware used for practical application of graphics in different streams of life. The software section however, will be discussed in Block 4 of this course.

1.1 OBJECTIVES

After completing this unit, you should be able to:

- describe computer graphics, its features and characteristics;
- discuss applications of computer graphics in various fields, and
- describe various types of hardware, required to work with graphic systems.

1.2 WHAT IS COMPUTER GRAPHICS?

The meaning of the term Graphics, is Graphical Tricks. Every image or picture is in fact a graph and when different mathematical tricks are used to manipulate some change in its properties like shape, size, motion etc., through the help of computers then, the representation is nothing but computer graphics, so we can say that "*Computer Graphics (CG) is the field of visual computing, where one utilises computers both to generate visual images synthetically and to integrate or alter visual and spatial information sampled from the real world*." **Or** "*Computer Graphics is the pictorial representation manipulation of data by a computer*" **Or** "*Computer Graphics refers to any sketch, drawing, special artwork or other material generated with the help of computer to pictorially depict an object or a process or otherwise convey information, as a supplement to or instead of written descriptions*". Computer Graphics is a complex and diversified field. A Picture is a fundamental cohesive concept in Computer Graphics. Each picture consists of points called pixels (Picture- element). If we consider a complex picture, then complex database for pixels are considered, hence, complex algorithm are required to access them. These complex database contain data organised in various data structures such as ring structures,
B-tree etc.

In our earlier courses CS-60, we had learned mathematical tricks to do jugglery with the graphs that resulted from different functions, now let us learn how to juggle with graphics by using computers. There are many algorithms, which can be materialised to produce graphical effects on the screen through several graphical tools based on different languages that are available in the market.

Computer graphics can be broadly divided into the following classes:

- Business Graphics or the broader category of Presentation Graphics, which refers to graphics, such as bar-charts (also called histograms), pie-charts, pictograms (i.e., scaled symbols), x-y charts, etc. used to present quantitative information to inform and convince the audience.
- Scientific Graphics, such as x-y plots, curve-fitting, contour plots, system or program flowcharts etc.
- Scaled Drawings, such as architectural representations, drawings of buildings, bridges, and machines.
- Cartoons and artwork, including advertisements.
- Graphics User Interfaces (GUIs) which are the images that appear on almost all computer screens these days, designed to help the user utilise the software without having to refer to manuals or read a lot of text on the monitor.

We will discuss the various classes of computer graphics mentioned above in the following sections of this unit.

The most familiar and useful class of computer graphics involves movies and video games. Movies generally need graphics that are indistinguishable from physical reality, whereas video games need graphics that can be generated quickly enough to be perceived as smooth motion. These two needs are incompatible, but they define two-ways of communications between users and computations. In video games, the subject matter of computations is generally characters chasing and shooting at each other. A more familiar use of computer graphics exists for interacting with scientific computations apart from movies and games. This familiarisation of the use of computer graphics has influenced our life, through simulations, virtual reality, animation, we can extend the scope of education, entertainment, analysis etc. So, in global terms Computer graphics can be categorised in two ways:

Interactive Computer Graphics which is interactively used by users e.g., games. We will discuss details about this type of graphic systems in Block 4.

Passive Computer Graphic which has no option for users to interact or use computer graphics e.g., movies. We will discuss details about this type of graphic systems in Block 4.

1.3 APPLICATIONS

Research in computer graphics covers a broad range of application including both photorealistic and non-photorealistic image synthesis, image-based modeling and rendering and other multi-resolution methods, curve and surface design, range scanning, surface reconstruction and modeling, motion capture, motion editing, physics-based modeling, animation, interactive 3D user interfaces, image editing and colour reproduction. Work is going on in various fields but computer vision is a hot topic, where research tackles the general problem of estimating properties of an object or scene through the processing of images, both 2D photographs and 3D range maps. Within this broad scope, we investigate efficient ways to model, capture, manipulate, retrieve, and visualise real-world objects and environments.

Once you get into computer graphics, you’ll hear about all kinds of applications that do all kinds of things. This section will discuss not only the applications but also the software suitable for that type of application, so it is necessary to give you an understanding of what various applications do. While working on a project, you may need images, brochures, a newsletter, a PowerPoint presentation, poster, DVD etc. Thus, the question arises what software do I need to get my job done. The section will help to straighten all of that out in your head. Hopefully, if you know what does what, you won’t waist money duplicating purchases, and when other designers or co-workers are talking shop, you’ll know what is going on.

Graphic design applications are basically broken down on the basis of a few considerations. The first two considerations are, “Is your project for print, or web”. When I say web, what I really mean is monitor based publishing. This means that you are going to see your work on a computer screen, and television, or a big screen projector. So, as you read through this section, whenever we say “web based”, we mean monitor based. Beyond print and web, here are the various categories that we can think of that various applications would fit into; Image Manipulation; Vector Graphics; Page Layout; Web sight development; Presentation Software; Video Editing; DVD Production; Animation and Interactivity etc. If you are creating, or learning to create graphic design, computer art, or maybe “Digital Media” is the term that we should use, then it’s a good thing to understand the function of each application. There are many applications in the market and most of them are expensive. A few of the various application areas that are influenced by Computer graphics are:

- Presentation Graphics
- Painting and Drawing
- Photo Editing
- Scientific Visualisation
- Image Processing

- Education, Training, Entertainment and CAD
- Simulations
- Animation and Games

Let us discuss these fields one by one.

1.3.1 Presentation Graphics

The moment you are going to represent yourself or your company or product or research paper etc. simply standing and speaking is quite ineffective. Now, in such a situation where no one stands with you, your ultimate companions are the slides which have some information either in the form of text, charts, graphs etc., which make your presentation effective. If you think more deeply, these are nothing but the ways some curves (text/graph/charts) which are used in some sequence and to create such things, graphics is the ultimate option, this application of graphics is known as presentation graphics, which can be done very effectively through computers now-a-days. There are some softwares which helps you present you and your concerned effectively. Such application softwares are known as **Presentation Graphics softwares – which is a software that shows information in the form of a slide show** (A slideshow is a display of a series of chosen images, which is done for artistic or instructional purposes. Slideshows are conducted by a presenter using an apparatus which could be a computer or a projector).

Three major functions of presentation graphics are:

- an editor that allows text to be inserted and formatted,
- a method for inserting and manipulating graphic images, and
- a slide-show system to display the content.

The program that helps users to create presentations such as visual aids, handouts, and overhead slides to process artwork, graphics, and text and produce a series of 'slides'– which help speakers get their message across are presenation graphics softwares.

Example programs include some softwares like Apple's Keynote, Openoffice's (Star Office-by Sun microsystems) Impress, Microsoft Powerpoint and (for multimedia presentations, incorporating moving pictures, and sounds) Macromedia Director. Custom graphics can also be created in other programs such as Adobe Photoshop or Adobe Illustrator and then imported. With the growth of video and digital photography, many programs that handle these types of media also include presentation functions for displaying them in a similar "slide show" format.

Similar to programming extensions for an Operating system or web browser, "add ons" or plugins for presentation programs can be used to enhance their capabilities. For example, it would be useful to export a PowerPoint presentation as a Flash animation or PDF document. This would make delivery through removable media or sharing over the Internet easier. Since PDF files are designed to be shared regardless of platform and most web browsers already have the plugin to view Flash files, these formats would allow presentations to be more widely accessible.

We may say that Presentation graphics is more than just power point presentation because it includes any type of slide presentation, bar chart, pie chart, graphs and multimedia presentation. The key advantage of this software is that it help you show abstracts of representation of work.

Note: There are some softwares like canvas that improves the presentation created through powerpoint or keynote software. Although these software packages contain a lot of handy features, they lack many vector and image creation capabilities, therefore, creating a need for a graphic/illustration program. Scientists, engineers, and other technically-oriented professionals often call upon Canvas and its host of vector, image editing, and text features to create the exciting visual components for their presentation projects.

General questions that strike many graphic designers, students, and engineers rushing to import their illustrations and images into presentations are:

- What resolution should be used?
- Which file format is best?
- How do I keep the file size down?

Let us discuss in brief the suitability of the technique (Vector or Bitmap), and the file format appropriate to the creation of a better presentation.

Resolution

Graphic illustrations are used in presentations to help convey an idea or express a mood, two kinds of illustration graphics are:

1) Vector, and
2) Bitmap.

You may wonder which one of these is a better format when exporting to some software PowerPoint or Keynote or impress. The truth is that there are different situations that call for different methods, but here are some things to look out for. For instance, vectors are objects that are defined by anchor points and paths, while bitmapped graphics are digital images composed of pixels. **The advantage** of using vector graphics is that they are size independent, meaning that they could be resized with no loss in quality. Bitmapped graphics, on the other hand, provide a richer depth of colour but are size dependent and appear at the stated 72 dpi size**.**

File format

Say, we want an image of a Fly. The wings are partially transparent and to represent that in our presentation what be problematic if proper file format is not there. This choice of file format is hidden in the software that you may be using. Two cases for the same situation are discussed below:

- **The right file format that will allow us to create a transparent background in Keynote presentation.** Even though Keynote could import all common file formats such as GIF, JPG, and BMP, there is one format that will work particularly well which is .PSD. Using .PSD (Photoshop format) we are able to easily place a transparent image, even partially transparent sections of the image, such as the wings of the fly, as well as retain their properties.

- **The right file format that will allow us to create a transparent background in PowerPoint.** Even though PowerPoint could import all common file formats such as GIF, JPG, and BMP, there are two particular file formats that will work exceptionally well: TIFF and PNG. Using TIFF (Tagged-Image File Format) or PNG (Portable Network Graphic), we could easily remove the unwanted background quickly and easily in PowerPoint, a feature not available to the other mentioned file formats.

Note: TIFF or PNG: TIFF has been around longer than PNG, which was originally designed to replace GIF on the Web. PowerPoint works well with both these files when creating transparent backgrounds but generally PNG creates smaller file sizes with no loss of quality.

1.3.2 Painting and Drawing

When we talk about graphics, we mean pictures, and pictures can be either illustrations or photographs. If you want to get graphics into a Web page or multimedia presentation, you either have to create them in some kind of graphics application by drawing or painting them right there in the application, or bringing them into the application via a digital camera or scanner, and then editing and saving them in a form suitable to your medium.

Many software applications offer a variety of features for creating and editing pictures on the computer. Even multimedia authoring and word processing programs include some simple features for drawing on the computer.

So, painting and drawing application in computer graphics allows the user to pick and edit any object at any time. The basic difference is as follows:

Drawing in a software application means using tools that create "objects," such as squares, circles, lines or text, which the program treats as discrete units. If you draw a square in PowerPoint, for example, you can click anywhere on the square and move it around or resize it. It's an object, just like typing the letter "e" in a word processor. i.e., a drawing program allows a user to position standard shape (also called symbols, templates, or objects) which can be edited by translation, rotations and scaling operations on these shapes.

Painting functions, on the other hand, don't create objects. If you look at a computer screen, you'll see that it's made up of millions of tiny dots called pixels. You'll see the same thing in a simpler form if you look at the colour comics in the Sunday newspaper — lots of dots of different colour ink that form a picture. Unlike a drawing function, a paint function changes the colour of individual pixels based on the tools you choose. In a photograph of a person's face, for example, the colours change gradually because of light, shadow and complexion. You need a paint function to create this kind of effect; there's no object that you can select or move the way you can with the drawn square i.e., a painting program allows the user to paint arbitrary swaths using brushes of various sizes, shapes, colour and pattern. More painting program allows placement of such predefined shapes as rectangles, polygon and canvas. Any part of the canvas can be edited at pixel level.

The reason why the differences are important is that, as noted earlier, many different kinds of programs offer different kinds of graphics features at different levels of sophistication, but they tend to specialise in one or the other. For example:

1) Many word processors, like Word, offer a handful of simple drawing functions. They aren't that powerful, but if all you need is a basic illustration made up of simple shapes to clarify a point, they're fine.

2) Some programs specialise in graphics creation. Of these, some are all-purpose programs, like KidPix, which offers both drawing and painting functions. KidPix is targeted specifically at children; it has a simplified interface and lacks the sophisticated functions a professional artist might want.

   Other programs, like Adobe PhotoShop, specialise in painting functions, even though they may include drawing functions as well. Painter is a paint-oriented program that offers highly sophisticated, "natural media" functions that approximate the effects of watercolours or drawing with charcoal on textured paper.

   Other graphics programs, such as Adobe Illustrator, specialise in drawing for professional artists and designers; AutoCAD is used mainly for technical and engineering drawing.

3) Page layout, presentation, multimedia authoring and Web development programs usually contain a variety of graphics functions ranging from the simple to the

complex, but their main purpose is composition, not image creation or editing. That is, they allow you to create or import text and graphics and, perhaps, sound, animation and video.

Most of the graphics features in these types of programs are limited to drawing functions because they assume that you will do more complex work in a program dedicated to other functions (e.g., writing in a word processor, editing photos in a paint program), then import your work to arrange the different pieces in the composition program. (Some multimedia authoring systems, however, also offer painting and drawing functions.)

By the way, the differences in composition programs are mainly in the form of their output: Page layout programs, such as PageMaker and QuarkXPress, are for composing printed pages; presentation and multimedia authoring programs, such as PowerPoint and HyperStudio, are for slide shows and computer displays; and Web development applications, like Netscape Composer, are for, well, Web pages.

4) What if you are going to make a magazine, newspaper, book or maybe a multipage menu for a restaurant. In that case, we need a page layout program. The well known softwares in page layout are:

   a) Quark Express
   b) Page Maker (Adobe)
   c) Indesign (Adobe)
   d) Publisher (Microsoft)

   The Queen of Page Layout is Quark Express, owned by Quark Express and Indesign is the King owned by Adobe and finally there is Microsoft Publisher, which is very easy to use.

5) To Create posters, brochures, business cards, stationary, coffee mug design, cereal boxes, candy wrappers, half gallon jugs of orange juice, cups, or anything else you see in print, most designers are going to use vectorised programs to make these things come to life. Vectors are wonderful because they print extremely well, and you can scale them up to make them large, or scale them down to make them small, and there is no distortion. Adobe Illustrator is the King of Vector Programs, hands down. In Adobe Illustrator, you can create a 12 foot, by 12 foot document. If we are going to make anything that is going to be printed, we are doing it in Illustrator. Anything that you create in Illustrator, and the text you use, will come out great. The thing is, Illustrator is hard to learn. It is not an intuitive program at all. This is because vectors use control points called paths and anchor points. To someone new, they are hard to understand, find, and control. That's another story. If you are making a poster, you would make your logo, artwork and text in Illustrator. You would still manipulate your images in Photoshop, and then, "place" them to the Illustrator.

☞ Check Your Progress 1

1) What are the application areas of Computer Graphics. Write short notes on each.

   ………………………………………………………………………………………

   ………………………………………………………………………………………

   ………………………………………………………………………………………

   ………

2) What are the file formats available for Presentation Graphics?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

3) Write the full form of (1) TIFF (2) PNG.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

4) Differentiate between Drawing and Painting.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

5) What is Adobe Illustrator used for?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

6) Give some softwares that are suitable for Page Lay out generators like multipage menu for a restaurant?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

7) Is Powerpoint the only presentation graphics software available? If no, then, name some softwares and their developers otherwise provide features of the Powerpoint softwares.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

1.3.3 Photo Editing

Photo-editing programs are paint programs—it's just that they include many sophisticated functions for altering images and for controlling aspects of the image, like light and colour balance.

For the most part, any paint program can open and display a digital photo image, but it will probably not offer the range and depth of features that a true photo-editing program like PhotoShop does.

Note: KidPix is a general graphics program and PhotoShop is an image-editing program. PhotoShop is the standard used by almost all professional artists and image editors. Key graphic application involves image editing or manipulation, No matter what type of design you are creating, you are going to manipulate some images. You might change the content of the images, crop, resize, touchup, falsify, fade in, fade out, and/or whatever. Anything that you are going to do to change an image will get done in an image editing or image manipulation application.

There are three big players in image manipulation:

1) PhotoShop (Adobe)
2) FireWorks (Macro Media)
3) Corel (owned by Corel)

Almost everything you see in print or on the web has gone through PhotoShop. It is the king of image manipulation. With PhotoShop you can make anything look real. Photoshop comes bundled with a program called, “ImageReady”.

ImageReady helps your created animated gif, web site rollover effects, image maps and more. Most people that own PhotoShop use less than 10 per cent of its powerful tools.

Fireworks is a super image manipulation application. The thing is, if you open the program, many of the icons, the tool bar, the panels and many of the options in the drop down menus look just like options in PhotoShop. It kind of looks like somebody copied the other persons application.

Note:

- Video editing is in a new and revolutionary stage. Computers really weren’t ready to edit video affordably until right now. Right now, if you have a fast computer and a lot of storage memory, you can create video segments just like anything you see on TV. And, it works well. I would say that the most popular video editing programs are:

  IMovie (apple. Not the best, but easy to use.)
  Adobe Premiere (Adobe)
  Final Cut Pro (Apple)
  Studio Version 9 (Pinnacle Systems)

- **Web Design and Editing**

  To make and edit a website, the big three softwares are:

  1) DreamWeaver (MacroMedia)
  2) Frontpage (MicroSoft)
  3) Go Live (Adobe)
  4) Netscape Composer (Netscape).

Most web developers use DreamWeaver. It is a super tool. It will write your html, css, javascript and create your forms. It is the best.

Frontpage is known for writing lots of code that you don’t need. Go Live is great, but I have never met a person that uses it.

We listed Netscape Composer basically because it is free. It’s not a bad product for free. We teach a lot of people, “Intro do web design,” and if they don’t have any software to make web pages, and if they don’t want to learn html, we show them Composer.

1.3.4 Scientific Visualisation

It is difficult for the human brain to make sense out of the large volume of numbers produced by a scientific computation. Numerical and statistical methods are useful for solving this problem. Visualisation techniques are another approach for interpreting large data sets, providing insights that might be missed by statistical methods. The pictures they provide are a *vehicle for thinking* about the data.

As the volume of data accumulated from computations or from recorded measurements increases, it becomes more important that we be able to make sense out of such data quickly. Scientific visualisation, using computer graphics, is one way to do this.

Scientific visualisation involve interdisciplinary research into robust and effective computer science and visualisation tools for solving problems in biology, aeronautics, medical imaging, and other disciplines. The profound impact of scientific computing upon virtually every area of science and engineering has been well established. The increasing complexity of the underlying mathematical models has also highlighted the critical role to be played by Scientific visualisation. It, therefore, comes as no surprise that Scientific visualisation is one of the most active and exciting areas of Mathematics and Computing Science, and indeed one which is only beginning to mature. Scientific visualisation is a technology which helps to explore and understand scientific phenomena visually, objectively, quantitatively. Scientific visualisation allow scientists to think about the unthinkable and visualise the unviable. Through this we are seeking to understand data. We can generate beautiful pictures and graphs; we can add scientific information (temperature, exhaust emission or velocity) to an existing object thus becoming a scientific visualisation product.

Thus, we can say scientific visualisation is a scientists tool kit, which helps to simulate insight and understanding of any scientific issue, thus, helping not only in solving or analysing the same but also producing appropriate presentations of the same. This concept of scientific visualisation fits well with modeling and simulation. The *Figure 1* describes steps for visualisation of any scientific problem under consideration, these steps are followed recursively to visualize any complex situation.

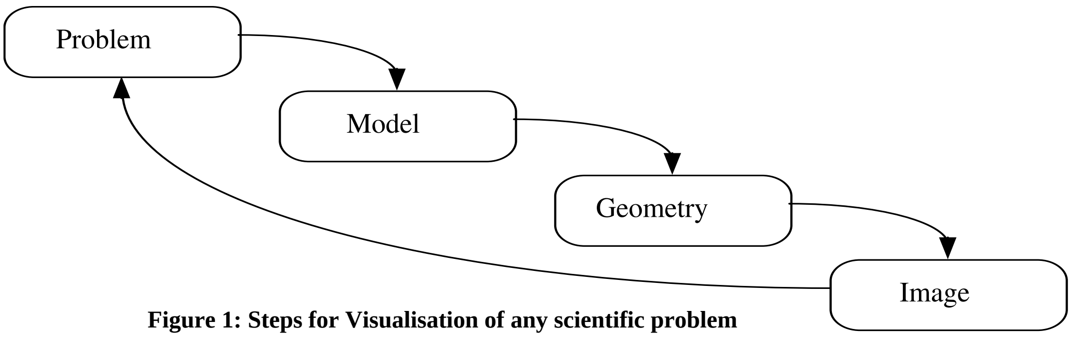


Figure 1: Steps for Visualisation of any scientific problem

Hence, *computer graphics* has become an important part of scientific computing. A large number of software packages now exist to aid the scientist in developing graphical representations of their data. Some of the tools or packages used to express the graphical result for modeling and simulation of any scientific visualisation are:

Matlab (by The Math Works Inc.)

Mathematica or Maple (graphical computer algebra system)

Stella (models dynamic systems)

IDS (Interactive Data Systems) by Research System Inc.

AVS (Application Visualisation System) by Advance visual System Inc.

Excel.

1.3.5 Image Processing

Modern digital technology has made it possible for the manipulation of multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. The goal of this manipulation can be divided into three categories:

1) Image Processing *image in -> image out*
2) Image Analysis *image in -> measurements out*
3) Image Understanding *image in -> high-level description out*

We will focus on the fundamental concepts of *image processing*. We can only make a few introductory remarks about *image analysis* here, as to go into details would be beyond the scope of this unit. *Image understanding* requires an approach that differs fundamentally from the theme of this section. Further, we will restrict ourselves to two-dimensional (2D) image processing although, most of the concepts and techniques that are to be described can be extended easily to three or more dimensions.

We begin with certain basic definitions. An image defined in the “real world” is considered to be a function of two real variables, for example, $a(x,y)$ with a as the amplitude (e.g., brightness) of the image at the *real* coordinate position (x,y). An image may be considered to contain sub-images sometimes referred to as *regions-of-interest*, *ROIs*, or simply *regions*. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus, one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve colour rendition.

The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantisation process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantised. In other image forming procedures, such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real phase.

A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitisation.

Let us discuss details of digitization. The 2D continuous image $a(x,y)$ is divided into *N rows* and *M columns*. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates $[m,n]$ with $\{m=0,1,2,...,M-1\}$ and $\{n=0,1,2,...,N-1\}$ is $a[m,n]$. In fact, in most cases $a(x,y)$ – which we might consider to be the physical signal that impinges on the face of a 2D sensor – is actually a function of many variables including depth (z), colour (λ), and time (t). The effect of digitisation is shown in *Figure 2*.

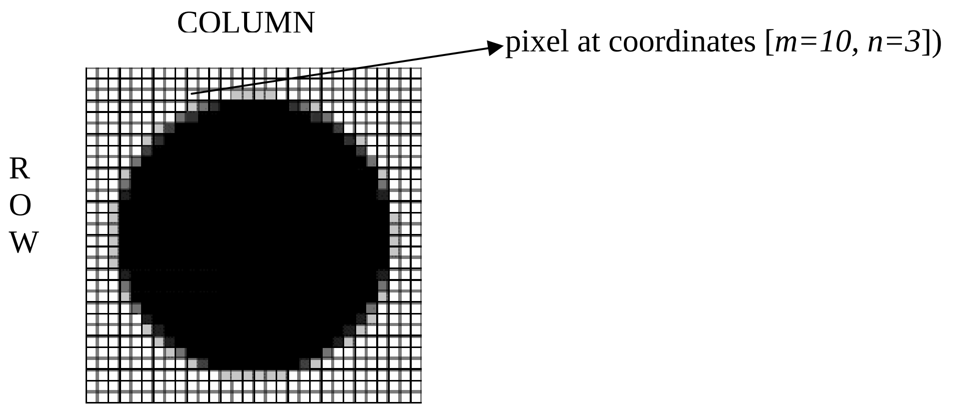


Figure 2: The effect of digitization

The image shown in *Figure 1* has been divided into *N* = 30 rows and *M* = 30 columns for digitisation of a continuous image. The value assigned to every pixel (pixel at coordinates [*m*=10, *n*=3]) is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with *L* different gray levels is usually referred to as amplitude quantisation or simply quantisation.

Example Tools and Software for Image Processing

Certain tools are central to the processing of digital images. These include mathematical tools such as *convolution*, *Fourier analysis*, and *statistical* descriptions, and manipulative tools such as *chain codes* and *run codes*. But these tools are worked with at very core levels, in general we use some software to process the image with the help of computers. Some of the categories of image processing software with their respective examples and features are listed below:

1) ***Graphics Image Processing:*** The most commonly used software is: Photoshop.

Features:
- Most common image processing software.
- Focuses on creating a pretty picture.
- Usually limited to popular graphics formats such as: TIFF, JPEG, GIF
- Best suited for working with RGB (3-band) images.
- Does not treat an image as a “map”.

2) ***Geographic Information Systems (GIS):*** The most commonly used software is: ArcMap.

Features:
- Works within a geographic context.
- Great for overlaying multiple vector and raster layers.
- It has a somewhat limited analysis although capability, these limitations are being reduced.
- More common than remote sensing software.

3) ***Remote Sensing Packages:*** Commonly used software example is: ERDAS

Features:
- Best suited for satellite imagery.
- Uses geo-spatial information.
- Easily works with multi-spectral data.
- Provides analysis functions commonly used for remote sensing applications.
- Often easy to use but it helps to be familiar with remote sensing.

4) ***Numerical Analysis Packages:*** Commonly used software is: MatLab.

Features:
- Focus usually on numeric processing.
- Programming or mathematical skills usually helpful.
- Used to build more user-friendly applications.

5) ***Web-based Services:*** Commonly used software is: Protected Area Archive.

Features:
- Image display, roam, zoom.
- Image enhancement.
- Simple image processing.

- Distance and area measurement.
- Comparison of old and new images.
- Image annotation (adding text, lines, etc).
- Overlaying vector layers.

Note:

1) Images are the final product of most processes in computer graphics. The ISO (International Standards Organization) defines computer graphics as the sum total of methods and techniques for concerning data for a graphics device by computer, it summarise computer graphics as converting data into images, also called visualisation.

$$\begin{tikzcd} \boxed{\text{Data}} \arrow[r] & \boxed{\text{Image}} \end{tikzcd}$$

2) Computer graphics concerns the pictorial synthesis of real or imaginary objects from their computer-based models, whereas the related field of image progressing treats the converse process, analysing and reconstruction of sciences. Images processing has sub-areas image enhancement, pattern detection and recognition. This one is used to improve the quality of images by using pattern detection and recognition. OCR is one of example.

1.3.6 Education, Training, Entertainment and Computer Aided Design (CAD)

CAD (or CADD) is an acronym that, depending on who you ask, can stand for:

- Computer Aided Design.
- Computer Aided Drafting.
- Computer Assisted Design.
- Computer Assisted Drafting.
- Computer Assisted Design and Drafting.
- Computer Aided Design and Drafting.

In general acronym for CAD is C*omputer-Aided Design*. In CAD interactive graphics is used to design components and systems of mechanical, electrical, and electronic devices. Actually CAD system is a combination of hardware and software that enables engineers and architects to design everything from furniture to airplanes. In addition to the software, CAD systems require a high-quality graphics monitor; a mouse, light pen or digitised tablets for drawing; and a special printer or plotter for printing design specifications.

CAD systems allow an engineer to view a design from any angle with the push of a button and to zoom in or out for close-ups and long-distance views. In addition, the computer keeps track of design dependencies so that when the engineer changes one value, all other values that depend on it are automatically changed accordingly.

Generally we use CAD as a tool for imparting education and training to the engineers, so that, they can produce beautifully carved and engineered pieces in bulk with the same amount of finishing and perfection. Generally a few terms are used repeatedly with CAD and they are CAM and CNC. Let us discuss **"What are CAD/CAM and CAD/CNC(or NC)"?**

The term CAD/CAM is a shortening of Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM). The term CAD/NC (Numerical Control) is equivalent in some industries.

CAD/CAM software uses CAD drawing tools to describe geometries used by the CAM portion of the program to define a tool path that will direct the motion of a machine tool to machine the exact shape that is to be drawn on the computer. Let us discuss the terms in brief.

Note:

- **Numerically-Controlled Machines:** Before the development of Computer-aided design, the manufacturing world adopted tools controlled by numbers and letters to fill the need for manufacturing complex shapes in an accurate and repeatable manner. During the 1950s these Numerically-Controlled machines used the existing technology of paper tapes with regularly spaced holes punched in them (think of the paper roll that makes an old-fashioned player piano work, but only one inch wide) to feed numbers into controller machines that were wired to the motors positioning the work on machine tools. The electro-mechanical nature of the controllers allowed digital technologies to be easily incorporated as they were developed. NC tools immediately raised automation of manufacturing to a new level once feedback loops were incorporated (the tool tells the computer where it is, while the computer tells it where it should be).

  What finally made NC technology enormously successful was the development of the universal NC programming language called APT (Automatically Programmed Tools). Announced at MIT in 1962, APT allowed programmers to develop postprocessors specific to each type of NC tool so that, the output from the APT program could be shared among different parties with different manufacturing capabilities.

  Now-a-days many new machine tools incorporate CNC technologies. These tools are used in every conceivable manufacturing sector, like CNC technology is related to Computer Integrated Manufacturing (CIM), Computer Aided Process Planning (CAPP) and other technologies such as Group Technology (GT) and Cellular Manufacturing. Flexible Manufacturing Systems (FMS) and Just-In-Time Production (JIT) are made possible by Numerically-Controlled Machines.

CAD and CAM

The development of Computer-aided design had little effect on CNC initially due to the different capabilities and file formats used by drawing and machining programs. However, as CAD applications such as SolidWorks and AutoCad incorporate CAM intelligence, and as CAM applications such as MasterCam adopt sophisticated CAD tools, both designers and manufacturers are now enjoying an increasing variety of capable CAD/CAM software. Most CAD/CAM software was developed for product development and the design and manufacturing of components and moulds, but they are being used by architects with greater frequency. Thus, a CAD program introduces the concept of real-world measurement. For example, a car or building can be drawn as if it were life-size, and later arranged into sheets and printed on paper at any desired scale.

Note:

1) CAD (or CADD) stands for Computer-Aided Design and Drafting. It differs from both "paint" and "draw" programs in that (i.e., CAD) measurement is central to its abilities. Whereas a "paint" program lets you manipulate each pixel in an array of pixels that make up an image, and a "draw" program goes a step further – it is composed of separate entities or objects, such as circles, lines, etc. It may provide facilities to group these into any object.

2) Is CAD only useful for design drawings?

   No. While true-scale, structurally valid drawings are the reason for CAD's existence, its use is as diverse as our customer's imaginations. For instance, it may be used for:

(a) page layout, web graphics (when scaling and relationships are important to an image, making the image in CAD and exporting it as a bitmap for touchup and conversion can be very productive),

(b) visually accessed databases (imagine a map with detail where you can zoom into an area and edit textual information "in place" and you can then see what other items of interest are "in the neighborhood" - our program's ability to work very rapidly with large drawings is a real plus here),

(c) sign layout, laser-cutting patterns for garment factories, schematic design (where CAD's symbol library capabilities come in handy), and printed-circuit board layout (This was the application that our first CAD program, created in 1977).

Software packages for CAD applications typically provide designer with a multi-window environment. Animations are often used in CAD application, Real-time animations using wire frame displays on a video monitor are useful for testing the performances of a vehicle or a system. The inter frame system allows the user to study the interior of the vehicle and its behaviour. When the study of behaviour is completed, realistic visualising models, surface rendering are used for background scenes and realistic display.

There are many CAD software applications. Some of them with their respective vendors are listed below:

CAD Applications	**Scanner Vendor: Desktop**
AlphaCAM	Canon
Ashlar Vellum	Epson
AutoCAD	Hewlett Packard
CATIA/CADCAM	UMAX

CAD Applications	**Scanner Vendor: Large Format**
Eagle point	Action Imaging
FastCAD	Contex
Pro/E	Xerox
FelixCAD	Kip
IntelliCAD	Ricoh
MasterCAM	Vidar
MasterStation	Widecom

There are many more applications not listed in the list given above.

These applications replicate the old drafting board as a means to draw and create designs. As CAD applications run on computers they provide a great deal more functionality than a drafting board, and a great deal more complexity. The lines and text created in CAD are **vectors**. This means that their shapes and positions are described in mathematical terms. **These vectors are stored on computer systems in *CAD files.***

There are a great **many different file formats for CAD**. Most CAD applications produce their own proprietary file format. The CAD applications from AutoDesk Inc. are used widely. As a result their DWG format is very common. Many other CAD applications from other vendors can produce and open DWG files, as well as their own proprietary formats. CAD data is often exchanged using DXF format.

Note: The DWG file format is a CAD vector format developed by the Autodesk and created by their AutoCAD application. DXF is also a CAD vector format. It is designed to allow the exchange of vector information between different CAD applications. Most CAD applications can save to and read from DXF format.

When CAD drawings are sent to printers the format commonly used is HPGL. HPGL files typically have the extension .plt.

Note: The HPGL file format is a vector format developed by Hewlett Packard for driving plotters. The file extensions used include .plt, .hpg, .hp2, .pl2 and sometimes .prn. However, the use of the .prn extension is not an absolute indicator that the file contains HPGL code. They are often referred to as 'plot files'. Trix Systems offers several options for handling HPGL and the later HPGL2 file formats.

☞ Check Your Progress 2

1) What is Photo Editing? What are the softwares used for image editing?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

2) What do you understand by term scientific visualisation, name some software used in this area?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

…………

3) What is image processing? Give some areas of importance is which the concept of image processing is of use also mention the fruitful softwares in the respective fields.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

…………

4) Is CAD useful only for designing drawings?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

…………

5) Briefly discuss DWG, DXF, formats and HPGL files.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

…………

1.3.7 Simulations

Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analysing the execution output. Simulation embodies the principle of “learning by doing” – to learn about the system we must first build a model of some sort and then operate the model. The use of simulation is an activity that is as natural as a child who *role plays*. Children understand the world around them by simulating (with toys and figures) most of their interactions with other people, animals and objects. As adults, we lose some of this childlike behaviour but recapture it later on through computer simulation. To understand reality and all of its complexity, we must build artificial objects and dynamically act our roles with them. Computer simulation is the electronic equivalent of this type of role playing and it serves to drive synthetic environments and virtual world. Within the overall task of simulation, there are three primary sub-fields: model design, model execution and model analysis (*Figure 3*).

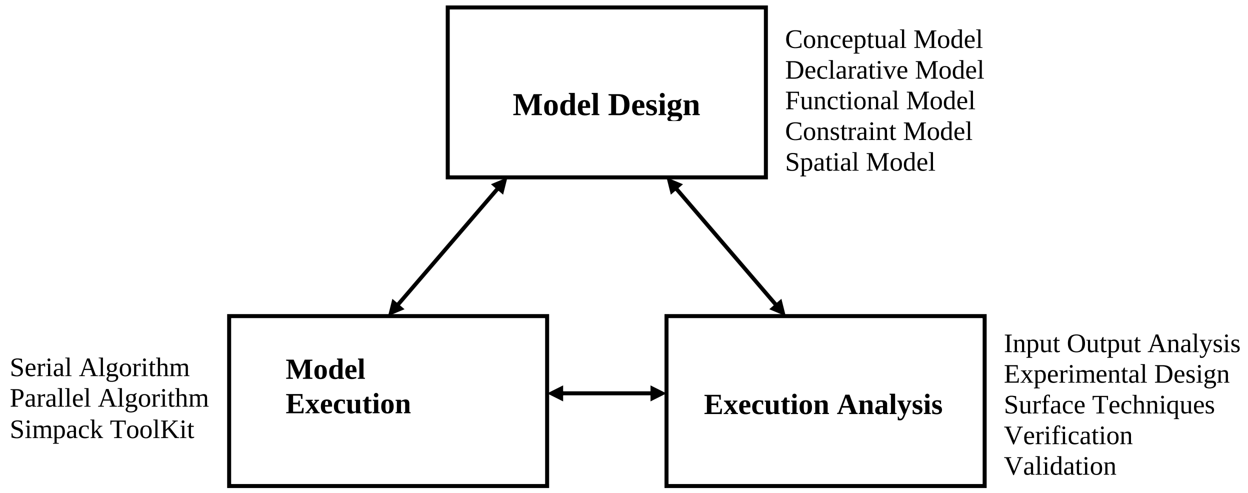


Figure 3: Three Sub-Fields of Computer Simulation

To simulate something physical, you will first need to create a *mathematical model,* which represents that physical object. Models can take many forms including declarative, functional, constraint, spatial or multimodel. A multimodel is a model containing multiple integrated models each of which represents a level of granularity for the physical system. The next task, once a model has been developed, is to execute the model on a computer – that is, you need to create a computer program which steps through time while updating the state and event variables in your mathematical model. There are many ways to “step through time”. You can, for instance, *leap* through time using *event scheduling* or you can employ small time increments using *time slicing*. You can also execute (i.e., simulate) the program on a massively parallel computer. This is called *parallel and distributed simulation*. For many large-scale models, this is the only feasible way of getting answers back in a reasonable amount of time.

You may want to know why to do simulation? Is there any other way to do the tasks? To discuss these issues lets briefly discuss the cases in which simulation is essential. There are many methods of modeling systems which do not involve simulation but which involve the solution of a closed-form system (such as a system of linear equations). Let us not go into these issues, as they are not part of our current discussion.

Simulation is often essential in the following cases:

1) The model is very complex with many variables and interacting components;
2) The underlying variables relationships are nonlinear;
3) The model contains random variates;
4) The model output is to be visual as in a 3D computer animation.

The Advantage of Simulation is that – even for easily solvable linear systems – a uniform model execution technique can be used to solve a large variety of systems without resorting to a "bag of tricks" where one must choose special-purpose and sometimes arcane solution methods to avoid simulation. Another important aspect of the simulation technique is that one builds a simulation model to replicate the actual system. When one uses the closed-form approach, the model is sometimes twisted to suit the closed-form nature of the solution method rather than to accurately represent the physical system. A harmonious compromise is to tackle system modeling with a hybrid approach using both closed-form methods and simulation. For example, we might begin to model a system with closed-form analysis and then proceed later with a simulation. This evolutionary procedure is often very effective.

Pitfalls in computer simulation

Although generally ignored in computer simulations, in strict logic the rules governing floating point arithmetic still apply. For example, the probabilistic risk analysis of factors determining the success of an oilfield exploration program involves combining samples from a variety of statistical distributions using the MonteCarlo methods. These include normal, lognormal, uniform and the triangular distributions. However, a sample from a distribution cannot sustain more significant figures than were present in the data or estimates that established those distributions. Thus, abiding by the rules of significant arithmatic, no result of a simulation can sustain more significant figures than were present in the input parameter with the least number of significant figures. If, for instance the net/gross ratio of oil-bearing strata is known to only one significant figure, then the result of the simulation cannot be more precise than one significant figure, although it may be presented as having three or four significant figures.

Note: *Monte Carlo methods* are a widely used class of computational algorithm for simulating the behaviour of various physical and mathematical systems. They are distinguished from other simulation methods (such as molecular dynamics) by being stochastic, that is non-deterministic in some manner – usually by using random number – as opposed to deterministic algorithms. Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to calculation using a computer, utilising many techniques of computer simulation. Further, *Monte Carlo algorithm* is a numerical Monte Carlo method used to find solutions to mathematical problems (which may have many variables) that cannot easily be solved, for example, by integral calculus, or other numerical methods. For many types of problems, its efficiency relative to other numerical methods increases as the dimensions of the problem increases.

1.3.8 Animation and Games

In our childhood, we have all seen the flip books of cricketers which came free along with some soft drink, where several pictures of the same person in different batting or bowling actions are sequentially arranged on separate pages, such that when we flip the pages of the book the picture appears to be in motion. This was a flipbook (several papers of the same size with an individual drawing on each paper so the viewer could flip through them). It is a simple application of the basic principle of physics called persistence of vision. This low tech animation was quite popular in the 1800s when the persistence of vision (which is 1/16th of a second) was discovered. This discovery led to some more interesting low tech animation devices like the zoetrope, wheel of life, etc. Later, depending on many basic mathematics and physics principles, several researches were conducted which allowed us to generate 2d/3d animations. In units 1 and 2 of block 2 we will study the transformations involved in computer graphics but you will notice that all transformations are related to space and not to time. Here lies the basic difference between animation and graphics. The difference is that animation adds to graphics the dimension of time which vastly increases the amount of information to be transmitted, so some methods are used to handle this vast information and these methods are known as animation methods which are classified as:

First Method: In this method, the artist creates a succession of cartoon frames, which are then combined into a film.

Second Method: Here, the physical models are positioned to the image to be recorded. On completion, the model moves to the next image for recording and this process is continued. Thus the historical approach of animation has classified computer animation into two main categories:

(a) *Computer-Assisted Animation* usually refers to 2d systems that computerise the traditional animation process. Here, the technique used is interpolation between key shapes which is the only algorithmic use of the computer in the production of this type of animation equation, curve morphing (key frames, interpolation, velocity control), image morphing.

(b) *Computer Generated Animation* is the animation presented via film or video, which is again based on the concept of persistence of vision because the eye-brain assembles a sequence of images and interprets them as a continuous movement and if the rate of change of pictures is quite fast then it induces the sensation of continuous motion.

This motion specification for computer-generated animation is further divided into 2 categories:

Low Level Techniques (Motion Specific) techniques are used to control the motion of any graphic object in any animation scene fully. Such techniques are also referred as motion specific techniques because we can specify the motion of any graphic object in the scene. Techniques such as interpolation, approximation etc., are used in motion specification of any graphic object. *Low level techniques* are used when animator usually has a fairly specific idea of the exact motion that s/he wants.

High Level Techniques (Motion Generalised) *are techniques* used to describe the general motion behaviour of any graphic object. *These techniques are* algorithms or models used to generate motion using a set of rules or constraints. The animator sets up the rules of the model, or chooses an appropriate algorithm, and selects initial values or boundary values. The system is then set into motion and the motion of the objects is controlled by the algorithm or model. This approach often relies on fairly sophisticated computation such as, vector algebra and numerical techniques among others.

So, the animation concept can be defined as: *A time based phenomenon for imparting visual changes in any scene according to any time sequence. The visual changes could be incorporated through the Translation of the object, scaling of the object, or change in colour, transparency, surface texture etc.*

Note: It is to be noted that computer animation can also be generated by changing camera parameters such as its position, orientation, focal length etc. plus changes in the light effects and other parameters associated with illumination and rendering can produce computer animation too.

Before the advent of computer animation, all animation was done by hand, which involved an enormous amount of work. You may have an idea of the amount of work by considering that each second of animation film contains 24 frames (film). Then, one can imagine the amount of work in creating even the shortest of animated films. Without going into details of traditional methods let us categorise computer animation technique. Computer animation can be categorised in two ways:

Interactive Computer Animation which is interactively used by users e.g., games. Sprite animation is interactive and used widely in Computer games. In its simplest form it is a 2D graphic object that moves across the display. Sprites often have transparent areas. Sprites are not restricted to rectangular shapes. Sprite animation lends itself well to interactivity. The position of each sprite is controlled by the user or by an application program (or by both). It is called “external” animation.We refer to animated objects (sprites or movies) as “animobs”. In games and in many multimedia applications, the animations should adapt themselves to the environment, the program status or the user activity. That is, animation should be *interactive*. To make the animations more event driven, one can embed a script, a small executable program, in every animob. Every time an animob touches another animob or when an animob gets clicked, the script is activated. The script then decides how to react to the event
(if at all). The script file itself is written by the animator or by a programmer. We will discuss about this in Block 4.

Passive Computer Animations: which has no option for users to use computer graphics today is largely interactive e.g., movies. Frame animation is non-interactive animation and is generally used in generating Cartoon movies. This is an “internal” animation method, i.e., it is animation inside a rectangular frame. It is similar to cartoon movies: a sequence of frames that follow each other at a fast rate, fast enough to convey fluent motion. It is typically pre-compiled and non-interactive. The frame is typically rectangular and non-transparent. Frame animation with transparency information is also referred to as “cel” animation. In traditional animation, a cel is a sheet of transparent acetate on which a single object (or character) is drawn.We will discuss this in Block 4.

There are various software which are used to generate computer animations. Some of them are:

- **Flash:** Learning MacroMedia’s Flash can be quite complex, but you can do almost anything with it. You can develop presentations, websites, portions of websites, games, or full-length feature, animated cartoons.

  You can import just about anything into Flash. You can drop in images of almost any file format, video clips, sounds and more. It is generally a 2D program.

- **Poser:** Poser by Curious Labs Creates 3D complex models that you can view, from any angle, distance or perspective. You can make the model look like any body you want it to. For instance, if you wanted to make a model that looks just like your Grandmother, you would do it in Poser (the learning curve is vast). Taking that to another level, you could then animate your Grandmother and make her run down a picture of a beach.

There are many more software related to this animation, we will discuss them in the Unit 1 of Block 4.

☞ Check Your Progress 3

1) What do you mean by simulation? What are its uses? Discuss the advantages and pitfalls of simulation.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

………………………………………………………………………………………
………………………………………………………………………………………
…………

2) Differentiate between Graphics and Animation.

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………

1.4 GRAPHICS HARDWARE

No matter with which advance graphic software you are working with, if your output device is not good, or hardware handling that software is not good, then ultimate result will be not good, as it could be. We want to say, hardwares also dominate the world of graphics. So, let us discuss some hardware devices which helps us to work with graphic packages.

1.4.1 Input and Output Devices

Input and Output devices are quite important for any software because an inappropriate selection of the concerned hardware may produce some erroneous results or may process data of some other format. So, in the following sections we have planned to discuss some of the input and output devices such as:

- Touch Panel
- Light Pens
- Graphics Tablet
- Plotters
- Film Recorders.

Touch Panels

Touch panels allow displayed object or screen positions to be selected with the touch of the finger and is also known as Touch Sensitive Screens (TSS). A typical application of touch panels is for the selection of processing options that are represented with graphical icons. Touch input can be recorded using optical electrical or acoustical methods.

Optical touch panels employ a line of intra red LEDS (light emitting diodes) along one vertical edge and along one horizontal edge of frame. The opposite vertical and horizontal edges contain light detection. These detections are used to record the beams that may have been interrupted when the panel was touched. Two crossing beams that are interrupted identify the horizontal and vertical coordinates of screen position selected.

An electrical touch panel is constructed with two transparent plates separated by a short distance. One of the plates is coated with a conducting material and the other is resistive material. When the outer plate is touched, it is forced into contact with the inner plate. The contact creates a voltage drop that is converted to a coordinate value of the selected screen position. They are not too reliable or accurate, but are easy to use. Four types are commonly in use. They are as follows:

1) **Electrical TSS:** Wire grid or other conductive coating is utilised to indicate a voltage drop at the point touched point, from which the position may be determined.

2) **Electro-Mechanical TSS:** A glass or plastic sheet with strain gages placed around the edges records the position by the relative magnitude of the deformation of the slightly bent plate.

3) **Optical TSS:** Infrared light from light-emitting diodes (LED) along two perpendicular edges of the screen and detectors along the opposite edges provide an (invisible) optical grid, with reference to which the finger's position is determined.

4) **Acoustic TSS:** Inaudible high-frequency sound waves are emitted along two perpendicular edges and reflected to the emitters by the finger; the echo interval is used as a measure of the distances from the edges.

Light Pen

Light pen is a pointing device. It has a light sensitive tip which is excited when the light is emitted and an illuminated point on the screen comes in its field of view. Unlike other devices which have associated hardware to track the device and determine x and y values, the light pen needs software support (some kind of tracking program). Pointing operations are easily programmed for light pens.

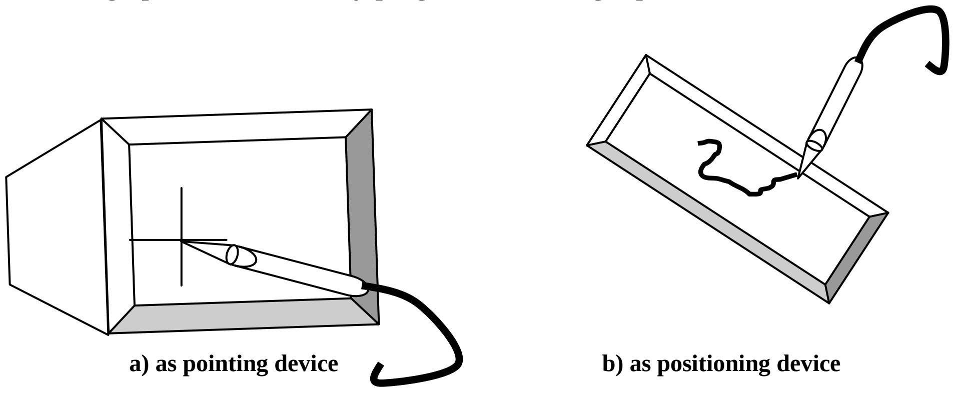


Figure 4: Light Pen Application

Figure 4 shows two typical applications of a light pen. It has a light sensitive tip and a photocell mounted in a pen-like case. If the light pen is pointed at an item on the screen it generates information from which the item can be identified by the program.

When the light pen senses an illuminated phosphor, it interrupts the display processor's interpreting of the file display. The processor's instruction register tells which instruction in the display file was being executed. By identifying the instruction responsible for the illuminated point, the machine can discover which object the pen is pointing to.

A light pen is an **event driven** device. The processor has to wait till it comes across an illuminated point on the screen to obtain any information. The keyboard is another typical example of an event driven device. The processor has to wait for a key to be pressed before it can determine what the user wants to input. Event driven devices can be handled in two ways as follows:

(a) ***Polling:*** The status of each device is periodically checked in a repetitive manner by a polling loop. When an event occurs, the loop is exited and the corresponding event is handled by executing some special event-handling routine or task. Again the polling continues. The disadvantage is that the processor has to wait in an idle state until some event occurs. Data entered can be lost if an event occurs at a time when the main program is not in its polling loop.

(b) ***Interrupts:*** An alternative to polling is the interrupt feature. The device sends an interrupt signal to the processor when an event occurs. The processor breaks from its normal execution and executes some special interrupt-handling routine or task. After the task is complete the control returns to the main program. To handle situations when more than one event occurs, different priorities are assigned to tasks so that higher priority tasks may interrupt tasks of lower priority.

Several events may occur before the program is ready for them. When more than one event occurs, the associate information is entered into the event queue. A polling loop can be employed to check the status of the event queue. The event queue can then pass input data from the polling task to the main program in the correct order. The main program takes events off the head of the queue and invokes the appropriate process. The devices need not be checked repeatedly for occurrence of events. Devices can interrupt even with the processor being unaware of it.

Two kinds of light pen interrupts may occur. If the user points the pen at an item on the screen to select it, as in *Figure 4(a)*, a selection interrupt occurs. If the user is positioning with the pen, as in *Figure 4(b)* a pattern called tracking pattern in displayed along the pen's movement and tracking interrupts occur when the pen sees the tracking pattern.

Modified versions of the light pen may also be used to draw lines, read barcodes, or do transformation operations on objects on the screen (or on a tablet).

Graphics Tablet

Before going into details on the graphic tablet, we need to know what we mean by tablet in computer terminology because, in other disciplines, the word tablet carries different meanings. In terms of computer science "Tablet is a special flat surface with a mechanism for indicating positions on it, normally used as a locator". This small digitiser is used for interactive work on a graphics workstation. Actually this device is essential when someone wants to do free hand drawing or to trace any solid geometrical shape. So a graphic tablet is a drawing tablet used for sketching new images or tracing old ones. Or we may say that a **graphics tablet** is a computer input device that allows one to hand-draw images and graphics, similar to the way one draws images with a pencil on paper. Or a Graphics tablet is a computer peripheral device that allows one to hand images directly to a computer, generally through an imaging program. Graphics tablets consists of a flat surface upon which the user may 'draw' an image using an attached pen-like drawing apparatus using which the user contacts the surface of the tablet, this apparatus is categorised into two types known as pen (or stylus) and puck (a flat block with cross-hairs and some switch keys), which may be wired or wireless. Often mistakenly called a mouse, the puck is officially the "tablet cursor." The image drawn or traced generally does not appear on the tablet itself but rather is displayed on the computer monitor.

The tablet and a hand-held pointer in the form of a stylus (pen) or puck, can serve one or more of these three functions:

(i) For selecting positions (on a drawing or on a menu) on the screen by moving the stylus on the tablet, in a sense using the stylus and tablet as pen on paper.

(ii) For issuing a command or to input a parameter by pressing the stylus at specified pre-programmed locations of a menu on the tablet.

(iii) For digitising the location on a drawing or map placed on the tablet with the stylus or puck.

This device is more accurate and efficient than a light pen. These are two types in use:

(a) **Voltage or Electro-Magnetic Field Tablet and Pointer:** This has a grid of wires, embedded in the tablet surface, with different voltages or magnetic fields corresponding to different coordinates. Intermediate positions within a cell can also be interpolated.

(b) **Acoustic or Sonic (Radio-Wave) Tablet and Pointer:** The sound of a spark at the tip of the stylus is picked up by strip microphones along two edges of the tablet. From the arrival time of the sound pulse at the microphones, the perpendicular distances of the stylus tip from the two axes are known. The acoustic method suffers from its inherent noisiness as well as its susceptibility to interference from other noise.

A combination of electric pulses and time-delay detection by a sensor in the stylus, called **Electro-acoustic Table** is also available.

Tablets typically support two modes of operation:

1) **Digitiser Mode:** creates a one-for-one correspondence between tablet and screen. Wherever you make contact on the tablet, is the exact location on the screen that is affected.

2) **Mouse Mode:** Mouse mode moves the screen pointer relative to any starting position on the tablet surface, just like a mouse.

When drawing or tracing on the tablet, a series of x-y coordinates (vector graphics) are created, either as a continuous stream of coordinates, or as end points. Further the drawings created or traced on tablets are stored as mathematical line segments; and these features of tablets help to produce, tablet computers, tablet PCs and pen tablets.

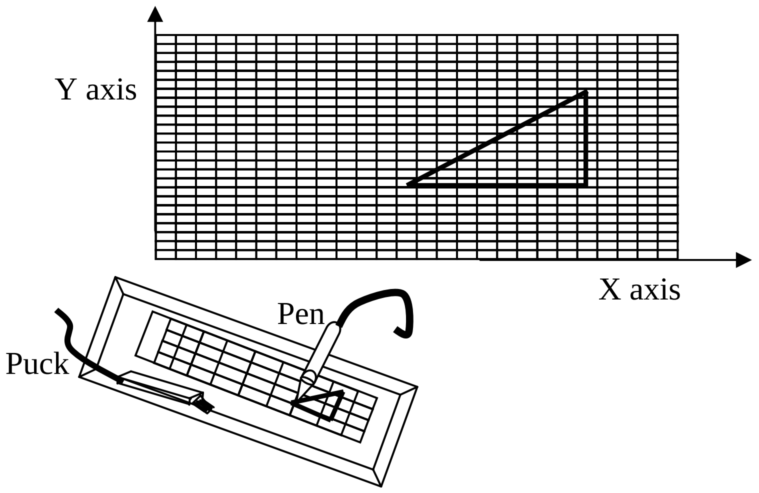


Figure 5: Graphic Tablet

Note: Objects are drawn with a pen (or stylus) or puck, but are traced with the puck only.

Tablet Computer: A complete computer contained in a touch screen. Tablet computers can be specialised for only Internet use or be full-blown, general-purpose PCs with all the bells and whistles of a desktop unit. The distinguishing characteristic is the use of the screen as an input device using a stylus or finger. In 2000, Microsoft began to promote a version of Windows XP for tablet computers, branding them “Tablet PCs”.

Pen Tablet: A digitiser tablet that is specialised for handwriting and hand marking. LCD-based tablets emulate the flow of ink as the tip touches the surface and pressure is applied. Non-display tablets display the handwriting on a separate computer screen.

Plotter: A plotter is a vector graphics-printing device that connects to a computer. Now-a-days, we use the plotter right from the field of engineering, to media and advertising. Even in our day-to-day lives we see a large number of computer designed hoardings and kiosks as publicity material. This fine output is achieved by using plotters with computers.

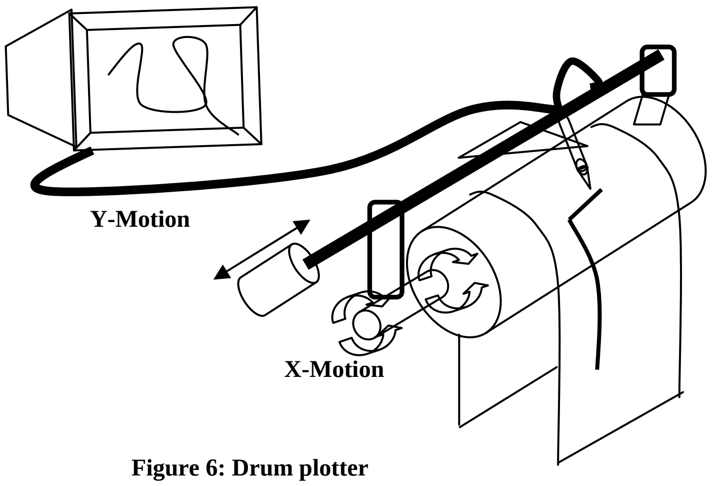


Figure 6: Drum plotter

But the printer may also be connected to the computer. The question then arises, as to how they differ from each other. So let us discuss the differences between them.

1) Plotters print their output by moving a pen across the surface of a piece of paper. This means that plotters are restricted to line art, rather than raster graphics as with other printers. They can draw complex line art, including text, but do so very slowly because of the mechanical movement of the pen.

2) Another difference between the plotter and the printer is that, the printer is aimed primarily at printing text. Thus, the printer is enough to generate a page of output, but this is not the case with the line art of the plotter.

Film Recorders

Film recorder is a graphical output devices for transferring digital images to photographic films.The simplest film recorders typically work by displaying the image on a grayscale Cathode Ray Tube (CRT) placed in front of a photographic camera. For colour images, the red, green, and blue channels are separately displayed on the same grayscale CRT, and exposed to the same piece of film through a filter of the appropriate colour. (This approach yields better resolution and colour quality than one could obtain with a colour CRT). The three filters are usually mounted on a motor-driven wheel. The filter wheel, as well as the camera's shutter, aperture, and film motion mechanism are usually controlled by the recorder's electronics and/or the driving software.

Higher-quality film recorders called LVT (Light Value Transfer) use laser to write the image directly onto the film, one pixel at a time. This method is better suited to print to large-format media such as poster-size prints. In any case, the exposed film is developed and printed by regular photographic chemical processing. Self-developing (polaroid) film can be used for immediate feedback.

Film recorders are used in digital printing to generate master negatives for offset and other bulk printing processes. They are also used to produce the master copies of movies that use computer animation or other special effects based on digital image processing. For preview, archiving, and small-volume reproduction, film recorders have been rendered obsolete by modern printers that produce photographic-quality hardcopies directly on plain paper.

Film recorders were also commonly used to produce slides for slide projectors; but this need is now largely met by video projectors that project images straight from a computer to a screen.

Film recorders were among the earliest computer graphics output devices. Nowadays, film recorders are primarily used in the motion picture film-out process for the ever increasing amount of digital intermediate work being done. Although significant advances in large venue video projection alleviates the need to output to film, there remains a deadlock between the motion picture studios and theatre owners over who should pay for the cost of these very costly projection systems. This, combined with the increase in international and independent film production, will keep the demand for film recording steady for at least a decade.

☞ Check Your Progress 4

1) What is a graphics tablet? How do the components of a graphic tablet i.e., Pen and Puck differ?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

2) What are touch panels? Discuss different touch panels that are currently available for use?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

3) How does a printer differ from a plotter?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

4) What are file recorders?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

1.4.2 Display Devices

As the importance of input and output devices has been discussed above, let us now focus our discussion specifically on display devices, which present the output to the end user who may not be a technically sound client. If the output display is appealing then your creation will definitely receive a word of appreciation otherwise you may be at the receiving end. Hence, it is pertinent to discuss some of the display devices next.

Refreshing Display Devices

Cathode Ray Tube: It is a refreshing display device. The concept of a refreshing display is depicted pictorially below:

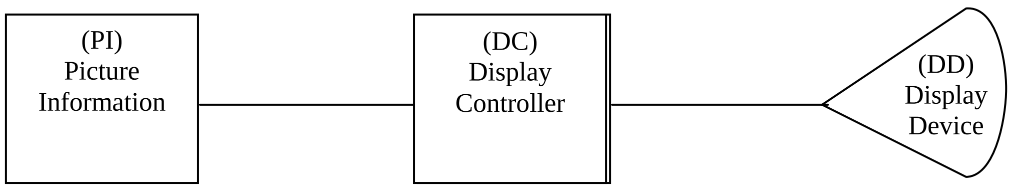


Figure 7: Block Diagram of Display Device

Actually the picture information is given through the stream of electrons (e^-) and its flow is controlled by the display controller (the control is as per the information supplied by the picture) finally the controlled e^- flow produces scintillations on the screen of the display device and the image is formed. The display is known as a refreshing display because display is continuously created by the continuous impugnation/striking of electrons on the screen at the same point. (i.e., same image is continuously made 40-50 times per second i.e., continuous refreshing occurs).

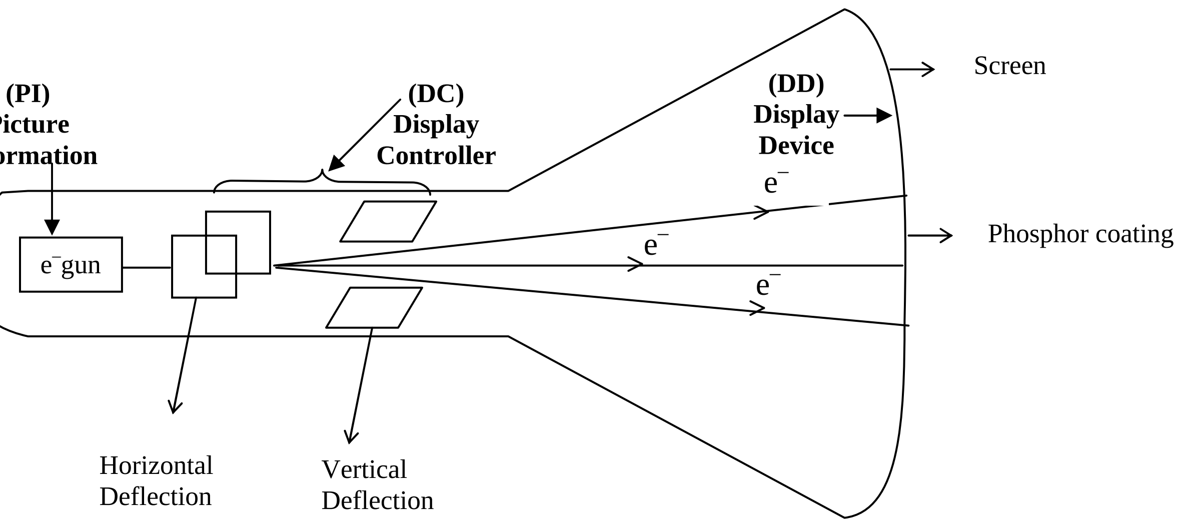


Figure 8: CRT

For proper image we also need to implant a “Digital to Analog converter” (**DAC –** it is an electronic circuit for digital to analog conversion) between the display controller and the display device.

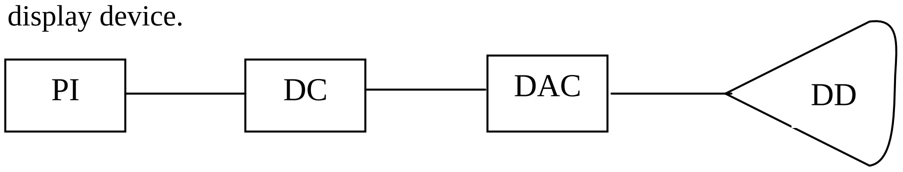


Figure 9: Block Diagram of Display Device with DAC

There are two kinds of refresh monitors, namely, the Random Scan and Raster Scan, which will be described separately.

Note:

1) In a Random Scan System, the Display buffer stores the picture information. Further, the device is capable of producing pictures made up of lines but not of curves. Thus, it is also known as “Vector display device or Line display device or Calligraphic display device”.

2) In a Raster Scan System, the Frame buffer stores the picture information, which is the bit plane (with m rows and n columns).

Because of this type of storage the system is capable of producing realistic images, but the limitation is that, the line segments may not appear to be smooth.

Random Scan Display Device

The original CRT, developed in the late fifties and early sixties, created charts and pictures, line by line on the tube surface in any (*random*) order or direction given, in a vectorial fashion. The electron beam was moved along the particular direction and for the particular length of the line as specified. For this reason, the type of device was known as a **Vector, Calligraphic** or **Stroke** (because it drew lines just as our ancestors drew letters like pictures, stroke by stroke). The process was similar to a hand-sketch or pen-plot.

The graphics commands are transmitted to a display-file program generated by the computer and stored in the **refresh storage area** or buffer memory. The program is executed once in each refresh cycle (of about 1/30 sec.) The electron beam is moved to trace the image line-by-line by a **display processor.** Each line, whether straight or curved, is displayed by the activation of specific points between specified end-point by means of **vector generators** of the analog or digital type, the former being smoother, the latter being faster and cheaper. Curved lines and text characters are displayed as a series of short lines or by a sequence of points.

The display by this system is called **Line Drawing Display.** The sequence operates the following stages, illustrated in *Figure 10*:

1) Graphics Commands
2) Display-File Translator
3) Display-File Program
4) Display (File) Processor
5) VDU

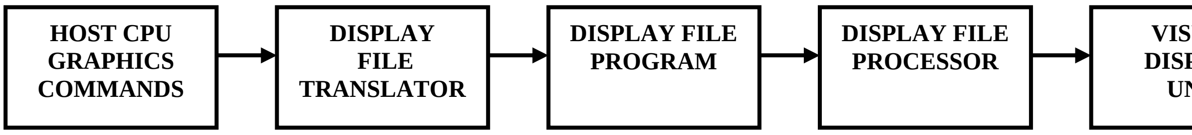
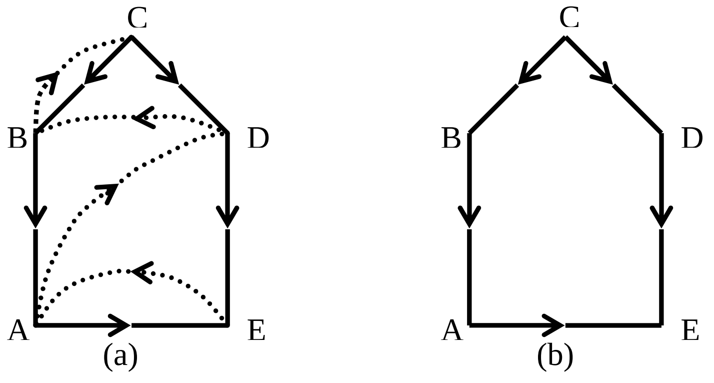


Figure 10: Block Diagram of Line Drawing Display

The process is quite similar to the way a person or a plotter draws a line segment, the pen being moved along a certain direction for a certain length, then changing direction or lifting the pen and moving to a new point, and so on.

For instance, the image of a hut shown in *Figure 11* would be traced as five line segments, the left and right roof slopes, the left and right walls, and the floor line. Efficiency is achieved by minimising wasted motion as would happen if a line segment starts at a point different from the end point of the previous segment – mequivalent to picking up the pen and putting it down at another point to draw the next segment. Thus, it would be better to trace the hut as ABCDE, rather than as CB, CD, BA, DE, and AE, as a draftsman might draw.



a) CB-CD-BA-DE-AE **b) ABCDEA (--- beam off, beam on)**

Figure 11: Vector Scan Display

Raster Scan Display Device

Current day screen display is also based on CRT technology, except that instead of displaying the picture tracing along one vector after another, the image is displayed as a collection of phosphor dots of regular shape arranged in a matrix form. These regular shapes are the pixels (picture elements) and cover the entire screen. The pixels could be (as in earlier times) rectangular, round, or (as is common now) square.
A pixel is the smallest unit addressable by the computer and is made up of a number of smaller dots, comprising the smallest phosphor particles in the CRT coating. However, in this text, we shall use the word “dot” synonymously with “pixel”.
The reasons as to why the original CRT did not become popular with the people using computer was because, the refresh procedure required a large memory and high speed, and the equipment to provide these was too expensive. It had to yield to the cheaper storage tube display.

Advances in television technology and chip memories in the mid-seventies overcame both problems with the development of the **raster display**, in which the picture was formed not directly by the lines forming the image but by a choice of appropriate dots from the array of pixels, the entire collection being called the **Raster.** Each row of pixels is called a **Raster Line,** or **Scan Line**.

The electron beam covers the display area horizontally, row by row, from top to bottom in a sweeping motion called **Raster Scan**, each dot lighting up with the intensity and shade of gray or a colour as directed by the Display Controller. Each complete sweep from top left to bottom right of the screen is one complete cycle, called the **Refresh Cycle**.

When viewed from a distance, all the dots together make up the effect of a picture, whether it is a scene from a serial as in the TV or a drawing in computer graphics. The picture looks smooth or coarse-grained depending on the screen **resolution**, that is the number of dots. Typically, on a computer monitor, there are 640 dots in the horizontal direction and 200 to 480 dots in the vertical direction. (The home TV is much finer grained, about three times or more, the scanning being done with analog signals for an entire line at a time, as against the digital information for each dot in the computer monitor). Today’s monitors can have resolutions of 1024 by 768 or even higher.

Even with the advent of raster scan, the concept of vector (random scan) graphics has not been completely eliminated. Certain standard geometric shapes such as straight-line segments, circles and ellipses are built into compilers and packages as equations, and developed into pixel graphics for the particular size and parameters specified. Similarly, and more recently, even many fonts in text typography have been reduced to equations so that the input letters, number etc. are really *drawn* from computed lines, as a form of vector graphics. Before going into more details on raster scan display device, let us discuss what is **Raster Scan**. It is the action, which is very similar to that of a dot matrix printer, or even a typewriter that is used to print a pretty border around a message, one line at a time. The image grows from top to bottom, one line at a time, unit completed only when the last line is printed.

The Raster Scan Proceeds as follows: Starting from the top left corner of the screen, the electron gun scans (sweeps) horizontally from left to right, one scan line, that is, one row at a time, jumping (without tracing the line) to the left end of the next lower row until the bottom right corner is reached. Then it jumps (again without tracing) to the top left corner and starts again, finishing one complete refresh cycle. *Figure 12* shows the track of a raster cycle.

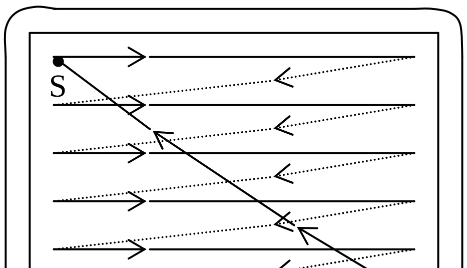


Figure 12: Raster Scan Cycle S-start, E-End

One cycle is completed in about 1/30th of a second, which is faster than the human eye can perceive-thus creating the impression of continuous display and motion.

This technology became very cost-effective, even inexpensive, and because of the availability of large memory and of its high refresh speed, it has become very popular. It takes us back to the refresh-type displays, and especially caters to the interactive and dynamic nature of modern-day computer graphics.

The main disadvantage of the raster scan is the jagged nature of the lines, arising from the fact that the pixels are aligned along regular rows and columns, and points on a line will not, in general, fall on the exact centres of the pixels. But the advantages far outweigh this disadvantage and further developments have diminished this jaggedness problem as well. Hence, almost all the monitors used today have raster display, using pixels, and all subsequent discussions in this text will be concerned with this last type.

All of the preceding characteristics of raster scan will apply to any image on the screen, whether it is graphics proper in the sense of a drawing, or it is a character (letter, number or other symbol).

Three components are necessary for the raster scan display. They are:

1) The **Frame Buffer** which is also called the **Refresh Buffer** or Bitmap. It is the refresh storage area in the digital memory, in which the matrix (array) of intensity values and other parameters (called **attributes**) of all the pixels making up the image are stored in binary form.

2) The display device, which converts the electrical signals into visible images, namely the VDU.

3) The **Display Controller**, the interface that transmits the contents of the frame buffer to the VDU in a form compatible with the display device, a certain number of (30 or more) times a second. The display controller also adjusts and makes allowances for the differences in the operating speeds of the various devices involved, and may also generate line segments and text characters.

Creating points, lines, characters, as well as filling in areas with shades or colours are all accomplished by this scan technique, known as the **Frame Buffer Display.** A common method for storing characters is to store the pixel information for the entire matrix (5*7 to 9*14, horizontal to vertical) assigned to represent a character.

The sequence of operations here is by the following stages, as depicted in *Figure 13.*

1) Graphics Commands
2) Display Processor (Scan Conversion)
3) Frame Buffer
4) Display Controller
5) VDU

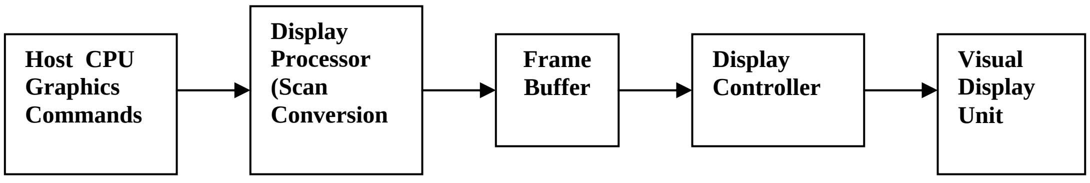

Host CPU Graphics Commands
Display Processor (Scan Conversion
Frame Buffer
Display Controller
Visual Display Unit

Figure 13: Block diagram of Raster Scan Display Device

Frame Buffers

The storage area in a raster scan display system is arranged as a two-dimensional table. Every row-column entry stores information such as brightness and/or colour value of the corresponding pixel on the screen. In a frame buffer each pixel can be represented by 1 to 24 or more bits depending on the quality (resolution) of the display system and certain attributes of the pixel. Higher the resolution, better the quality of the pictures. Commands to plot a point or line are converted into intensity and colour values of the pixel array or bitmap of an image by a process called **Scan Conversion**.

The display system cycles through the refresh buffer, row-by-row at speeds of 30 or 60 times per second to produce the image on the display. The intensity values picked up from the frame buffer are routed to the Digital/Analog converter which produces the necessary deflection signals to generate the raster scan. A flicker-free image is produced by interlacing all odd-numbered scan lines that are displayed first from, top to bottom and then, all even-numbered scan lines that are displayed. The effective refresh rate to produce the picture becomes much greater than 30 Hz.

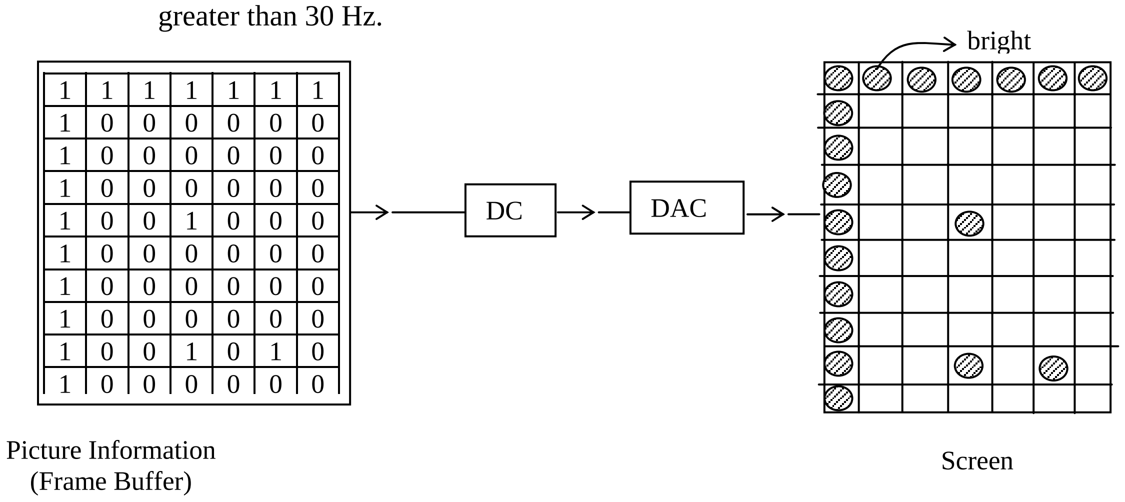


Figure 14: If information stored in frame buffer is 1 then, the corresponding pixel is made bright on the screen and if it is zero then no brightness appears i.e., 0→off; 1 → ON so the image obtained on the screen is discrete.

Different kinds of memory have been used in frame buffers: The earliest type of frame buffers used drums and disks with rotational frequency compatible to the rate of refresh. Such frame buffers were called **rotating-memory frame buffers**. But the relatively lower costs of integrated-circuit shift registers saw the rotating-memory frame buffer being replaced by the **shift-register frame buffers**.

A frame buffer can be constructed with a number of shift registers, each representing one column of pixels on the screen. Each shift register contributes one bit per horizontal scan line. However, changing a given spot on the screen is not very easy with shift registers. So they are not suitable for interactive applications.

Modern frame buffers use random-scan integrated circuits (discussed earlier) where the pixel intensities are represented by 1,2,4,8,16 or 24 bits. Encoding text and simple images does not require more than say, 8 bit per pixel. But to produce a good quality coloured image more than 8 bits, something like 24 bits, are required.

One of the best methods to encode coloured pictures involves the use of a colour map. The pixel values in the frame buffer are treated as addresses of a look-up-table, which has entries for every pixel's red, green and blue components. The entry value is used to control the intensity or colour on the CRT; each of the colour components can be defined to high precision providing accurate control over the colours displayed.

Another type of frame buffer is the **multiple-plane frame buffer,** where the frame buffer can be treated as consisting of several frames or planes, each containing information (intensity and/or colour) values of a separate image. An 8-bit per pixel frame buffer can be made to represent a single image with 8-bits of intensity precision or it can represent tow images each of 4-bit intensity precision or eight black and white images with 1-bit intensity precision each. A variety of image mixing can be done. For example, in animation systems, several moving objects can be displayed as separate planes.

Note:

1) In a frame buffer, information storage starts from top left corner and goes till the bottom right corner.
2) Using this type of buffer we can draw curves too.
3) So in order to draw live images of objects of day-to-day life, enormous picture information is required and this is the limitation.
4) How to vary intensity of a point (bright point) in Raster scan display device)

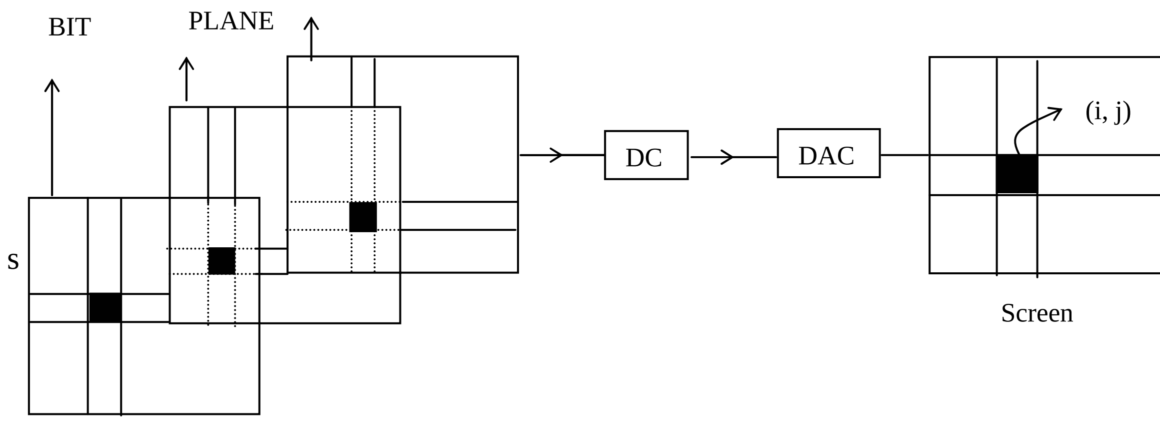


Figure 15: Frame buffer: Intensity Variation of a pixel

Picture Information

Here, the picture information is stored in the form of bit plans (on each bit plane full information of picture is stored) and for brightest intensity 3 bit plane (say for simplicity) should have 1 as respective information in the frame buffer, if it is zero then, the intensity will be the least; some of the intensity combination are discussed below:

bit plane			
A	B	C	
0	0	0	$\longrightarrow$ min. intensity of a point
0	0	1	
0	1	0	$\longrightarrow$ value of a pixel in 2nd bit plane
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	$\longrightarrow$ max intensity of a point

Maximum intensity of a pixel implies that the value for that point on each bit plane is 1 which generates maximum intensity (to have a bit low intensity any one of the bit plane is given information 0 for that point and so on):

If picture information is stored in 2 bit planes then possibilities are:

0 0 $\longrightarrow$ min. intensity of a point
0 1 $\longrightarrow$ medium intensity of a point
1 0 $\longrightarrow$ medium intensity of a point
1 1 $\longrightarrow$ max intensity of a point

For corresponding pixel on screen:

- each information digit is 0 or 1 for respective bit plane in continuation,
- each digit constitutes the information for a point in the respective bit plane,
- for 2 bit planes we have 4 levels, similarly, 3 bit planes we have 8 levels i.e., for n bit planes we have 2^n levels.
  $\Rightarrow$ for n bit plane 2^n amount of information is needed.
- By adopting this method of picture information storage, no doubt, we can vary the intensity of adopting but more memory consumption occurs because there exists more bit planes.
- For controlling the intensity of the colours, we need a minimum of 8 bit planes i.e., picture information has to have 8 digits (0, 1) further 2^8 combinations occur.
- For colours case we need in general 3 electron guns of 3 colours (red, green and blue) for picture display with varying colours.

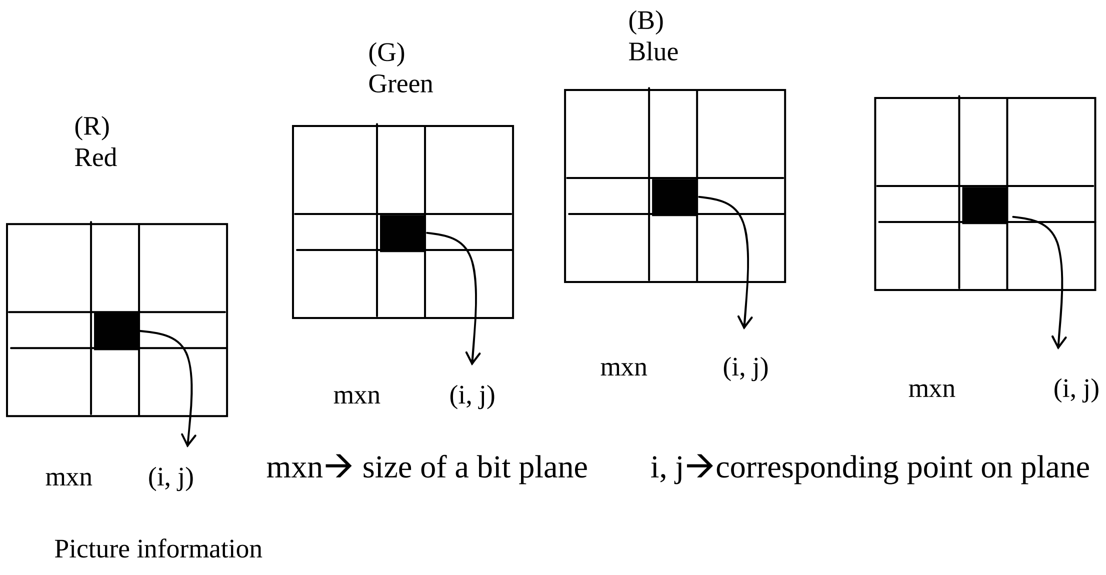


Figure 16: Frame Buffer: Colour Variation of a Pixel

As in the figure above we have 3 bit planes so we have 3 digit numbers. If information in each plane (R, G, B) is zero i.e., (0, 0, 0) then there is no display. Other situations are listed below:

	R	**G**	**B**	
So,	0	0	0	no display
	0	0	1	Blue display
	0	1	0	Green display
	0	1	1	Green-blue mix display and so on.

Similarly, for more colours we should have more bit planes and hence more numbers of digits signify more colour combinations.

Similarly we can control the intensity of colours

We have more bit planes for each colour say 3 bit planes for R, 3 for G and 3 for B (each bit plane of size mxn) so there exist a digit number and by respective 1s and 0s we can control colour intensity too. Here, the amount of information needed is 2^9 or amount of information needed is 2^3, 2^3, 2^3 for each R, G, B. Thus we can generate light red, light green or combination of light red, dark green, medium blue and so on.

Example 1: What is the number of memory bits required for 3 bit –plane frame buffer for a 512 x 512 raster.

Solution: Total memory bits required are 3 x 512 x 512 = 786,432

Example 2: What is the refresh rate in a 512 x 512 raster if pixels are accessed at the rate of 200 nanoseconds.

Solution. For individual access of pixels rate is 200×10^{-9} seconds, for 512x512 pixels 0.0524 seconds required.

Refresh rate per second is 1/0.0524 =19 frames/seconds.

Plasma Panel

It is an inherent memory device. Images can be written onto the surface point by point and they remain stable, without flicker, for a long time after being intensified. An inert mixture of noble gases (neon and xenon) gas is filled and sealed between two glass sheets with fine-mesh gold-wire electrodes attached to their inner faces. The particles of the gas behave in a bi-stable manner, that is stable at two levels (on/off). When voltage is applied across the electrodes the gas molecules get ionised and are activated giving rise to a glow.

The fine mesh of electrodes makes up thousands of addressable cells. A definite picture is generated when gas particles in the corresponding cells are excited by the application of the appropriate voltages. The ionised gas molecules in the cells excited by the applied voltages glow, making the picture visible until the current is turned off. This gives a steady image but the resolution is poor (60 dots per inch), and the hardware is much more complex (hence costlier) than raster scan.The plasma display panel is less bulky than the CRT but also vary the cost of construction is very high. Addressing of cells and wiring is complex.

Advantage

- Slim design (Wall mountable)
- Larger than LCD screens

Disadvantage

- Expensive, although cheaper than LCDs in larger sizes.
- Is subject to screen burn-in, but modern panels have a manufacturer rated lifespan of 50,000 or more hours.
- First 2000 hours is its brightest point. Every hour thereafter, the display gradually dims.
- At higher elevations, usually 6000 ft or higher, they exhibit noticeable humming.

LCD

Liquid Crystal Display is a type of display used in digital watches and many portable computers. These work with polarised ambient (outside source) light consisting of liquid crystal (a material which polarises light when a voltage is applied to it), with a conductive coating for the voltage application, set between two transparent glass or plastic plates and a polarised film on one side which will show the excited portions of the liquid crystal as dark dots or lines. (The seven-segment display of most digital watches is an example of LCD by lines). This technology is now applied to Data Projectors to project computer generated or stored images directly on to big screens in auditoriums.

LCD displays utilise two sheets of polarising material with a liquid crystal solution between them. An electric current passed through the liquid causes the crystals to align so that light cannot pass through them. Each crystal, therefore, is like a shutter, either allowing light to pass through or blocking the light.

Monochrome LCD images usually appear as blue or dark gray images on top of a grayish-white background. Colour LCD displays use two basic techniques for producing colour: *Passive matrix* is the less expensive of the two technologies. The other technology, called *Thin Film Transistor* (TFT) or *active-matrix*, produces colour images that are as sharp as traditional CRT displays, but the technology is expensive. Recent passive-matrix displays using new CSTN and DSTN technologies produce sharp colours rivaling active-matrix displays. Most LCD screens used in notebook computers are backlit, or transmissive, to make them easier to read.

☞ **Check Your Progress 5**

1) What are Refreshing display devices? What are their limitations?
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
………………

2) Differentiate between Random and Raster Scan display devices.
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………

1.5 SUMMARY

In this unit, we have discussed the conceptual meaning of computer graphics, with its application in various fields right from presentation graphics to animation and games. We have also discussed the variety of software and their respective file formats used in various applications of computer graphics. In the end, we have discussed the working of various input and output devices. Finally, the discussion on display devices was done with coverage to Refreshing Display Devices, and Plasma Panels, and LCD Display Devices.

1.6 SOLUTIONS/ANSWERS

Check Your Progress 1

1) A few of the various applications areas which are influenced by Computer graphics are:

- Presentation Graphics
- Painting Drawing
- Photo Editing
- Scientific Visualisation
- Image Processing
- Digital Art
- Education, Training, Entertainment and CAD
- Simulation
- Animation and games.

2) GIF, JPG, BMP, PSD, TIFF, PNG etc.

3) TIFF or PNG: TIFF has been around longer than PNG, which was originally designed to replace GIF on the Web. PowerPoint works well with both of these files when creating transparent backgrounds but generally PNG creates smaller file sizes with no loss of quality.

4)

Drawing	***Painting***
Drawing is a software application means using tools that create "objects," such as squares, circles, lines or text, which the program treats as discrete units. If you draw a square in PowerPoint, for example, you can click anywhere on the square and move it around or resize it. It's an object, just like typing the letter "e" in a word processor, i.e., a drawing program allows a user to position standard shape (also called symbols, templates, or objects) which can be edited by translation, rotations and scaling operations on these shapes.	***Painting*** functions, on the other hand, don't create objects. If you look at a computer screen, you'll see that it's made up of millions of tiny dots called pixels. You'll see the same thing in a simpler form if you look at the colour comics in the Sunday newspaper—lots of dots of different colour ink that form a picture. Unlike a drawing function, a paint function changes the colour of individual pixels based on the tools you choose. In a photograph of a person's face, for example, the colours change gradually because of light, shadow and complexion. You need a paint function to create this kind of effect; there's no object that you can select or move the way you can with the drawn square, i.e., a painting program allows the user to paint arbitrary swaths using a brush various size, shape, colour and pattern. More painting programs allows placement of such predefined shapes as rectangles, polygon and canvas. Any part of the canvas can be edited at the pixel level.

The reason why the differences are important is that, as noted earlier, many different kinds of programs offer different kinds of graphics features at different levels of sophistication, but they tend to specialise in one or the other.

5) To Create posters, brochures, business cards, stationary, coffee cup mug design, cereal boxes, candy wrappers, orange juice gallon jugs, cups, or anything else you see in print. Most designers will use vectorised programs to make these things come to life. Vectors are wonderful because they print extremely well, and you can scale them up to make them large, or scale them down to make them small, and there is no distortion. **Adobe Illustrator is the King of Vector Programs**. In Adobe Illustrator, you can create a 12 foot, by 12 foot document.

6) If you are going to make a magazine, newspaper, book or maybe a multipage menu for a restaurant. In that case, we need a page layout program. The well known softwares in page layout are:

   (a) Quark Express
   (b) Page Maker (Adobe)
   (c) Indesign (Adobe)
   (d) Publisher (MicroSoft)

7) No, other example of softwares are Apple's Keynote, Openoffice's (Star Office-by Sun microsystems), Impress, Microsoft Powerpoint and (for multimedia presentations, incorporating moving pictures, and sounds) Macromedia Director. Custom graphics can also be created in other programs such as Adobe Photoshop or Adobe Illustrator.

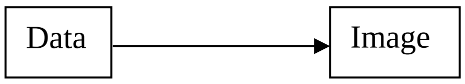
Check Your Progress 2

1) Photo-editing stream involves programs, which are not just paint programs—but they include many sophisticated functions for altering images and for controlling aspects of the image, like light and colour balance. Some of the professionally used software for photo editing are PhotoShop (Adobe), FireWorks (Macro Media), Corel (owned by Corel) etc.

2) Scientific visualisation involves interdisciplinary research into robust and effective computer science and visualisation tools for solving problems in biology, aeronautics, medical imaging, and other disciplines. The profound impact of scientific computing upon virtually every area of science and engineering has been well established. Some examples of the software used in this field are:

   Matlab (by The Math Works Inc.)
   Mathematica or Maple (graphical computer algebra system)
   Stella (models dynamic systems)
   IDL (Interactive Data Systems) by Research System Inc.
   AVS (Application Visualisation System) by Advance visual System Inc.

3) Image Processing means *image in* → processed *image out*
   Images are the final product of most processes in computer graphics. The ISO (International Standards Organisation) defines computer graphics as the sum total of methods and techniques for concerning data for a graphics device by a computer. It summarise and computer graphics as converting data into images, which is known as visualisation.



   Some of the categories of image processing software with their respective examples and features are listed below:

   (a) ***Graphics Image Processing:*** Commonly used software example is: Photoshop.
   (b) ***Geographic Information Systems (GIS):*** Commonly used software example is: ArcMap.
   (c) ***Remote Sensing Packages:*** Commonly used software example is: ERDAS.
   (d) ***Numerical Analysis Packages:*** Commonly used software example is: MatLab.

(e) ***Web-based Services:*** Commonly used software example is: Protected Area Archive.

4) No. While true-scale, structurally valid drawings are the reason for CAD's existence, its use is as diverse as our customer's imaginations.

   (a) Page layout, web graphics (when scaling and relationships are important to an image, making the image in CAD and exporting it as a bitmap for touchup and conversion can be very productive).

   (b) Visually accessed databases (imagine a map with details where you can zoom into an area and edit textual information "in place" and you can then see what other items of interest are "in the neighbourhood" – our program's ability to work very rapidly with large drawings is a real plus here).

   (c) Sign layout, laser-cutting patterns for garment factories, schematic design (where CAD's symbol library capabilities come in handy), and printed-circuit board layout (This was the application that our first CAD program, created in 1977).

5) The DWG file format is a CAD vector format developed by Autodesk and created by their AutoCAD application. DXF is also a CAD vector format. It is designed to allow the exchange of vector information between different CAD applications. Most CAD applications can save to and read from DXF format.

   When CAD drawings are sent to printers the format commonly used is HPGL. HPGL files typically have the extension .plt.

   The HPGL file format is a vector format developed by Hewlett Packard for driving plotters. The file extensions used include .plt, .hpg, .hp2, .pl2 and sometimes .prn. However, the use of the .prn extension is not an absolute indicator that the file contains an HPGL code. They are often referred to as 'plot files'. Trix Systems offers several options for handling HPGL and the later HPGL2 file formats.

Check Your Progress 3

1) Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analysing the execution output. Simulation embodies the principle of "learning by doing" – to learn about the system we must first build a model of some sort and then operate the model. Simulation is often essential in the following cases:

- the model is very complex with many variables and interacting components;
- the underlying variables relationships are nonlinear;
- the model contains random variates;
- the model output is to be visual as in a 3D computer animation.

The Advantage of Simulation is that – even for easily solvable linear systems – a uniform model execution technique can be used to solve a large variety of systems without resorting to a "bag of tricks" where one must choose special-purpose and sometimes arcane solution methods to avoid simulation. Another important aspect of the simulation technique is that one builds a simulation model to replicate the actual system. When one uses the closed-form approach, the model is sometimes twisted to suit the closed-form nature of the solution method rather than to accurately represent the physical system. A harmonious compromise is to tackle system modeling with a hybrid approach using both closed-form methods and simulation. For example, we might begin to model a system with closed-form analysis and then proceed later with a simulation. This evolutionary procedure is often very effective.

Pitfalls in Computer Simulation

Although generally ignored in computer simulations, in strict logic the rules governing floating point arithmetic still apply. For example, the probabilistic risk analysis of factors determining the success of an oilfield exploration program involves combining samples from a variety of statistical distributions using the MonteCarlo methods. These include normal, lognormal, uniform and the triangular distributions. However, a sample from a distribution cannot sustain more significant figures than were present in data or estimates that established those distributions. Thus, abiding by the rules of significant arithmatic, no result of a simulation can sustain more significant figures than were present in the input parameter with the least number of significant figures. If, for instance the net/gross ratio of oil-bearing strata is known to only one significant figure, then the result of the simulation cannot be more precise than one significant figure, although it may be presented as having three or four significant figures.

2) Animation is a time based phenomenon for imparting visual changes in any scene according to any time sequence, the visual changes could be incorporated through translation of object, scaling of object, or change in colour, transparency, surface texture etc., whereas Graphics does not contain the dimension of time.

Graphics + Dimension of Time = Animation

Check Your Progress 4

1) A graphic tablet is a drawing tablet used for sketching new images or tracing old ones. Or we can say that a graphics tablet is a computer input device that allows one to hand-draw images and graphics, similar to the way one draws images with a pencil or paper. Or Graphics tablet is a computer peripheral device that allows one to hand images directly into a computer, generally through an imaging program.

   Difference between pen and puck: Objects are drawn with a pen (or stylus) or puck, but are traced with the puck only.

2) Touch panels allow displayed objects or screen positions to be selected with the touch of the finger, also known as Touch Sensitive Screens (TSS). Four types of touch panels commonly in use, are Electrical, Electro-Mechanical, Optical, Acoustic touch panels.

3) The differences between printers and plotters:

   (a) Plotters print their output by moving a pen across the surface of piece of a paper. This means that plotters are restricted to line art, rather than raster graphics as with other printers. They can draw complex line art, including text, but do so very slowly because of mechanical movement of pen.

   (b) Another difference between plotter and printer is that the printer is aimed primarily at printing text. Thus, the printer is enough to generate a page of output, but this is not the case with the line art on a plotter.

4) Film recorder is a graphical output devices for transferring digital images to photographic films.

Check Your Progress 5

1) Refreshing display devices are those display devices in which the picture continuously refreshes. The general refresh rate is 40 to 60 times per second example CRT. There are two kinds of refreshing display devices, namely the random scan and raster scan, the limitation is that these devices are not competent to provide smooth and good quality images.

2) In Random Scan system the Display buffer stores the picture information, further the device is capable of producing pictures made up of lines but not of curves. Thus, it is also known as “Vector display device or Line display device or Calligraphic display device.

   In Raster Scan system the Frame buffer stores the picture information which is the bit plane (with m rows and n columns) because of this type of storage the system is capable of producing realistic images, but the limitation is that the line segments may not appear to be smooth.

UNIT 2 GRAPHIC PRIMITIVES

Structure **Page Nos.**

2.1 INTRODUCTION

In unit 1 of this block, we have discussed refreshing display devices and its types that are Random Scan display devices and Raster Scan display devices. We have also discussed the limitations of these display devices. In Raster Scan display device, which we have discussed in the previous unit, the picture information is stored in the frame buffer, the concept of frame buffer conveys that the information for the image to be projected on the screen is stored in the form of 0s and 1s, making respective pixels activate and deactivate on the screen, and it is the concept itself which contributes to the discreteness in the picture under display. So, in this unit we are going to extend our discussion to algorithms, which are responsible for actual display of the geometries like line, circle etc., on display devices, such that there is decrease in discreteness and hence, more realistic finished image is the outcome.

2.2 OBJECTIVES

After going through this unit, you should be able to:

- describe the Line Generation Algorithm;
- apply Line Generation Algorithm to practical problems;
- describe the different types of Line Generation Algorithm;
- describe Circle Generation Algorithm;
- apply Circle Generation algorithm to practical problems;
- describe the different types of Circle Generation Algorithms;
- describe Polygon fill algorithm, and
- describe Scan Line Polygon fill algorithm.

2.3 POINTS AND LINES

In the previous unit, we have seen that in order to draw primitive objects, one has to first scan convert the objects. This refers to the operation of finding out the location of pixels to be intensified and then setting the values of corresponding bits, to the desired intensity level. Each pixel on the display surface has a finite size depending on the screen resolution and hence, a pixel cannot represent a single mathematical point. However, we consider each pixel as a unit square area identified by the coordinate of its lower left corner, the origin of the reference coordinate system being located at the lower left corner of the display surface. Thus, each pixel is accessed by a non-negative integer coordinate pair (*x*, *y*). The x values start at the origin and increase from left to right along a scan line and the y values

(i.e., the scan line numbers) start at bottom and increase upwards.

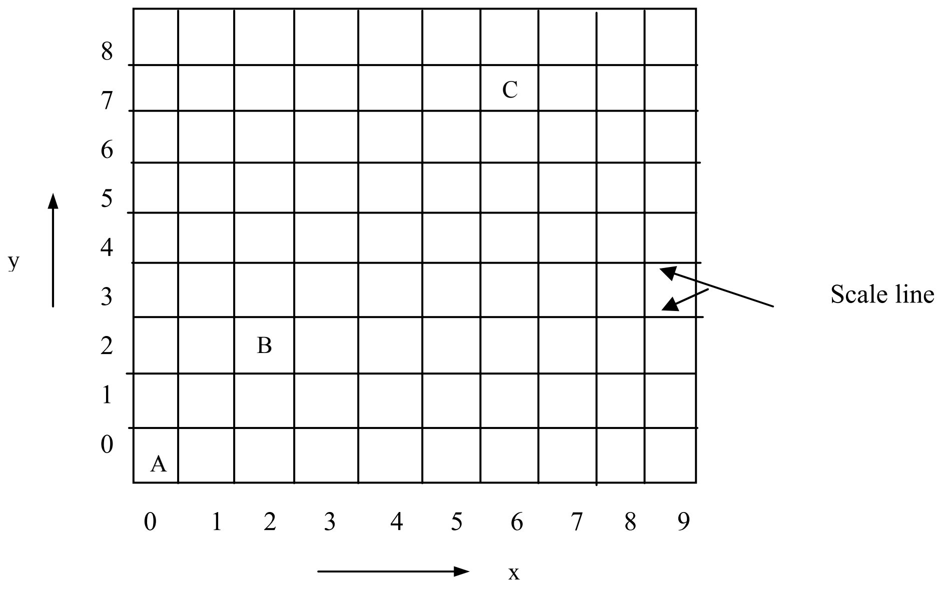


Figure 1: Scan lines

Figure 1, shows the Array of square pixels on the display surface. Coordinate of pixel A: 0, 0; B: 2, 2; C: 6, 7. A coordinate position (6.26, 7.25) is represented by C, whereas (2.3, 2.5) is represented by B. Because, in order to plot a pixel on the screen, we need to round off the coordinates to a nearest integer. Further, we need to say that, it is this rounding off, which leads to distortion of any graphic image.

Line drawing is accomplished by calculating the intermediate point coordinates along the line path between two given end points. Since, screen pixels are referred with integer values, plotted positions may only approximate the calculated coordinates – i.e., pixels which are intensified are those which lie very close to the line path if not exactly on the line path which is in the case of perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line, we will discuss such algorithms in our next section. Screen resolution however, is a big factor towards improving the approximation. In a high resolution system the adjacent pixels are so closely spaced that the approximated line-pixels lie very close to the actual line path and hence, the plotted lines appear to be much smoother — almost like straight lines drawn on paper. In a low resolution system, the same approximation technique causes lines to be displayed with a "stairstep apperance" i.e., not smooth as shown in *Figure 2*, the effect is known as the stair case effect. We will discuss this effect and the reason behind this defect in the next section of this unit.

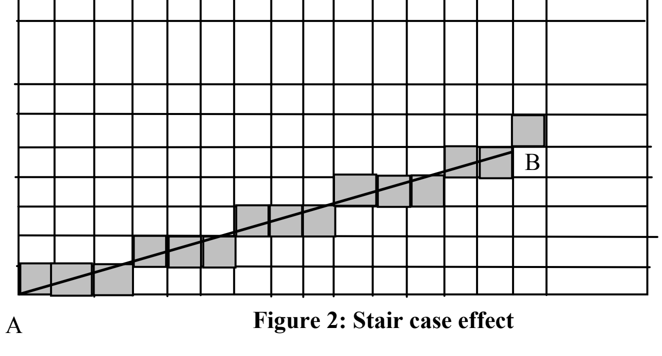


Figure 2: Stair case effect

2.4 LINE GENERATION ALGORITHMS

- In unit 1, we have discussed the case of frame buffer where information about the image to be projected on the screen is stored in an *m*n* matrix, in the form of 0s and 1s; the 1s stored in an *m* n* matrix positions are brightened on the screen and 0's are not brightened on the screen and this section which may or may not be brightened is known as the Pixel (picture element). This information of 0s and 1s gives the required pattern on the output screen i.e., for display of information. In such a buffer, the screen is also in the form of *m* n* matrix , where each section or niche is a pixel (i.e., we have *m* n* pixels to constitute the output).

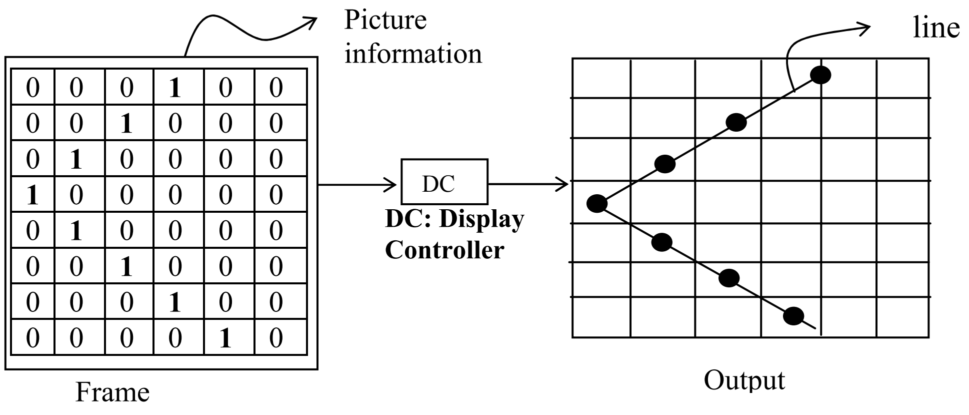


Figure 3: Basic Line Generation

Now, it is to be noted that the creation of a line is merely not restricted to the above pattern, because, sometimes the line may have a slope and intercept that its information is required to be stored in more than one section of the frame buffer, so in order to draw or to approximate such the line, two or more pixels are to be made ON. Thus, the outcome of the line information in the frame buffer is displayed as a stair; this effect of having two or more pixels ON to approximating a line between two points say A and B is known as the Staircase effect. **The concept is shown below in *Figure 4.***

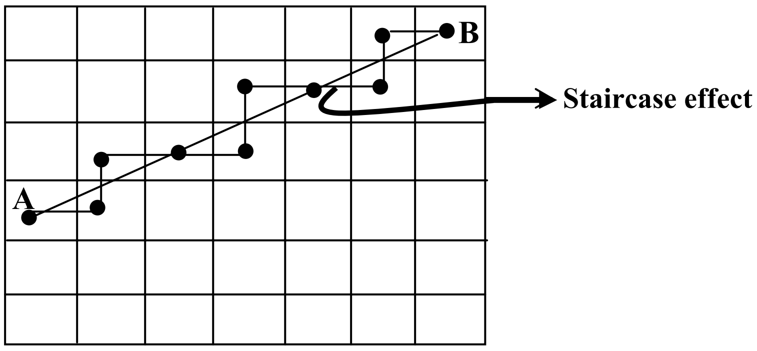


Figure 4: Staircase effect

So, from the *Figure 4*, I think, it is clear to you that when a line to be drawn is simply described by its end points, then it can be plotted by making close approximations of the pixels which best suit the line, and this approximation is responsible for the staircase effect, which miss projects the information of the geometry of line stored in the frame buffer as a stair. This defect known as Staircase effect is prominent in DDA Line generation algorithms, thus, to remove the defect Bresenham line generation Algorithm was introduced. We are going to discuss DDA (Digital Differential Analyser) Algorithm and Bresenham line generation Algorithm next.

2.4.1 DDA (Digital Differential Analyser) Algorithm

From the above discussion we know that a Line drawing is accomplished by calculating intermediate point coordinates along the line path between two given end points. Since screen pixels are referred with integer values, or plotted positions, which may only approximate the calculated coordinates – i.e., pixels which are intensified are those which lie very close to the line path if not exactly on the line path which in this case are perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line, one such algorithm is the DDA (Digital Differential Analyser) algorithm. Before going to the details of the algorithm, let us discuss some general appearances of the line segment, because the respective appearance decides which pixels are to be intensified. It is also obvious that only those pixels that lie very close to the line path are to be intensified because they are the ones which best approximate the line. Apart from the exact situation of the line path, which in this case are perfectly horizontal, vertical or 45° lines (i.e., slope zero, infinite, one) only. We may also face a situation where the slope of the line is > 1 or < 1.Which is the case shown in *Figure 5*.

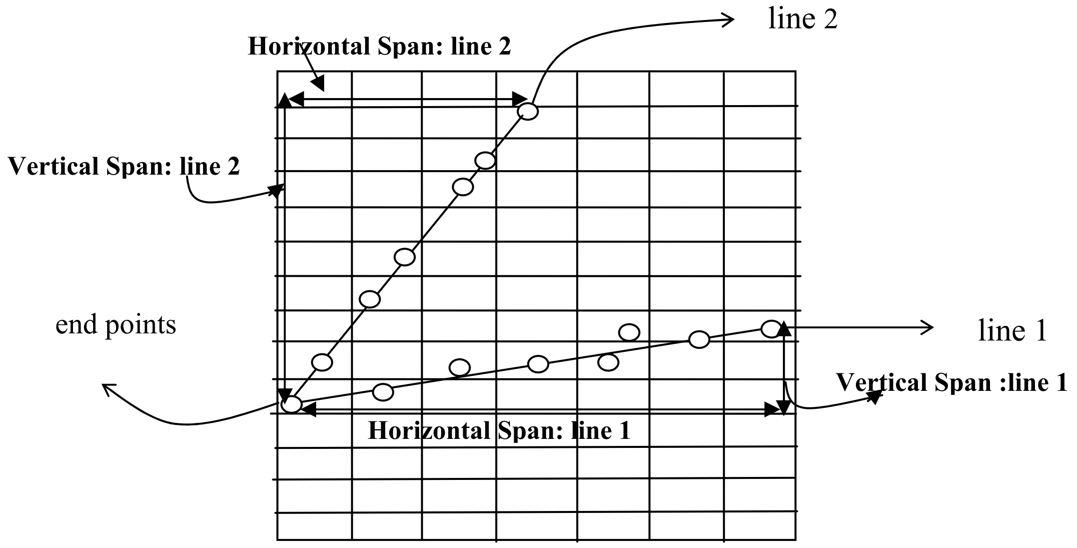


Figure 5: DDA line generation

In *Figure 5*, there are two lines. Line 1 (slope<1) and line 2 (slope>1). Now let us discuss the general mechanism of construction of these two lines with the DDA algorithm. As the slope of the line is a crucial factor in its construction, let us consider the algorithm in two cases depending on the slope of the line whether it is > 1 or < 1.

Case 1: slope (*m*) of line is < 1 (i.e., line 1): In this case to plot the line we have to move the direction of pixel in x by 1 unit every time and then hunt for the pixel value of the y direction which best suits the line and lighten that pixel in order to plot the line.

So, in Case 1 i.e., $0 < m < 1$ where x is to be increased then by 1 unit every time and proper y is approximated.

Case 2: slope (*m*) of line is > 1 (i.e., line 2) if $m > 1$ *i.e.*, case of line 2, then the most appropriate strategy would be to move towards the y direction by 1 unit every time and determine the pixel in x direction which best suits the line and get that pixel lightened to plot the line.

So, in Case 2, i.e., (infinity) > m > 1 where y is to be increased by 1 unit every time and proper x is approximated.

Assumption: The line generation through DDA is discussed only for the Ist Quadrant, if the line lies in any other quadrant then apply respective transformation (generally reflection transformation), such that it lies in Ist Quadrant and then proceed with the algorithm, also make intercept Zero by translational transformation such that (x_i, y_i) resolves to (x_i, $mx_i + c$) or (x_i, mx_i) and similar simplification occurs for other cases of line generation. The concept and application of transformations is discussed in Block 2 of this course.

Note:

1) If in case 1, we plot the line the other way round i.e., moving in y direction by 1 unit every time and then hunting for x direction pixel which best suits the line. In this case, every time we look for the x pixel, it will provide more than one choice of pixel and thus enhances the defect of the stair case effect in line generation. Additionally, from the *Figure 5*, you may notice that in the other way round strategy for plotting line 1, the vertical span is quite less in comparison to the horizontal span. Thus, a lesser number of pixels are to be made *ON*, and will be available if we increase *Y* in unit step and approximate *X*. But more pixels will be available if we increase *X* in unit steps and approximate *Y* (this choice will also reduce staircase effect distortion in line generation) ($\because$ more motion is to be made along x-axis).

2) Consider a line to be generated from (X_0, Y_0) to (X_1, Y_1). If $(X_1 - X_0 > Y_1 - Y_0)$ and $X_1 - X_0 > 0$ then slope (m) of line is < 1 hence case 1 for line generation is applicable otherwise case 2, i.e., If $(X_1 - X_0 < Y_1 - Y_0)$ and $X_1 - X_0 > 0$ then slope m > 1 and case 2 of line generation is applicable.
Important: Assume that $X_1 > X_0$ is true else swap (X_0, Y_0) and (X_1, Y_1)

3) When $0 < m < 1$: increment X in unit steps and approximate Y
   - Unit increment should be iterative $\Rightarrow x_i = (x_{i-1}) + 1$ such that (x_i, y_i) resolves to (x_i, $mx_i + c$) or (x_i, mx_i) . *It is to be noted that while calculating y_i, if y_i turned out to be a floating number then we round its value to select the approximating pixel. This rounding off feature contributes to the staircase effect.*

   When (infinity) > m > 1 increment Y in unit steps and approximate X, simplify (x_i, y_i) accordingly.

Case 1: slope (m) of line is < 1 (or 0 < m < 1)

Consider a line to be generated from *(X_0, Y_0)* to *(X_1, Y_1)* , **Assume that *$X_1 > X_0$* is true else swap *(X_0, Y_0)* and *(X_1, Y_1)*.** Now, if $(X_1 - X_0 > Y_1 - Y_0)$ that means slope (m) of line is < 1 hence, case 1 for line generation is applicable. Thus, we need to increment X in unit steps and approximate Y. So, from X_0 to X_1 , x_i is incremented in unit steps in the horizontal direction, now for these unit steps the satisfying value of Y can be estimated by the general equation of line $y = mx + c$.

Similarly, for case 2, let us sum up our discussion on DDA algorithm for both cases.

We will examine each case separately.

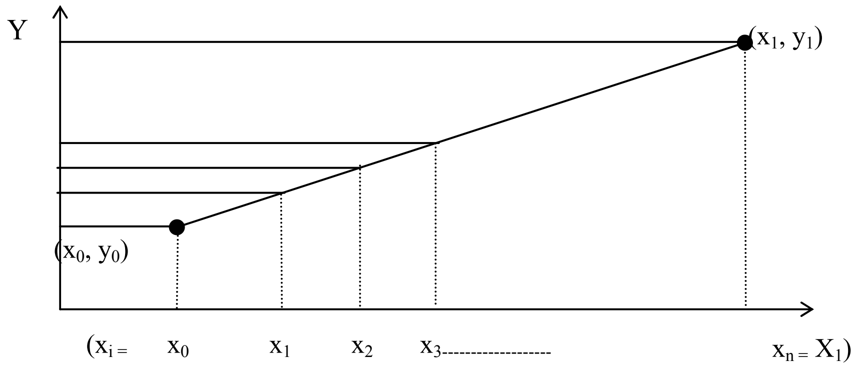


Figure 6: Slope (*m*) of line is < 1 (i.e., line 1)

Sum up:

For instance, in a given line the end points are (X_0, Y_0) and (X_1, Y_1). Using these end points we find the slope ($m = Y_1 - Y_0/ X_1 - X_0$) and check that the value of *m* lies between 0 and 1 or is > 1. If 0 < m < 1 then case 1 applies, else, case 2 applies.
For case 1, increase *x* by one Unit every time, for case 2 increase *y* by one Unit every time and approximate respective values of *y* and *x*.

We assume equation of line is $y = mx+c$

$$\left.\begin{array}{lll} \text{At } x = x_i & \text{we have} & y_i = mx_i+c \\ \text{Similarly at } x = x_{i+1} & \text{we have} & y_{i+1} = mx_{i+1}+c \end{array}\right\}$$

Case 1: Slope (m) of line is 0 < *m*1 (i.e., line 1)

Since *x* is to be increase by 1 unit each time

$\Rightarrow x_{i+1} = x_i + 1$ ----------- (1)

So by using equation of line $y = mx+c$ we have

$$\begin{aligned} y_{i+1} &= m(x_i + 1) + c \\ &= mx_i + c + m \\ &= y_i + m \qquad \text{------------ (2)} \end{aligned}$$

Equations (1) and (2) imply that to approximate line for case 1 we have to move along *x* direction by 1 unit to have next value of *x* and we have to add slope *m* to initial *y* value to get next value of *y*.

Now, using the starting point (x_0, y_0) in the above equations (1) and (2) we go for x_i and y_i (i = 1, 2, ……n) and put colour to the pixel to be lightened.

It is assumed that $X_0 < X_1$ ∴ the algorithm goes like:

```
          $x \leftarrow X_0$
          $y \leftarrow Y_0$
          $m \leftarrow (Y_1 - Y_0)/ (X_1 - X_0)$
     while ($x <= X_1$) do
{    put-pixel ($x$, round ($y$), color)
     (new $x$-value) $x \leftarrow$     (old $x$-value) $x + 1$
     (new $y$-axis)  $y \leftarrow$     (old $y$-value) $y + m$
}
```

Sample execution of algorithm case 1:

at (x_0, y_0) : put-pixel (x_0, y_0, colour)

$x_1 = x_0 + 1$; $y_1 = y_0 + m$

at (x_1, y_1) = put pixel $(x_1, y_1$, colour)

similarly, $x_2 = x_{1+}1$; $y_2 = y_{1+}$m
at (x_2, y_2) : put pixel $(x_2, y_2$, colour) and so on.

Case 2: slope (*m*) of line is > 1 (i.e., line 2): Same as case 1 but, this time, the y component is increased by one unit each time and appropriate *x* component is to be selected. To do the task of appropriate selection of *x* component we use the equation of Line: $y = mx+c$.

Unit increment should be iterative $\Rightarrow y_{i+1} = y_i + 1$; for this y_{i+1} we find corresponding x_{i+1} by using equation of line $y = mx + c$ and hence get next points (x_{i+1}, y_{i+1}).

$\Rightarrow y_{i+1} - y_i = m\,(x_{i+1} - x_i)$ ----------- (3)

as y is to be increase by unit steps
$\therefore y_{i+1} - y_i = 1$ ----------- (4)

using equation (4) in (3) we get
$\Rightarrow 1 = m\,(x_{i+1} - x_i)$ ----------- (5)

rearranging (5) we get

$\Rightarrow 1/m = (x_{i+1} - x_i)$

$$\Rightarrow \boxed{x_{i+1} = x_i + \frac{1}{m}}$$

So, procedure as a whole for Case 2 is summed up as Algorithm case 2:

Assuming $Y_0 < Y_1$ the algorithm goes like

Algorithm for case 2:

```
        $x \leftarrow X_0$;
        $y \leftarrow Y_0$;
        $m \leftarrow (Y_1 - Y_0)/(X_1 - X_0)$;
        $m_1 \leftarrow 1/m$;
    while ($y < Y_1$) do
{
    put-pixel (round ($x$), $y$, colour)
    $y \leftarrow y + 1$;
    $x \leftarrow x + m_1$;
X
}
```

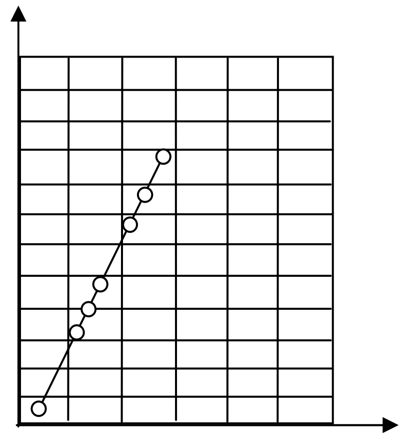


Figure 7: Slope (*m*) of line is > 1

Example 1: Draw line segment from point (2, 4) to (9, 9) using DDA algorithm.

Solution: We know general equation of line is given by

$y = mx+c$ where $m = (y_1 - y_0/(x_1 - x_0)$

given $(x_0, y_0) \rightarrow (2, 4)$; $(x_1, y_1) \rightarrow (9, 9)$

$$\Rightarrow m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{9-4}{9-2} = \frac{5}{7} \qquad \text{i.e., } 0 < m < 1$$

$$C = y_1 - mx_1 = 9 - \frac{5}{7}\times 9 = \frac{63-45}{7} = \frac{18}{7}$$

So, by equation of line *(y = mx + c)* we have

$$y = \frac{5}{7}x + \frac{18}{7}$$

DDA Algorithm Two case:	
Case 1: $m < 1$	$x_{i+1} = x_i + 1$
	$y_{i+1} = y_i + m$
Case 2: $m > 1$	$x_{i+1} = x_i + (1/m)$
	$y_{i+1} = y_i + 1$

As $0 < m < 1$ so according to DDA algorithm case 1

$x_{i+1} = x_i + 1$ $\qquad$ $y_{i+1} = y_i + m$

given $(x_0, y_0) = (2, 4)$

1) $x_1 = x_0 + 1 = 3$

$$y_1 = y_0 + m = 4 + \frac{5}{7} = \frac{28+5}{7} = \frac{33}{7} = 4\frac{5}{7}$$

put pixel (x_0, round y, colour)

i.e., put on (3, 5)

2) $x_2 = x_1 + 1 = 3 + 1 = 4$

$$y_2 = y_1 + m = (33/7) + \frac{5}{7} = 38/7 = 5\frac{5}{7}$$

put on (4, 5)

Similarly go on till (9, 9) is reached.

☞ Check Your Progress 1

1) What do you mean by the term graphic primitives?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

2) What is the Staircase effect?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

3) What is the reason behind the Staircase effect?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

4) How does the resolution of a system affect graphic display?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

5) Modify the DDA algorithm for negative sloped lines ; discuss both the cases i.e., slope > 1 and 0 < slope < 1.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

6) Using DDA algorithm draw line segments from point (1,1) to (9,7).

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

2.4.2 Bresenham Line Generation Algorithm

Bresenham algorithm is accurate and efficient raster line generation algorithm. This algorithm scan converts lines using only incremental integer calculations and these calculations can also be adopted to display circles and other curves.

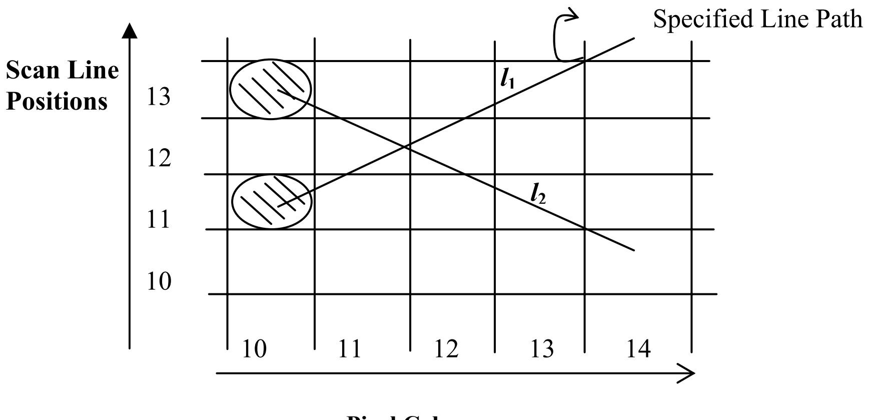


Figure 8: Bresenham line generation

Sampling at Unit *x* distance in *Figure 8*, we need to decide which of the two possible pixel position is closer to the line path at each sample step.

In l_1 : we need to decide that at next sample position whether to plot the pixel at position (11, 11) or at (11, 12).

Similarly, **In l_2**: the next pixel has to be (11, 13) or (11, 12) or what choice of pixel is to be made to draw a line is given by Bresenham, by testing the sign of the integer parameter whose value is proportional to the difference between the separation of the two pixel positions from actual line path. In this section, we will discuss the Bresenham line drawing algorithm for + ve slope ($0 < m < 1$). If the slope is negative then, use reflection transformation to transform the line segment with negative slope to line segment with positive slope. Now, let us discuss the generation of line again in two situations as in the case of DDA line generation.

Case 1: Scan Conversion of Line with the slope ($0 < m < 1$)

Here the pixel positions along the line path are determined by sampling at Unit x intervals *i.e.,* starting from the initial point. (x_0, y_0) of a given line we step to each successive column. (*i.e.*, x-position) and plot the pixel whose scanline y value is closest to the line path. Assume we proceed in this fashion up to the kth step. The process is shown in *Figure 9*. Assuming we have determined the pixel at (x_k, y_k). We need to decide which pixel is to be plotted in column x_{k+1}. Our choices are either (x_{k+1}, y_k) or (x_{k+1}, y_{k+1}).

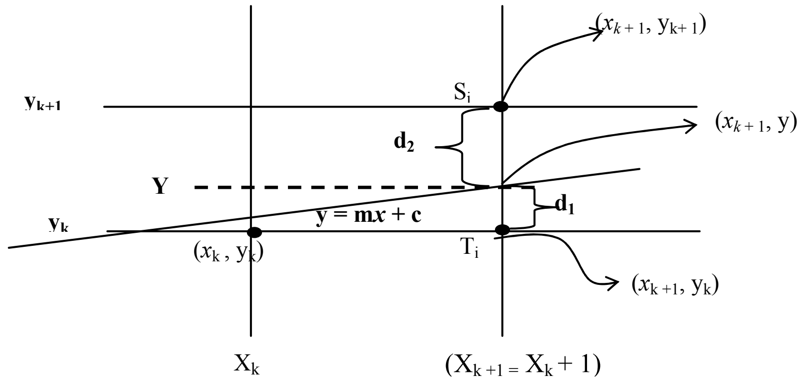


Figure 9: Scan conversion $0 < m < 1$

At sampling position X_{k+1} the vertical pixel (or scan line) separation from mathematical line ($y = mx + c$) is say d_1 and d_2.

Now, the y coordinate on the mathematical line at pixel column position X_{k+1} is:

$y = m(x_{k+1}) + c$ ---------------------(1)

by *Figure 9*:

$d_1 = y - y_k$ ---------------------(2)

$= m(x_{k+1}) + c - y_k$ ---------------------(3) (using (1))

similarly, $d_2 = (y_{k+1}) - y = y_{k+} 1 - m(x_{k+1}) - c$ ---------------------(4)

using (3) and (4) we find $d_1 - d_2$

$d_1 - d_2 = [m(x_{k+} 1) + c - y_k] - [y_k + 1 - m(x_{k+} 1) - c]$

$= mx_k + m + c - y_k - y_k - 1 + mx_k + m + c$

$= 2m(x_k + 1) - 2y_k + 2c - 1$ ---------------------(5)

Say, decision parameter p for kth step i.e., p_k is given by

$p_k = \Delta x(d_1 - d_2)$ ---------------------(6)

Now, a decision parameter p_k for the kth step in the line algorithm can be obtained by rearranging (5) such that it involves only integer calculations. To accomplish this substitute $m = \Delta y/\Delta x$ where, Δy and $\Delta x \Rightarrow$ vertical and horizontal separations of the end point positions.

$p_k = \Delta x(d_1 - d_2) = \Delta x[2m(x_k + 1) - 2y_k + 2c - 1]$

$= \Delta x[2(\Delta y/\Delta x)(x_k + 1) - 2y_k + 2c - 1]$

$= 2\Delta y(x_k + 1) - 2\Delta xy_k + 2\Delta xc - \Delta x$

$= 2\Delta y\, x_k - 2\Delta x\, y_k + [2\Delta y + \Delta x(2c - 1)]$

$p_k = 2\Delta y\, x_k - 2\Delta xy_k + b$ -------------------- (7)

where b is constant with value $b = 2\Delta y + \Delta x(2c - 1)$ --------------------(8)

Note: *sign of* p_k is same as sign of $d_1 - d_2$ since it is assumed that $\Delta x > 0$
[As in *Figure 9*, $d_1 < d_2$ *i.e.* $(d_1 - d_2)$ is –ve i.e., p_k is –ve so pixel T_i is more appropriate choice otherwise pixel S_i is the appropriate choice.
i.e., (1) if $p_k < 0$ choose T_i , so next pixel choice after (x_k, y_k) is (x_{k+1}, y_k)
else (2) if $p_k > 0$ choose S_i , so next pixel choice after (x_k, y_k) is (x_{k+1}, y_{k+1}).

At step $k + 1$ the decision parameter is evaluated by writing (7) as:

$$p_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + b \qquad \text{---------------------(9)}$$

Subtracting (7) from (9) we get

$$p_{k+1} - p_k = 2\Delta y\underbrace{(x_{k+1} - x_k)}_{1} - 2\Delta x\,\underbrace{(y_{k+1} - y_k)}_{\textbf{0 or 1}} = 2\Delta y - 2\Delta x\,(y_{k+1} - y_k)$$

$$\text{depending on sign of } p_k \quad \begin{bmatrix} p_k < 0 \;\; y_{k+1} = y_k \\ p_k > 0 \, y_{k+1} = y_k + 1 \end{bmatrix}$$

$$\boxed{\qquad\qquad\qquad\qquad} \qquad \text{---------------------(10)}$$

This recursive calculation of decision parameter is preformed at each integer position, beginning with the left coordinate end point of line.

This first parameter p_0 is determined by using equation (7), and (8) at the starting pixel position (x_0, y_0) with m evaluated as $\Delta y / \Delta x$ (*i.e.,* intercept $c = 0$)

$$p_0 = 0 - 0 + b = 2\Delta y + \Delta x\,(2 * 0 - 1) = 2\Delta y - \Delta x$$
$$p_0 = 2\Delta y - \Delta x \qquad \text{-----------------------(10 A)}$$

Summary:

(Bresenham line drawing algorithm for +ve slope ($0 < m < 1$).

- If slope is negative then use reflection transformation to transform the line segment with negative slope to line segment with a positive slope.

- Calculate constants $2\Delta y$, $2\Delta y - 2\Delta x$, Δy, Δx at once for each line to be scan converted, so the arithmetic involves only integer addition/subtraction of these constants.

Remark : The algorithm will be exactly the same if we assume $| m | < 1$

- **Algorithm $| m | < 1$:**
  (a) Input two line end points and store left end point in (x_0, y_0)
  (b) Load (x_0, y_0) on frame buffer *i.e.,* plot the first point.
  (c) Calculate Δx, Δy, $2\Delta y$, $2\Delta y - 2\Delta x$ and obtain the starting value of decision parameter as $p_0 = 2\Delta y - \Delta x$
  (d) At each x_k along the line, starting at $k = 0$, perform following test:

  if $p_k < 0$, the next plot is (x_{k+1}, y_k) and $p_{k+1} = p_k + 2\Delta y$
  else next plot is (x_{k+1}, y_{k+1}) and $p_{k+1} = p_k + 2(\Delta y - \Delta x)$

  (e) Repeat step (D) Δx times.

Bresenham Line Generation Algorithm (| m | < 1)

```
$\Delta x \leftarrow x_1 - x_0$
$\Delta y \leftarrow y_1 - y_0$
$p_0 \leftarrow 2\Delta y - \Delta x$
```

```
while ($x_0$ < = $x_1$) do
        {puton ($x_0$, $y_0$)
        if (pi > 0) then
        {$x_0 \leftarrow x_0 + 1$;
           $y_0 \leftarrow y_0 + 1$;
        $p_{i+1} \leftarrow pi + 2\ (\Delta y - \Delta x)$;

        }
     if (pi < 0) then
     {$x_0 \leftarrow x_0 + 1$
      $y_0 \leftarrow y_0$
   $p_{i+1} \leftarrow pi + 2\ \Delta y$
     }
    }
```

Note:

- Bresenhams algorithm is generalised to lines with arbitrary slopes by considering the symmetry between the various octants and quadrants of the coordinate system.
- For line with +ve slope (m) such that m > 1, then we interchange the roles of *x* and *y* direction i.e., we step along *y* directions in unit steps and calculate successive *x* values nearest the line path.
- for –ve slopes the procedures are similar except that now, one coordinate decreases as the other increases.

Example 2: Draw line segment joining (20, 10) and (25, 14) using Bresenham line generation algorithm.

Solution: $(x_0, y_0) \rightarrow (20, 10)$; $(x_1, y_1) \rightarrow (25, 14)$

$$m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{14-10}{25-20} = \frac{4}{5} < 1$$

As, $m = \frac{\Delta y}{\Delta x} = \frac{4}{5} \Rightarrow \Delta y = 4$

$\Delta x = 5$

$\rightarrow$ plot point (20, 10)

$p_i = 2\Delta y - \Delta x$

$i = 1$: $p_i = 2 * 4 - 5 = 3$

as $p_1 > 0$ *so* $x_0 \leftarrow 21;\ y_0 \leftarrow 11$

now plot (21,11)

$i = 2$ *as* $p_1 > 0$

$\therefore p_2 = p_1 + 2(\Delta y - \Delta x)$

$= 3 + 2\ (4-5) = 3 - 2 = 1$

$p_2 > 0$ *so* $x_0 \leftarrow 22;\ y_0 \leftarrow 12$

plot (22,12)

$i = 3$ *as* $p_2 > 0$

$\therefore p_3 = p_2 + 2\ (\Delta y - \Delta x) = 1 + 2\ (4 - 5) = -1$

$p_3 < 0$ $\therefore x_0 \leftarrow 23$

$y_0 \leftarrow 12$

plot (23, 12)

$i = 4$ *as* $p_3 < 0$

$\therefore p_4 = p_3 + 2\Delta y$

$= -1 + 2 * 4 = 7$

$\therefore x_0 \leftarrow 24; y_0 \leftarrow 13$

plot (24, 13)

$i = 5$ *as* $p_4 > 0$

$\therefore p_5 = p_4 + 2(\Delta y - \Delta x)$

$= 7 + 2(4 - 5) = 5$

$x_0 \leftarrow 25; y_0 \leftarrow 14$

plot (25, 14)

{for i = 6, x_0 will be > x_i so algorithm terminates

Example 3: Illustrate the Bresenham line generation algorithm by digitzing the line with end points (20, 10) and (30,18)

Solution: $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{18-10}{30-20} = \frac{8}{10} = 0.8$ -------------------(1)

$\Rightarrow \Delta y = 8$ and $\Delta x = 10$ -------------------(2)

value of initial decision parameter (p_0) = $2\Delta y - \Delta x = 2 * 8 - 10 = 6$ -----------------(3)

value of increments for calculating successive decision parameters are:

$2\Delta y = 2 * 8 = 16;$ -------------------(4)

$2\Delta y - 2\Delta x = 2 * 8 - 2 * 10 = -4$ -------------------(5)

plot initial point (x_0, y_0) = (20, 10) in frame buffer now determine successive pixel positions along line path from decision parameters value (20, 10).

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)

← [use step (d) of algorithm Δx times]

If $p_k > 0$ then increase both X and Y and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$

If $p_k < 0$ then increase X and not Y and $p_{k+1} = p_k + 2\Delta y$

☞ Check Your Progress 2

1) Compare Bresenham line generation with DDA line generation.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

2) Illustrate the Bresenham line generation algorithm by digitising the line with end points (15, 5) and (25,13).

……………………………………………………………………………………

……………………………………………………………………………………

2.5 CIRCLE-GENERATION ALGORITHMS

Circle is one of the basic graphic component, so in order to understand its generation, let us go through its properties first:

2.5.1 Properties of Circles

In spite of using the Bresenham circle generation (i.e., incremental approval on basis of decision parameters) we could use basic properties of circle for its generation.

These ways are discussed below:

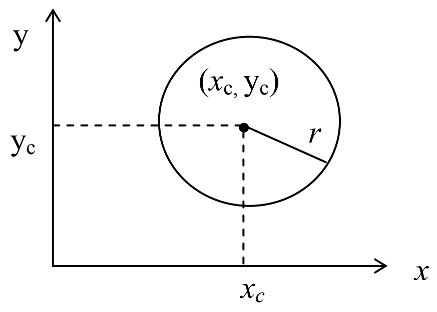


Figure 8: Property of circle

Generation on the basis of Cartesian Coordinates:

A circle is defined as the set of points or locus of all the points, which exist at a given distance r from center (x_c, y_c). The distance relationship could be expressed by using Pythagonous theorem in Cartesian coordinates as

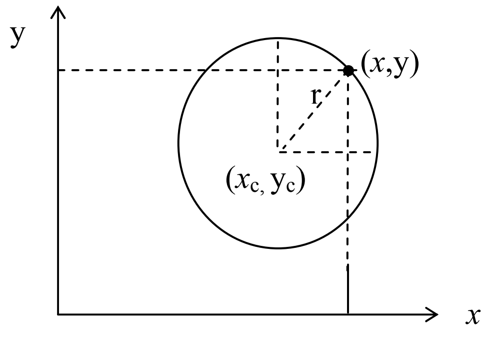


Figure 9: Circle generation (cartesian coordinate system)

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \qquad \text{--------------------(1)}$$

Now, using (1) we can calculate points on the circumference by stepping along x-axis from x_c–r to $x_c + r$ and calculating respective y values for each position.

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

Generation on basis of polar coordinates (*r* and θ)

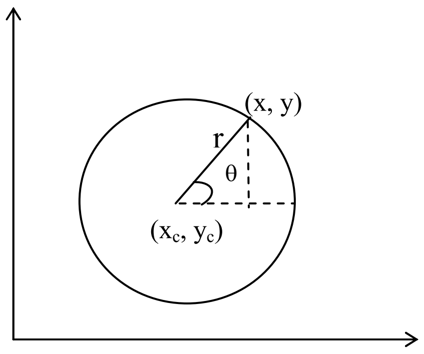


Figure 10: Circle generation (polar coordinate system)

Expressing the circle equation in parametric polar form yields the following equations

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

$$y = y_c + r\sin\theta \qquad \text{-------------------------(3)}$$

Using a fixed angular step size we can plot a circle with equally spaced points along the circumference.

Note:

- *Generation of circle with the help of the two ways mentioned is not a good choice ∵ both require a lot of computation equation (2) requires square root and multiplication calculations where as equation (3) requires trigonometric and multiplication calculations. A more efficient approach is based on incremental calculations of decision parameter. One such approach is Bresenham circle generation. (mid point circle algorithm*).

- This *Bresenham circle generation (mid point circle algorithm*) is similar to line generation. Sample pixels at Unit *x* intervals and determine the closest pixel to the specified circle path at each step.

- For a given radius and center position (*x*, *y*,) we first setup our algorithm to calculate pixel position around the path of circle centered at coordinate origin (0, 0) *i.e.,* we translate $(x_c, y_c) \rightarrow (0, 0)$ and after the generation we do inverse translation $(0, 0) \rightarrow (x_c, y_c)$ hence each calculated position (x, y) of circumference is moved to its proper screen position by adding x_c to x and y_c to y.

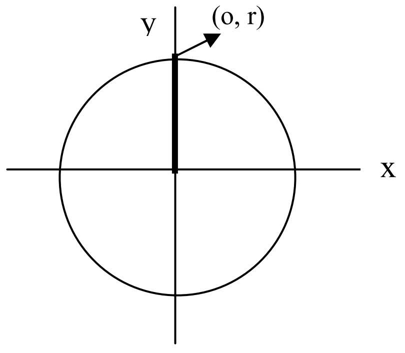


Figure 11: Circle generation (initialisation)

- In *Bresenham circle generation (mid point circle algorithm*) we calculate points in an octant/quadrant, and then by using symmetry we find other respective points on the circumference.

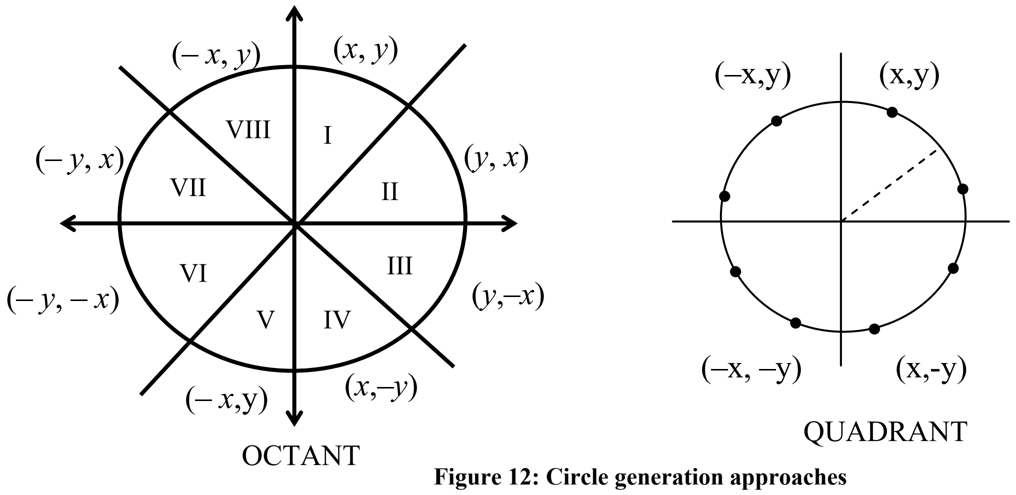


Figure 12: Circle generation approaches

2.5.2 Mid Point Circle Generation Algorithm

We know that equation of circle is $x^2 + y^2 = r^2$. By rearranging this equation, we can have the function, to be considered for generation of circle as $f_c\ (x, y)$.

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

The position of any point (x, y) can be analysed by using the sign of $f_c\ (x, y)$ i.e.,

$$\text{decision parameter} \rightarrow f_c\ (x, y) = \begin{cases} < 0 \text{ if } (x, y) \text{ is inside circle boundary} \\ = 0 \text{ if } (x, y) \text{ is on the circle boundary} \\ > 0 \text{ if } (x, y) \text{ is outside circle boundary} \end{cases} \qquad \text{---------(2)}$$

i.e., we need to test the sign of $f_c\ (x, y)$ are performed for mid point between pixels near the circle path at each sampling step.

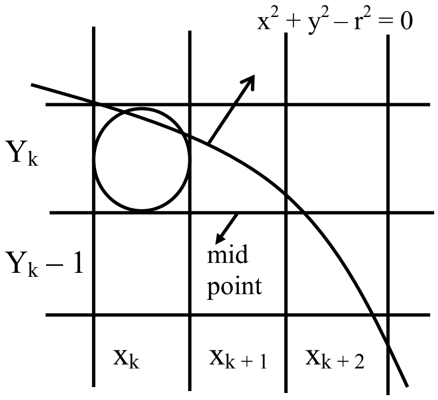


Figure 13: Mid point between Candidate Pixel at sampling Position x_{k+1} along Circular Path

Figure 13 shows the mid point of two candidate pixel (x_{k+1}, y_k) and (x_{k+1}, y_{k-1}) at sampling position $x_k + 1$.

Assuming we have just plotted the (x_k, y_k) pixel, now, we need to determine the next pixel to be plotted at (x_{k+1}, y_k) or (x_{k+1}, y_{k-1}). In order to decide the pixel on the circles path, we would need to evaluate our decision parameter p_k at mid point between these two pixels i.e.,

$$p_k = f_c\ (x_{k+1}, y_k - \frac{1}{2}) = (x_{k+1})^2 + (y_k - \frac{1}{2})^2 - r^2 \qquad \text{---------------------(3)}$$

(using (1)

If $p_k < 0$ then $\Rightarrow$ midpoint lies inside the circle and pixel on scan line y_k is closer to circle boundary else mid point is outside or on the circle boundary and we select pixel on scan line $y_k - 1$ successive decision parameters are obtained by using incremental calculations.

The recursive way of deciding parameters by evaluating the circle function at sampling positions $x_{k+1} + 1 = (x_k + 1) + 1 = x_k + 2$

$$p_{k+1} = f_c\left(x_{k+1} + 1,\ y_{k+1} - \frac{1}{2}\right) = \left[(x_k + 1) + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2\right] \text{--------(4)}$$

subtract equation (3) from (4) we get

$$p_{k+1} - p_k = \left\{[(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2\right\} - \left\{(x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2\right\}$$

$$p_{k+1} - p_k = [(x_k + 1)^2 + 1 + 2(x_k + 1).\ 1] + \left[y^2_{k+1} + \frac{1}{4} - \cancel{2}.\frac{1}{\cancel{2}}.\ y_{k+1}\right] - \cancel{r^2} - (x_k + 1)^2$$

$$- \left[y_k^2 + \frac{1}{4} - \cancel{2}.\ \frac{1}{\cancel{2}}.\ y_k\right] + \cancel{r^2}$$

$$= \cancel{(x_k + 1)^2} + 1 + 2\,(x_k + 1) + y^2_{k+1} + \cancel{\tfrac{1}{4}} - y_{k+1} - \cancel{(x_k + 1)^2} - y_k^2 - \cancel{\tfrac{1}{4}} + y_k$$

$$p_{k+1} = p_k + 2\,(x_k + 1) + (y^2_{k+1} - y_k^2) - (y_{k+1} - y_k) + 1 \qquad \text{--------------(5)}$$

Here, y_{k+1} could be y_k or y_{k-1} depending on sign of p_k

$$\left[\begin{array}{l} \text{so, increments for obtaining } p_{k+1} \text{ are : -} \\ \text{either } 2x_{k+1} + 1 \,(\text{if } p_k < 0)\,(i.e., y_{k+1} = y_k) \\ or\ 2x_{k+1} + 1 - 2y_{k-1} (\text{if } p_k > 0)\,(i.e., y_{k+1} = y_k - 1)\ So(2\,y_{k+1} = 2(y_k - 1) = 2y_k - 2 \end{array}\right.$$

How do these increments occurs?

- $y_{k+1} = y_k$: use it in (5) we get
  $p_{k+1} = p_k + 2\,(x_k + 1) + 0 - 0 + 1 = p_k + (2x_{k+1} + 1)$

$\Rightarrow$ $\boxed{}$ ----------------(6)

- $y_{k+1} = y_{k-1}$ use it in (5) we get
  $$p_{k+1} = p_k + 2x_{k+1} + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$
  $$= p_k + 2x_{k+1} + \frac{(y_{k+1} - y_k)}{}\ \frac{[(y_{k+1} + y_k) - 1]}{} + 1$$

$$\boxed{p_{k+1} = p_k + (2x_{k+1} - 2y_{k-1} + 1)} \qquad \text{------------------(7)}$$

Also,

$$2x_{k+1} = 2\,(x_k + 1) = 2x_k + 2 \qquad \text{------------------ (8)}$$

$$2y_{k+1} \rightarrow 2y_{k-1} = 2\,(y_k - 1) = 2y_k - 2 \qquad \text{-------------------(9)}$$

Note:

- At starting position $(0, r)$ the terms in (8) and (9) have value 0 and $2r$ respectively. Each successive value is obtained by adding 2 to the previous value of $2x$ and subtracting 2 from the previous value of $2y$.

- The initial decision parameter is obtained by evaluating the circle function at start position $(x_0, y_0) = (0, r)$

  i.e., $p_0 = f_c\left(1, r - \frac{1}{2}\right) = 1 + \left(r - \frac{1}{2}\right)^2 - r^2 = \frac{5}{4} - r$ (using 1)

  If radius r is specified as an integer then we can round p_0 to

  $\boxed{p_0 = 1 - r}$ (for r as an integer).

Midpoint Circle Algorithm

(a) Input radius r and circle, center (x_c, y_c) and obtain the first point on the circumference of the circle centered on origin as

$$(x_0, y_0) = (0, r)$$

(b) Calculate initial value of decision parameter as

$$p_0 = \frac{5}{4} - r \sim 1 - r$$

(c) At each x_k position starting at $k = 0$ perform following test.
   1) If $p_k < 0$ then next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and $p_{k+1} = p_k + 2x_{k+1} + 1$
   2) Else the next point along circle is $(x_{k+1}, y_k - 1)$ and
      $p_{k+1} = p_k + 2x_{k+1} - 2y_{k-1}$
      where $2x_{k+1} = 2(x_k + 1) = 2x_k + 2$ and $2y_{k-1} = 2(y_k - 1) = 2y_k - 2$

(d) Determine symmetry points in the other seven octants.

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

(f) Repeat step (c) through (e) until $x \geq y$.

Example 4: Given a circle radius $r = 10$ determine positions along the circle octants in 1st Quadrant from $x = 0$ to $x = y$.

Solution: An initial decision parameter $p_0 = 1 - r = 1 - 10 = -9$
For circle centered on coordinate origin the initial point $(x_0, y_0) = (0, 10)$ and initial increment for calculating decision parameter are:

$$2x_0 = 0, 2y_0 = 20$$

Using mid point algorithm point are:

k	p_k	(x_{k+1}, y_{k-1})	$2x_{k+1}$	$2y_{k-1}$
0	– 9	(1, 10)	2	20
1	– 6	(2, 10)	4	20
2	– 1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	– 3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7, 7)	14	14

2.6 POLYGON FILLING ALGORITHM

In many graphics displays, it becomes necessary to distinguish between various regions by filling them with different colour or at least by different shades of gray. There are various methods of filling a closed region with a specified colour or equivalent gray scale. The most general classification appears to be through following algorithms:

1) Scan Line polygon fill algorithm.
2) Seed fill algorithm./Flood fill algorithm which are further classified as:
   (a) Boundary fill algorithm
   (b) Interior fill algorithm

We restrict our discussion to scan line polygon fill algorithm, although we will discuss the others in brief at the end of this section.

Scan Line Polygon Fill Algorithm

In this, the information for a solid body is stored in the frame buffer and using that information all pixels i.e., of both boundary and interior region are identified and, are hence plotted.

Here we are going to do scan conversion of solid areas where the region is bounded by polygonal lines. Pixels are identified for the interior of the polygonal region, and are then filled plotted with the predefined colour.

Let us discuss the algorithm briefly, then we will go into details on the same.

This algorithm checks and alters the attributes (i.e., characteristics and parameters) of the pixels along the current raster scan line. As soon as it crosses over from the outside to the inside of a boundary of the specified polygon it starts resetting the colour (or gray) attribute. In effect filling the region along that scan line. It changes back to the original attribute when it crosses the boundary again. *Figure 14* shows variations of this basic idea.

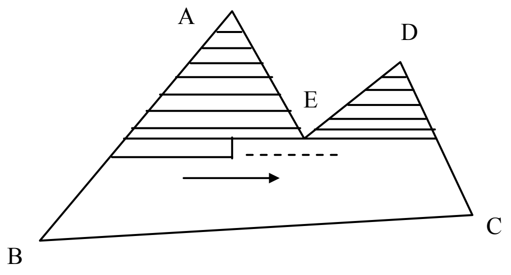


Figure 14: Concept of scan line polygon filling

So as to understand Scan Line Polygon Fill Algorithm in detail consider *Figure 15*:

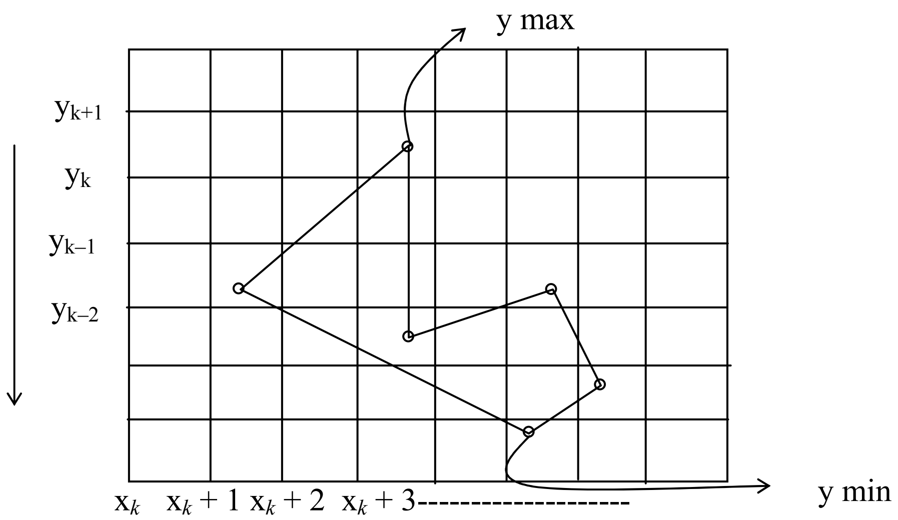


Figure 15: Scan line polygon filling

Here, in *Figure 15,*

- $y_k, y_{k-1}, \ldots\ldots \rightarrow$ are scan lines from top to bottom (value of y at top or scan line at top has maximum value and the scan line at bottom has minimum y value).
- $x_k, x_k + 1 \ldots\ldots \rightarrow$ are consecutive pixels on a scan line *i.e.,* (x_k, y_k), $(x_{k+1}, y_k), \ldots\ldots\ldots\ldots \Rightarrow$ a sea of pixels for scan lines $y_k, y_{k-1}, y_{k-2}, \ldots\ldots, y_i$ and for the chosen scan line we find pixels satisfying the condition as mentioned above.
- For a given scan line, let us assume that it intersects edges of the given polygonal region at the points *P1, P2, ..., Pk.* Sort these points *P1, P2, ..., Pk* in terms of the *X*-coordinates in increasing order.

Now, for a chosen scan line there are three cases:

(a) Scan line passes through the edges in between shown by point a in *Figure 16.*

(b) Scan line passes through the vertex of the polygon whose neighbouring vertices lie completely on one side of the scan line (point b, point c in *Figure 16*).

(c) Scan line passes through some bottom vertex of polygon whose neighbouring vertices lie on both sides of the scan line.

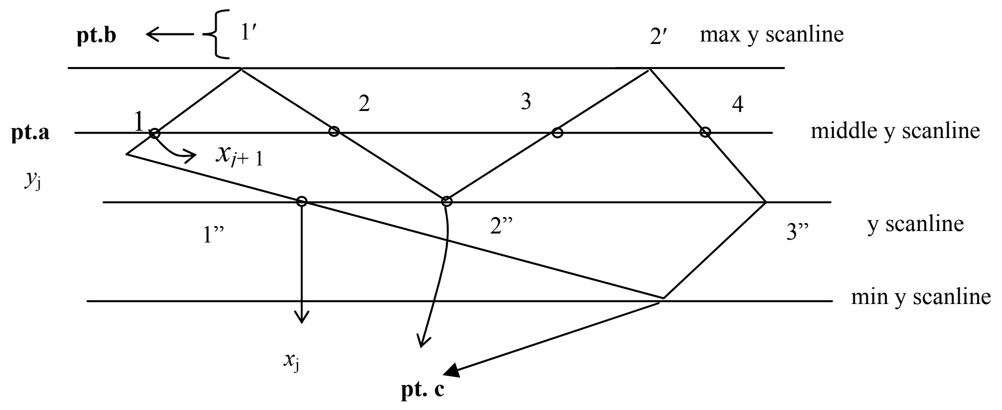


Figure 16: Cases for scan line polygon filling

In case a, *i.e.,* when a scan line intersects an edge at a point but not a vertex point of the edge (as in case of point 1, 2, 3, 4) then that point of intersection is considered as a single point and it is taken as ON/OFF point depending on the requirement. Therefore, in this case the intersection points are taken as 1234. Colour of the pixels between the intersection points 1 and 2 is replaced with the filling colour but the colour of the pixels between the points 2 and 3 are left unchanged. Again the colour of the pixels between 3 and 4 are changed by the filling colour.

In case b and c, case *c* i.e., when a scan line intersects an edge E1 at a vertex V1 i.e., a vertex point of the edge E1 whose *y* coordinate is greater (or less) than the *y* coordinate values of the other two vertex points V2 and V3 where, for example, edge E1 has end vertex points V1, V2 and edge E2 having end vertex points V1, V3. That is the vertices V2 and V3 will lie either completely above V1 or both V2 and V3 will lie completely below the vertex V1. In this case, the point of intersection, i.e., E1, will be counted two times. For example, the vertices 1′ and 2′ are such that their y-coordinate values are grater than the y-coordinate values of their neighbouring vertices and therefore they are counted twice.1′1′ 2′2′ i.e. the colour of the pixels between 1′, 1′ is replaced with the filling colour but the colour of the pixels between 1′ , 2′ is left unchanged. Again the colour of the pixels between 2′, 2′ is changed.

Assume that the scan line passes through a vertex point V1 of the polygonal region having neighbouring vertices V0 and V2., i.e. let E1 and E2 be two edges of the polygon so that V0, V1 and V1, V2 be respectively their end vertices. Suppose we assume that the vertex V1 is such that the y-coordinate for the neighbouring vertex V0 is grater than the y-coordinate for V1 but the y-coordinate for the neighbouring vertex

V2 is less than the y-coordinate for V1. In this case, the intersection vertex point V0 will be taken as a single point and the algorithm will remain the same as above.

The point of intersection is to be considered two times, i.e., 1″ 2″ 2″ 3″. 1″ 2″ ⇒ make pixels ON from 1″ to 2″, 2″ 2″ don't make pixel ON, 2″, 3″⇒ make pixels ON from 2″ to 3″.

Now, the requirements for scanning an image are:

1) Determine the point of intersection of an edge and scan line

2) Sort the points on scan line w.r.t *x* value. Get the respective pixels ON.

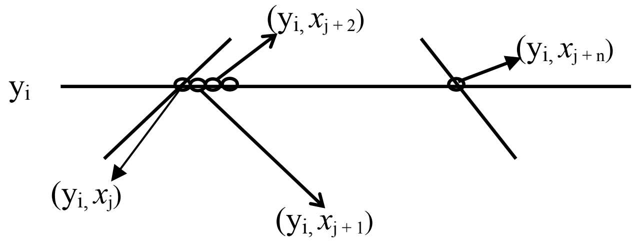


Figure 17: Requirement for image scanning

Sum up: To fill a polygonal image we have to identify all edges and vertices of intersections while moving from scan line y_{max} to y_{min}, for each scan line. Then check for the case and perform the algorithm accordingly.

Recursive way to scan a figure

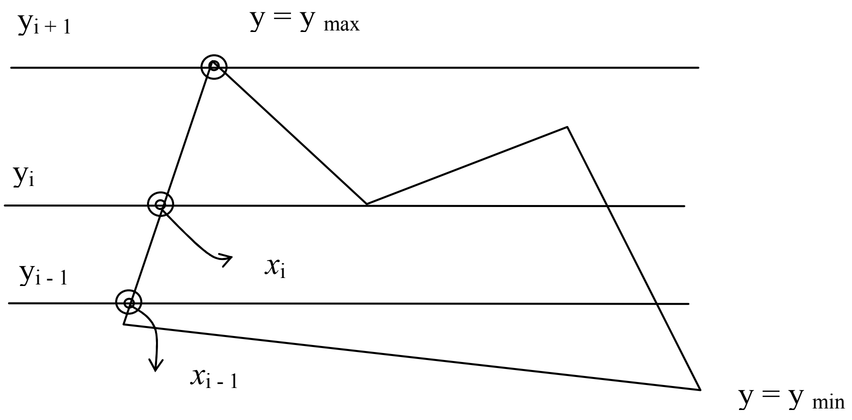


Figure 18: Scan line polygon filling (a recursive approach)

For a scan line $y = y_i$ identify the edges and use the data of the point of intersection $(x_i, y_i.)$. Say x_0 is point on scan line y_0 and x_1 is point on scan line y_1. Now between (x_0, y_0) and (x_1, y_1) we find slope (this all is to find recursive way to scan a figure).

x_0	y_0	x_1	y_1	$1/m$

$$\because \quad m = y_1 - y_0/x_1 - x_0 = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

$$\Rightarrow m\,(x_{i+1} - x_i) = (y_{i+1} - y_i) \qquad \text{-------------(1)}$$

now y_{i+1} is scan line next to y_i i.e. $\boxed{}$ -------------(2)

($\because$ moving from top to bottom we are moving from y_{max} to y_{min})

Using (2) in (1) we get

$$m\,(x_{i+1} - x_i) = (y_{i+1} - y_i) = (y_i + 1 - y_i)$$

$$\Rightarrow \quad \boxed{x_{i+1} = x_i - (1/m)}$$

Using y_{i+1} and x_{i+1} expressions we can find consecutive points. x for respective scan lines y_i . If x_i comes out to be real then round the value to integer values.

Note: Although Flood fill algorithms are not part of this block let me briefly describe Flood Fill Algorithms. The algorithm starts with the coordinates of a specified point (known as the "seed" pixel) inside the polygon to be filled, and proceeds outwards from it, testing the adjacent pixels around it in orderly fashion until the spreading wave reaches the boundary in every direction, as suggested in *Figure 19*.

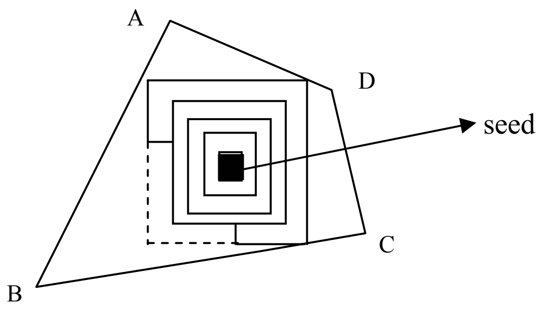


Figure 19: Flood filling method

To economise in storage and memory management, a combination of the two techniques may be used. Starting from an interior seed, first the scan-line through the seed is painted until the left and right boundaries are reached, then the line above and below are similarly covered. The procedure is repeated until the entire region is filled.

Filling may be solid (painted with a single colour), or it may be patterned or tiled (filled with different patterns).

Fill options in standard packages such as MS-Word are grouped generally under the heads: (a) Gradient, variations from light to dark of specified colour in specified directions; (b) Texture; (c) Pattern (i.e., Hatching); and (d) Picture, of the user's choice. Some of these will be discussed later.

☞ Check Your Progress 3

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

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

2) Given a circle radius $r = 5$ determine positions along the circle octants in 1st Quadrant from $x = 0$ to $x = y$?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

3) Distinguish between Scan line polygon fill and Seed fill or Flood fill algorithm?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2.7 SUMMARY

In this unit, we have discussed the basic graphic primitives that is point, line and circle; we have also discussed both theoretical and practical application of different algorithms related to their generation. At the end of the unit, we have emphasised on different seed fill and flood fill type of polygon filling algorithms. The filling algorithms are quite important as they give privilege to quickly fill colours into the graphics created by you.

2.8 SOLUTIONS/ANSWERS

Check Your Progress 1

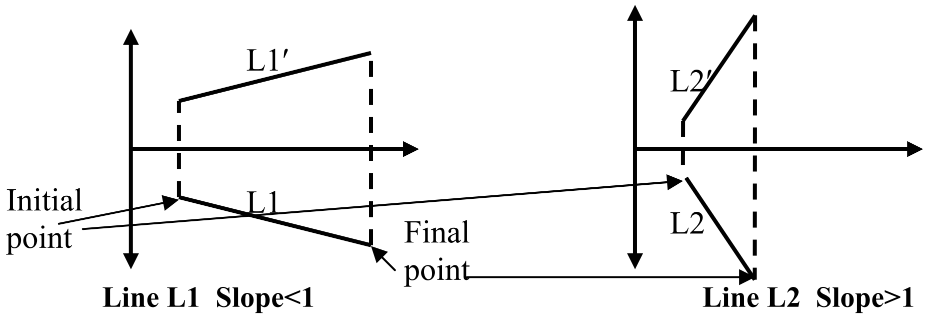
1) Graphic primitives are the fundamental graphic objects which can be united in any number and way to produce a new object or image.

2) Due to low resolution and improper approximation of pixel positions the images are generally distorted and the Zig-Zags (i.e., distortions) produced in the display of straight line or any other graphic primitive is the staircase effect.

3) The approximation involved in the calculation for determination of pixel position involved in the display of lines and other graphic primitives is the actual cause of this effect.

4) In a high resolution system the adjacent pixels are so closely spaced that the approximated line-pixels lie very close to the actual line path and hence the plotted lines appear to be much smoother — almost like straight lines drawn on paper. In a low resolution system, the same approximation technique causes lines to be displayed with a "stairstep apperance" i.e., not smooth.

5)



Line L1 Slope<1 **Line L2 Slope>1**

For the generation of lines with negative slopes:

Slope < 1 : successively increase y and respectively decrease x
Slope > 1 : successively increase x and respectively decrease y

Aliter: This hint you will understand after going through Block 2.

In the shown Figure say L1 and L2 are lines with negative slope with a magnitude of <1 and >1 respectively. Then for the generation of such a line, you may take reflection of the line about the concerned axis and follow the usual algorithm of line generation. After that you may inverse reflection the transformation, and this will provide you the line generation of negative slope.

6) We know that the general equation of the line is given by
$y = mx+c$ where $m =(y_1 – y_0/(x_1 – x_0)$

given $(x_0, y_0) \rightarrow (1, 1)$; $(x_1, y_1) \rightarrow (9, 7)$

$\Rightarrow m = (7-1)/(9-1) =6/8$
$C = y_1 – mx_1 = 7 – (6/8) *9 = 1/4$

So, by equation of line $(y = mx + c)$ we have
$y = (6/8)x+(1/4)$

DDA Algorithm Two case:	
Case 1: $m < 1$	$x_{i+1} = x_i + 1$
	$y_{i+1} = y_i + m$
Case 2: $m > 1$	$x_{i+1} = x_i + (1/m)$
	$y_{i+1} = y_i + 1$

as m < 1 so according to DDA algorithm case 1
$x_{i+1} = x_i + 1$ $\quad y_{i+1} = y_i + m$
given (x_0, y_0) = (1, 1)

1) $x_1 = x_0 + 1 = 2$
$y_1 = y_0 + m = 1+(6/8) = 7/4$

put pixel (x_0, round y, colour)
i.e., put on (2, 2)

Similarly, go on till (9, 7) is arrived at.

Check Your Progress 2

1) Bresenham line generation algorithm is better than DDA because it has better solution to the concept of approximation of pixel position to be enlightened for the display of any graphic primitive, thus, reduces the staircase effect, for more details refer to 2.4.2

2) Here we are using the Bresenham line generation algorithm by digitising the line with end points (15, 5) and (25,13).

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{13-5}{25-15} = \frac{8}{10} = 0.8 \qquad \text{------------------(1)}$$

$\Rightarrow \Delta y = 8$ and $\Delta x = 10$ -------------------(2)

value of initial decision parameter (p_0) = $2\Delta y - \Delta x = 2 * 8 - 10 = 6$ ------------(3)

value of increments for calculating successive decision parameters are:

$2\Delta y = 2 * 8 = 16;$ -------------------(4)

$2\Delta y - 2\Delta x = 2 * 8 - 2 * 10 = -4$ -------------------(5)

plot initial point $(x_0, y_0) = (15, 5)$ in the frame buffer now determine successive pixel positions along line path from decision parameters value (15, 5).

k	**p_k**	**(x_{k+1}, y_{k+1})**	**← [use step (d) of algorithm Δx time]**
0	6	(16, 6)	
1	2	(17, 7)	
2	– 2	(18, 7)	
3	14	(19, 8)	
4	10	(20, 9)	
5	6	(21, 10)	
6	2	(22, 11)	
7	– 2	(23, 11)	
8	14	(24, 12)	
9	10	(25, 13)	

If $p_k > 0$ then increase both X and Y and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$

If $p_k < 0$ then increase X and not Y and $p_{k+1} = p_k + 2\Delta y$

Check Your Progress 3

1) No, there is no need to generate the full circumference of the circle using the algorithm. We may generate it in a quadrant or octant only and then use it to produce the rest of the circumference by taking reflections along the respective axis.

2) Refer to the discussed example 4 on page 16.

3)

Scan Line Polygon	**Flood Fill Algorithms**
• This algorithm checks and alters the attributes (i.e., characteristics and parameters) of the pixels along the current raster scan line. As soon as it crosses from outside to the inside of a boundary of the specified polygon or other region, it starts resetting the colour (or gray) attribute, in effect filling the region along that scan line. It changes back to the original attribute when it crosses the boundary again.	• Flood Fill Algorithms**.** In these, the algorithm starts with the coordinates of a specified point (known as the "seed") inside the polygon to be filled, and proceeds outwards from it, testing the adjacent pixels around it in orderly fashion, until the spreading wave reaches the boundary in every direction.
• Polygon filling is time consuming.	• Polygon filling is quite quick.

UNIT 3 2-D VIEWING AND CLIPPING

Structure **Page No.**

3.1 INTRODUCTION

In the earlier two units of this block, we discussed the basic components of computer graphics, i.e., the software and hardware used for graphic systems along with the graphic primitives used to create graphic images. Now, we are in a position to discuss technicalities related to the display of any graphic primitive or image. Let us begin with, the concept of clipping, for both point and line clipping, followed by the concerned algorithms used to perform line clipping. We shall then examine the concept and algorithm related to polygon clipping, and end with a discussion on window to viewport transformations.

Clipping may be described as the procedure that identifies the portions of a picture lie inside the region, and therefore, should be drawn or, outside the specified region, and hence, not to be drawn. The algorithms that perform the job of clipping are called clipping algorithms there are various types, such as:

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

Further, there are a wide variety of algorithms that are designed to perform certain types of clipping operations, some of them which will be discussed in unit.

Line Clipping Algorithms:
- Cohen Sutherland Line Clippings
- Cyrus-Beck Line Clipping Algorithm

Polygon or Area Clipping Algorithm
- Sutherland-Hodgman Algorithm

There are various other algorithms such as, Liang – Barsky Line clipping, Weiler-Atherton Polygon Clipping, that are quite efficient in performing the task of clipping images. But, we will restrict our discussion to the clipping algorithms mentioned earlier.
Before going into the details of point clipping, let us look at some basic terminologies used in the field of clipping, such as, window and viewport.

Window may be described as the world coordinate area selected for display.

Viewport may be described as the area on a display device on which the window is mapped.

So, it is the window that specifies what is to be shown or displayed whereas viewport specifies where it is to be shown or displayed.

Specifying these two coordinates, i.e., window and viewport coordinates and then the transformation from window to viewport coordinates is very essential from the point of view of clipping.

Note:

- Assumption: That the window and viewport are rectangular. Then only, by specifying the maximum and the minimum coordinates *i.e.*, (Xw_{max}, Yw_{max}) and (Xw_{min}, Yw_{min}) we can describe the size of the overall window or viewport.
- Window and viewport are not restricted to only rectangular shapes they could be of any other shape (Convex or Concave or both).

For better understanding of the clipping concept refer to *Figure 1*:

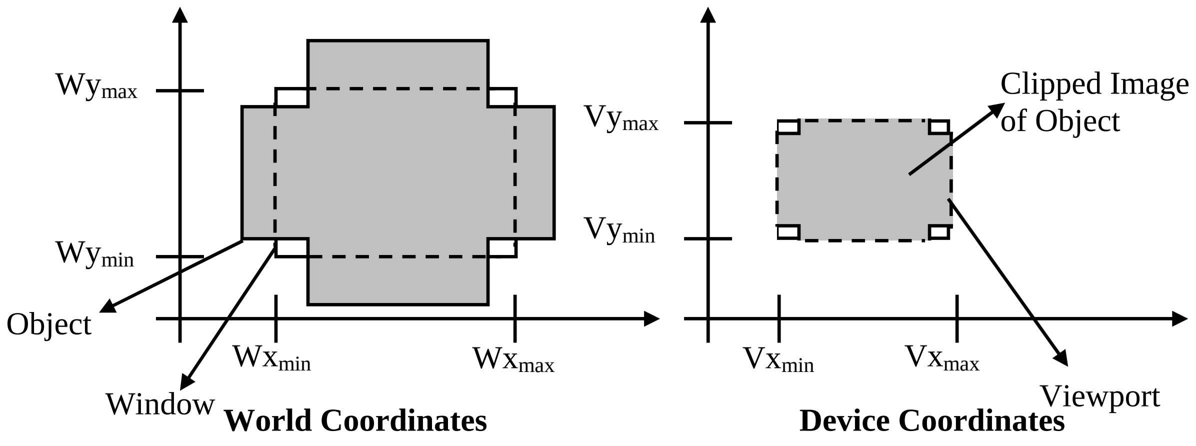


Figure 1: Clipping alongwith Viewing Transformation through rectangular window and viewport

3.2 OBJECTIVES

After going though this unit, you should be able to:

- explain the concept of clipping,
- examine how line clipping is performed,
- understand and implement the algorithms that work behind the concept of line clipping,
- explain how polygon clipping is performed,
- understand and implement the algorithms that work behind the concept of polygon clipping, and
- describe window to viewport transformation.

3.3 POINT CLIPPING

Point clipping is the technique related to proper display of points in the scene, although, this type of clipping is used less frequently in comparison to other types, i.e., line and polygon clipping. But, in some situations, e.g., the scenes which involve particle movements such as explosion, dust etc., it is quite useful. For the sake of simplicity, let us assume that the clip window is rectangular in shape. So, the minimum and maximum coordinate value, i.e., (Xw_{max}, Yw_{max}) and (Xw_{min}, Yw_{min})

are sufficient to specify window size, and any point (X,Y), which can be shown or displayed should satisfy the following inequalities. Otherwise, the point will not be visible. Thus, the point will be clipped or not can be decided on the basis of following inequalities.

Yw_{max}

$$Xw_{min} \leq X \leq Xw_{max}$$
$$Yw_{min} \leq Y \leq Yw_{max}$$

Yw_{min}

Xw_{min} Xw_{max}

Figure 2: Point Clipping

It is to be noted that (Xw_{max}, Yw_{max}) and (Xw_{min}, Yw_{min}) can be either world coordinate window boundary or viewport boundary. Further, if any one of these four inequalities is not satisfied, then the point is clipped (not saved for display).

3.4 LINE CLIPPING

Line is a series of infinite number of points, where no two points have space in between them. So, the above said inequality also holds for every point on the line to be clipped. A variety of line clipping algorithms are available in the world of computer graphics, but we restrict our discussion to the following Line clipping algorithms, name after their respective developers:

1) Cohen Sutherland algorithm,
2) Cyrus-Beck of algorithm

3.4.1 Cohen Sutherland Line Clippings

This algorithm is quite interesting. The clipping problem is simplified by dividing the area surrounding the window region into four segments Up, Down, Left, Right (U,D,L,R) and assignment of number 1 and 0 to respective segments helps in positioning the region surrounding the window. How this positioning of regions is performed can be well understood by considering *Figure 3*.

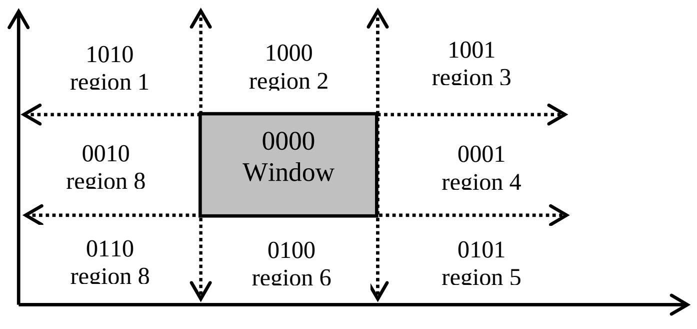


Figure 3: Positioning of regions surrounding the window

In *Figure 3* you might have noticed, that all coding of regions U,D,L,R is done with respect to window region. As window is neither Up nor Down, neither Left nor Right, so, the respective bits UDLR are 0000; now see region 1 of *Figure 3*. The positioning code UDLR is 1010, i.e., the region 1 lying on the position which is upper left side of the window. Thus, region 1 has UDLR code 1010 (Up so U=1, not Down so D=0, Left so L=1, not Right so R=0).

The meaning of the UDLR code to specify the location of region with respect to window is:

1st bit $\Rightarrow$ Up(U) ; 2nd bit $\Rightarrow$ Down(D) ;3rd bit $\Rightarrow$ Left(L) ; 4th bit $\Rightarrow$ Right(R), Now, to perform Line clipping for various line segment which may reside inside the window region fully or partially, or may not even lie in the widow region; we use the tool of logical ANDing between the UDLR codes of the points lying on the line.

Logical ANDing (^) operation between respective bits implies	=>	1 ^ 1 = 1; 1 ^ 0 = 0; 0 ^ 1 = 0; 0 ^ 0 = 0

Note:

- UDLR code of window is 0000 always and w.r.t. this will create bit codes of other regions.
- A line segment is visible if both the UDLR codes of the end points of the line segment equal to 0000 i.e. UDLR code of window region. If the resulting code is not 0000 then, that line segment or section of line segment may or may not be visible.

Now, let us study how this clipping algorithm works. For the sake of simplicity we will tackle all the cases with the help of example lines l_1 to l_5 shown in *Figure 4*. Each line segment represents a case.

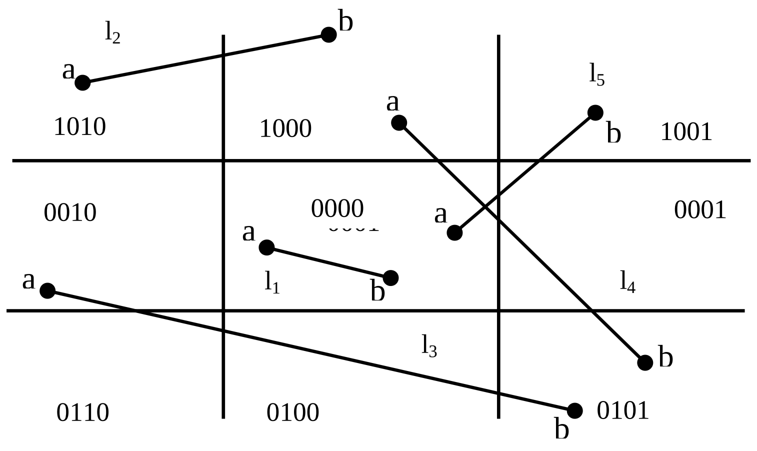


Figure 4: Various cases of Cohen Sutherland Line Clippings

Note, that in *Figure 4*, line l_1 is completely visible, l_2 and l_3 are completely invisible; $l_4$and l_5 are partially visible. We will discuss these out comings as three separate cases.

Case 1: $l_1 \rightarrow$ Completely visible, i.e., Trival acceptance
($\therefore$ both points lie inside the window)

Case 2: l_2 and $l_3 \rightarrow$ Invisible , i.e., Trival acceptance rejection

Case 3: l_4 and $l_5 \rightarrow$ partially visible ($\therefore$ partially inside the window)
Now, let us examine these three cases with the help of this algorithm:

Case 1: (Trivial acceptance case) *if the UDLR bit codes of the end points P,Q of a given line is 0000 then line is completely visible.* Here this is the case as the end points a and b of line l_1 are: a (0000), b (0000). If this trival acceptance test is failed then, the line segment PQ is passed onto Case 2.

Case 2: (Trivial Rejection Case) *if the logical intersection (AND) of the bit codes of the end points P, Q of the line segment is $\neq$ 0000 then line segment is not visible or is rejected.*

Note that, in *Figure 4,* line 2 is completely on the top of the window but line 3 is neither on top nor at the in bottom plus, either on the LHS nor on the RHS of the window. We use the standard formula of logical ANDing to test the non visibility of the line segment.

So, to test the visibility of line 2 and 3 we need to calculate the logical intersection of end points for line 2 and line 3.

line l2: bit code of end points are 1010 and 1000
logical intersection of end points = (1010) ^ (1000) = 1000
as logical intersection $\neq$ 0000. So line 2 will be invisible.

line l3: end points have bit codes 0010 and 0101 now logical intersection = 0000, i.e., 0010 ^ 0101 = 0000 from the *Figure 4*, the line is invisible. Similarly in line 4 one end point is on top and the other on the bottom so, logical intersection is 0000 but then it is partially visible, same is true with line 5. These are special cases and we will discuss them in case 3.

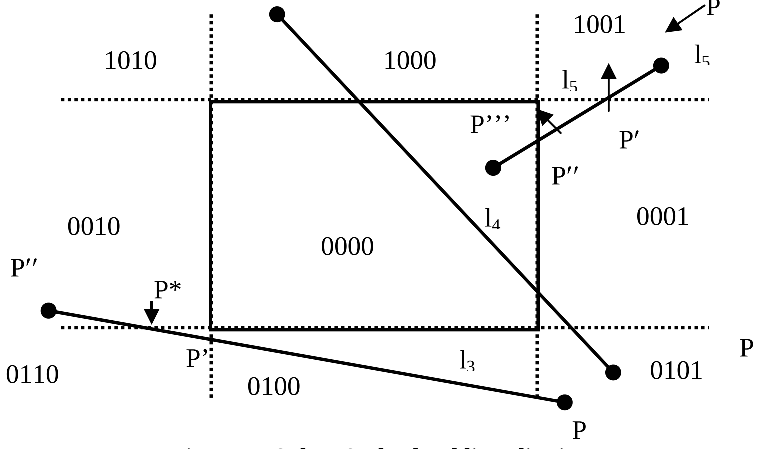


Figure 5: Cohen Sutherland line clipping

Case 3: Suppose for the line segment PQ, both the trivial acceptance and rejection tests failed (i.e., Case 1 and Case 2 conditions do not hold, this is the case for l_3, l_4 and l_5 line segments) shown above in *Figure 5*. For such non-trivial Cases the algorithm is processed, as follows.

Since, both the bitcodes for the end points P, Q of the line segment cannot be equal to 0000. Let us assume that the starting point of the line segment is P whose bit code is not equal to 0000. For example, for the line segment l_5 we choose P to be the bitcodes 1001. Now, scan the bitcode of P from the first bit to the fourth bit and find the position of the bit at which the bit value 1 appears at the first time. For the line segment l_5 it appears at the very first position. If the bit value 1 occurs at the first position then proceed to intersect the line segment with the UP edge of the window and assign the first bit value to its point of intersection as 0. Similarly, if the bit value 1 occurs at the second position while scanning the bit values at the first time then intersect the line segment PQ with the Down edge of the window and so on. This point of intersection may be labeled as P'. Clearly the line segment PP' is outside the window and therefore rejected and the new line segment considered for dipping will be P'Q. The coordinates of P' and its remaining new bit values are computed. Now, by taking P as P', again we have the new line segment PQ which will again be referred to Case 1 for clipping.

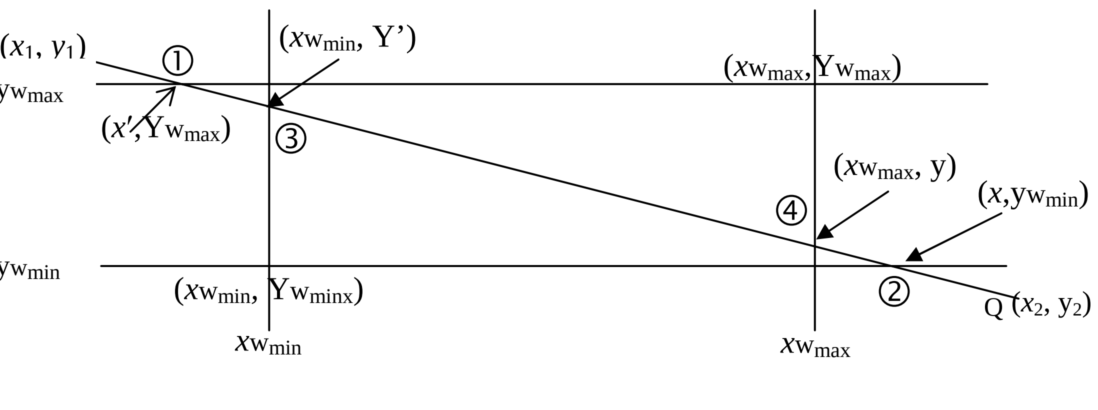


Figure 6: Line Clipping - Geometrically

Geometrical study of the above type of clipping (it helps to find point of intersection of line PQ with any edge).

Let (x_1, y_1) and (x_2, y_2) be the coordinates of P and Q respectively.

1) top case/above
   if $y_1 > yw_{max}$ then 1st bit of bit code = 1 (signifying above) else bit code = 0

2) Bottom case/below case
   if $y_1 < yw_{min}$ then 2nd bit = 1 (*i.e.* below) else bit = 0

3) Left case: if $x_1 < xw_{min}$ then 3rd bit = 1 (*i.e.* left) else 0

4) Right case: if $x_1 > xw_{max}$ then 4th bit = 1 (*i.e.* right) else 0

Similarly, the bit codes of the point Q will also be assigned.

1) **Top/above case:**

   equation of top edge is: $y = yw_{max}$. The equation of line PQ is
   $$y - y_1 = m (x - x_1)$$
   where,
   $$m = (y_2 - y_1)/ (x_2 - x_1)$$
   The coordinates of the point of intersection will be (x, yw_{max}) $\therefore$ equation of line between point P and intersection point is

   $(yw_{max} - y_1) = m (x - x_1)$
   rearrange we get
   $$x = x_1 + \frac{1}{m} (yw_{max} - y_1) \qquad \text{-------------------- (A)}$$

   Hence, we get coordinates (x, yw_{max}) i.e., coordinates of the intersection.

2) **Bottom/below edge** start with $y = yw_{min}$ and proceed as for above case.

   $\therefore$ equation of line between intersection point (x', yw_{min}) and point Q *i.e.* (x_2, y_2) is
   $$(yw_{min} - y_2) = m (x' - x_2)$$

   rearranging that we get,
   $$\boxed{x' = x_2 + \frac{1}{m} (yw_{min} - y_2)} \qquad \text{-------------------- (B)}$$

   The coordinates of the point of intersection of PQ with the bottom edge will be
   $$(x_2 + \frac{1}{m}(yw_{min} - y_2), yw_{min})$$

3) **Left edge:** the equation of left edge is $x = xw_{min}$.

   Now, the point of intersection is $(xwmin, y)$.
   Using 2 point from the equation of the line we get
   $(y - y_1) = m (xw_{min} - x_1)$

   Rearranging that, we get, $y = y_1 + m (xw_{min} - x_1)$. -------------------- (C)

Hence, we get value of xw_{min} and y both *i.e.* coordinates of intersection point is given by $(xw_{\min}, y_1 + m(xw_{\min} - x_1))$.

4) **Right edge:** proceed as in left edge case but start with x-xw_{max}.

Now point of intersection is (xw_{max}, y′).

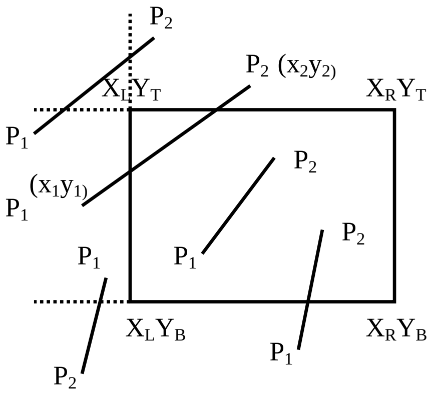
Using 2 point form, the equation of the line is

$(y' - y_2) = m\,(xw_{max} - x_2)$

$$\boxed{y' = y_2 + m\,(xw_{max} - x_2)} \quad \text{--------------------} \quad (D)$$

The coordinates of the intersection of PQ with the right edge will be $(xw_{\max}, y_2 + m(xw_{\max} - x_2))$.

Steps of the Cohen Sutherland Line Clipping Algorithm are



L→Left ;
R→ Right
T→Top ;
B→Bottom

Figure 7: Steps for Cohen Sutherland Line Clipping

STEP 1:

```
Input: $x_L, x_R, y_T, y_B, P_1(x_1, y_1), P_2(x_2, y_2)$
Initialise $i = 1$
While $i <= 2$
if $x_i < x_L$ then bit 1 of code –$P_i$ = 1 else 0
if $x_i > x_R$ then bit 2 of code –$P_i$ =1 else 0      : The endpoint codes of the line
                                                         are set
if $y_i < y_B$ then bit 3 of code –$P_i$ = 1 else 0
if $y_i > y_T$ then bit 4 of code –$P_i$ = 1 else 0
i = i +1
end while
i = 1
```

STEP 2:

```
Initialise j = 1
While j <= 2
if $x_j < x_L$ then $C_{j\,left} = 1$ else $C_{j\,left} = 0$
if $x_j > x_R$ then $C_{j\,right} = 1$ else $C_{j\,right} = 0$    :Set flags according to the position
                                                                    of the line endpoints w.r.t.
                                                        window
if $y_j < y_B$ then $C_{j\,bottom} = 1$ else $C_{j\,bottom} = 0$   edges
```

```
    if $y_j > y_T$ then $C_{j\,top} = 1$ else $C_{j\,top} = 0$
    end while
```

STEP 3: If codes of $P_1$and P_2 are both equal to zero then draw P_1P_2 (totally visible)

STEP 4: If logical intersection or AND operation of code –P_1 and code –P_2 is not equal to zero then ignore P_1P_2 (totally invisible)

STEP 5: If code –P_1= 0 then swap P_1 and P_2 along with their flags and set i = 1

STEP 6: If code –P_1 < > 0 then

```
    for i = 1,
    {if $C_{1\,left}$ = 1 then
    find intersection $(x_L, y'_L)$ with left edge vide eqn. (C)
    assign code to $(x_L, y'_L)$
    $P_1$ = $(x_L, y'_L)$
    end if
    i = i + 1;
    go to 3
    }

 for i = 2,
    {if $C_{1\,right}$ = 1 then
     find intersection $(x_R, y'_R)$ with right edge vide eqn. (D)
     assign code to $(x_R, y'_R)$
     $P_1$ = $(x_R, y'_R)$
    end if
    i = i + 1
    go to 3
    }

for i = 3
    {if $C_{1\,bottom}$ = 1 then
     find intersection $(x'_B, y_B)$ with bottom edge vide eqn. (B)
     assign code to $(x'_B, y_B)$
     $P_1$ = $(x'_B, y_B)$
    end if
    i = i + 1
    go to 3
    }

for i = 4,
    {if $C_{1\,top}$ = 1 then
     find intersection $(x'_T, y_T)$ vide eqn. (A) with top edge
     assign code to $(x'_T, y_T)$
     $P_1$ = $(x'_T, y_T)$
    end if
    i = i + 1
    go to 3
    }
end
```

Example: A Clipping window ABCD is located as follows:

A(100, 10), B(160, 10, C(160, 40), D(100, 40). Using Sutherland-Cohen clipping algorithm find the visible portion of the line segments EF, GH and P_1P_2. E(50,0), F(70,80), G(120, 20), H(140, 80), P_1(120, 5), P_2(180, 30).

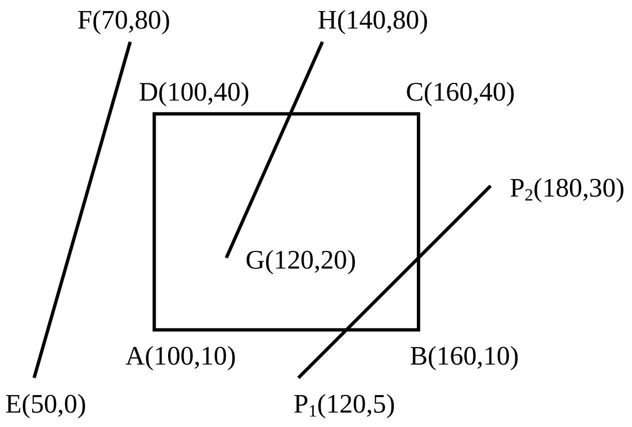


Figure 8: Example of Cohen Sutherland Line Clipping

At first considering the line P_1P_2

INPUT: P_1(120, 5), P_2(180, 30)
$x_L = 100$, $x_R = 160$, $y_B = 10$, $y_T = 40$

$x_1 > x_L$ then bit 1 of code –P_1 = 0 $C_{1\text{ left}} = 0$
$x_1 < x_R$ then bit 2 of code –P_1 = 0 $C_{1\text{ right}} = 0$
$y_1 < y_B$ then bit 3 of code –P_1 = 1 $C_{1\text{ bottom}} = 1$
$y_1 < y_T$ then bit 4 of code –P_1 = 0 $C_{1\text{ top}} = 0$

code –P_1 = 0100,

$x_2 > x_L$ then bit 1 of code –P_1 = 0 $C_{2\text{ left}} = 0$
$x_2 > x_R$ then bit 2 of code –P_1 = 1 $C_{2\text{ right}} = 1$
$y_2 > y_B$ then bit 3 of code –P_1 = 0 $C_{2\text{ bottom}} = 0$
$y_2 < y_T$ then bit 4 of code –P_1 = 0 $C_{2\text{ top}} = 0$

code –P_2 = 0010.

```
Both code –$P_1$ <> 0 and code –$P_2$ <> 0
                then $P_1P_2$ not totally visible

code –$P_1$ AND code –$P_2$ = 0000
          hence (code –$P_1$ AND code –$P_2$ = 0)
                then line is not totally invisible.

As code –P <> 0

for  i = 1
     {
          $C_{1\text{ left}}$ (= 0) <> 1 then nothing is done.
          i = i + 1 = 2
          }
          code –$P_1$ <> 0 and code –$P_2$ <> 0
                    then $P_1P_2$ not totally visible.

          code –$P_1$ AND code –$P_2$ = 0000
                    hence (code –$P_1$ AND code –$P_2$ = 0)
```

```
                              then line is not totally invisible.

for   i = 2
      {
            $C_{1\ right}$ (= 0) <> 1 then nothing is to be done.
            i = i + 1 = 2 + 1 = 3
            }
            code –$P_1$ <> 0 and code –$P_2$ <> 0
                  then $P_1P_2$ not totally visible.

            code –$P_1$ AND code –$P_2$ = 0000
                  Hence, (code –$P_1$ AND code –$P_2$ = 0)
                        then the line is not totally invisible.

for   i = 3
   {
            $C_{1\ bottom}$ = 1 then find intersection of $P_1P_2$ with bottom edge
                  $y_B$ = 10
                  $x_B$ = (180-120)(10-5)/(30-5) + 120
                    =132
            then $P_1$ = (132,10)

            $x_1 > x_L$   then bit 1 of code –$P_1$ = 0   $C_{1\ left}$ = 0
            $x_1 < x_R$   then bit 2 of code –$P_1$ = 0   $C_{1\ right}$ = 0
            $y_1 = y_B$   then bit 3 of code –$P_1$ = 0   $C_{1\ bottom}$ = 0
            $y_1 < y_T$   then bit 4 of code –$P_1$ = 0   $C_{1\ top}$ = 0

            code –$P_1$ = 0000
            i = i + 1 = 3 + 1 = 4
            }

            code –$P_1$ <> 0 but code –$P_2$ <> 0
                  then $P_1P_2$ not totally visible.

            code –$P_1$ AND code –$P_2$ = 0000
                  Hence, (code –$P_1$ AND code –$P_2$ = 0)
                        then line is not totally invisible.
As code –$P_1$ = 0

Swap $P_1$ and $P_2$ along with the respective flags

            $P_1$ = (180, 30)
            $P_2$ = (132, 10)
            code –$P_1$ = 0010
            code –$P_2$ = 0000
            $C_{1\ left}$ = 0                 $C_{2\ left}$ = 0
            $C_{1\ right}$ = 1                $C_{2\ right}$ = 0
            $C_{1\ bottom}$ = 0               $C_{2\ bottom}$ = 0
            $C_{1\ top}$ = 0                  $C_{2\ top}$ = 0
Reset i = 1
for i = 1
      {
            $C_{1\ left}$ (= 0) <> 1 then nothing is to be done.
            i = i + 1 = 1 + 1 = 2
      }
            code –$P_1$ <> 0, and code –$P_2$ <> 0
                  then $P_1P_2$ is not totally visible.

            code –$P_1$ AND code –$P_2$ = 0000
```

Hence, (code –P_1 AND code –P_2 = 0)
then line is not totally invisible.

```
for i = 2
    {
        C_{1 right} = 1 then find intersection of P_1P_2 with right edge
            x_R = 160
            y_R = (30 – 5)(160 – 120)/(180 – 120) + 5
                = 21.667
                = 22
        then P_1 = (160, 22)
        $x_1 > x_L$   then bit 1 of code –P_1 = 0   C_{1 left} = 0
        $x_1 = x_R$   then bit 2 of code –P_1 = 0   C_{1 right} = 0
        $y_1 > y_B$   then bit 3 of code –P_1 = 0   C_{1 bottom} = 0
        $y_1 < y_T$   then bit 4 of code –P_1 = 0   C_{1 top} = 0
        code –P_1 = 0000, i = i + 1 = 2 + 1 = 3
}
```

As both code –P_1 = 0 and code –P_2 = 0 then the line segment P_1P_2 is totally visible.

So, the visible portion of input line P_1P_2 is $P'_1P'_2$ where, P_1 = (160, 22) and P_2 = (132, 10).

Considering the line EF

1) The endpoint codes are assigned
   code – E $\rightarrow$ 0101
   code – F $\rightarrow$ 1001
2) Flags are assigned for the two endpoints
   E_{left} = 1 (as x coordinate of E is less than x_L)
   E_{right} = 0, E_{top} = 0 E_{bottom} = 1

   Similarly,
   F_{left} = 1, F_{right} = 0, F_{top} = 1 F_{bottom} = 0

3) Since codes of E and F are both not equal to zero the line is not totally visible.

4) Logical intersection of codes of E and F is not equal to zero. So, we may ignore EF line and declare it as totally invisible.

Considering the line GH

1) The endpoint codes are assigned
   code – G $\rightarrow$ 0000
   code – H $\rightarrow$ 1000

2) Flags are assigned for the two endpoints
   G_{left} = 0, G_{right} = 0, G_{top} = 0, G_{bottom} = 0.

   Similarly,
   H_{left} = 0, H_{right} = 0, H_{top} = 1, H_{bottom} = 0.

3) Since, codes of G and H are both not equal to zero so the line is not totally visible.

4) Logical intersection of codes of G and H is equal to zero so we cannot declare it as totally invisible.

5) Since, code – G = 0, Swap G and H along with their flags and set i = 1

Implying $G_{left} = 0$, $G_{right} = 0$, $G_{top} = 1$, $G_{bottom} = 0$.

$H_{left} = 0$, $H_{right} = 0$, $H_{top} = 0$, $H_{bottom} = 0$.
as $G \rightarrow 1000$, $H \rightarrow 0000$

6) Since, code – G <> 0 then

```
    for i = 1, {since $G_{left} = 0$
                   i = i + 1 = 2
                   go to 3
               }
```

The conditions 3 and 4 do not hold and so we cannot declare line GH as totally visible or invisible.

```
    for i = 2, {since $G_{right} = 0$
                   i = i + 1 = 3
                   go to 3
               }
```

The conditions 3 and 4 do not hold and so we cannot declare line GH as totally visible or invisible.

```
    for i = 3, {since $G_{bottom} = 0$
                   i = i + 1 = 4
                   go to 3
               }
```

The conditions 3 and 4 do not hold and so we cannot declare line GH as totally visible or invisible.

for i = 4, {since $G_{top} = 1$
Intersection with top edge, say P(x, y) is found as follows:

Any line passing through the points G, H and a point P(x, y) is given by

$y - 20 = \{(180 - 20) /(140 - 120)\}(x - 120)$
or, $y - 20 = 3x - 360$
or, $y - 30 = -340$

Since, the y coordinate of every point on line CD is 40, so we put y = 40 for the point of intersection P(x, y) of line GH with edge CD.

$40 - 3x = -340$
or, $- 3x = - 380$
or $x = 380/3 = 126.66 \approx 127$

So, the point of intersection is P(127, 40). We assign code to it.

Since, the point lies on edge of the rectangle so the code assigned to it is 0000.

Now, we assign G = (127, 40); i = 4 + 1 = 5. Conditions 3 and 4 are again checked.

Since, codes G and H are both equal to 0, so, the line between H(120, 20) and G(127, 40) is totally visible.

Limitation of Cohen Sutherland line Clipping Algorithm

The algorithm is only applicable to rectangular windows and not to any other convex shaped window.

So, a new line-clipping algorithm was developed by Cyrus, and Beck to overcome this limitation. This Cyrus Beck line-clipping Algorithm is capable of clip lining segments irrespective of the shape of the convex window.

You might wonder as to what a convex window? In general, on the basis of the shapes the windows are classified into two categories:

i) **Convex shaped windows:** Windows of a shape such that if we take any two points inside the window then the line joining them will never cross the window boundary, such shaped windows are referred as convex shaped windows.

ii) **Concave or non Convex shaped windows:** Windows of a shape such that if we can choose a pair of points inside the window such that the line joining them may cross the window boundary or some section of the line segment may lie outside the window. Such shaped windows are referred to as non-convex or concave shaped windows.

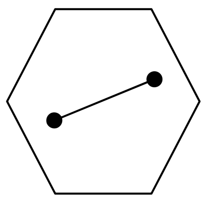


Figure (a)
Convex Polygonal Window

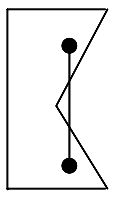


Figure (b)
Non-convex Polygonal window

Figure 9: Types of Windows

3.4.2 Cyrus-Beck Algorithm

Cyrus Beck Line clipping algorithm is infact, a parametric line-clipping algorithm. The term parametric implies that we need to find the value of the parameter t in the parametric representation of the line segment for the point at which the segment intersects the clipping edge. For better understanding, consider the *Figure 9(a),* where P Q is a line segment, which is intersecting at the two edges of the convex window.

Note: The algorithm is applicable to the “convex polygonal window”.

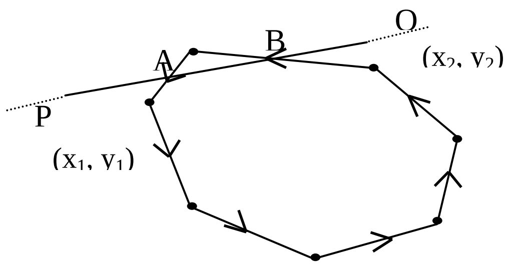


Figure 9(a): Interaction of line PQ and Window

Now, just recall the parametric equation of line segment PQ, which we have studied in the course CS-60.

It is simply $P + t(Q - P) \quad 0 \le t \le 1$

Where, $t \rightarrow$ linear parameter continuously changes value.

$\therefore P + t(Q - P) \Rightarrow (x_1, y_1) + t(x_2 - x_1, y_2 - y_1) = (x, y)$ be any point on PQ. ------ (1)

For this equation (1) we have following cases:

1) When $t = 0$ we obtain the point P.
2) When $t = 1$ we get the point Q.
3) When t varies such that $0 \le t \le 1$ then line between point P and Q is traced.

For t = ½ we get the mid-point of PQ.

4) When t < 0 line on LHS of P is traced.
5) When t > 1 line on RHS of Q is traced.

So, the variation in parameter t is actually generating line in point wise manner .The range of the parameter values will identify the portion to be clipped by any convex polygonal region having n-vertices or lattice points to be specified by the user. One such clipping situation is shown in *Figure 9(a).*

Remark:

- **How to specify the window region:** a convex polygonal region having n-vertices {P_0, P_1, P_2…, P_{n-1}, P_n, P_0} or lattice points to be specified by the user encloses the convex window region. To be specific about the window we may say that each edge between two points contributes to the boundary of the window under the situation that (when we traverse each edge in anticlockwise manner), the window region lies on left hand side (LHS) of edge. This orientation is preserved and while giving input, the user takes care of the orientation to specify the window region of any arbitrary convex shape.

- **Potentially entering and leaving points (P_E and P_L)**
  The point of intersection of the line and window may be classified either as Potentially entering or leaving. Before going into other details, let us describe the actual meaning of Potentially entering and leaving points (P_E and P_L). P_E and P_L are dependent on the edge orientation, *i.e.* direction. Let us understand **how to find P_E and P_L** (we know as we move anticlockwise across the window boundary then region of LHS encloses the window).

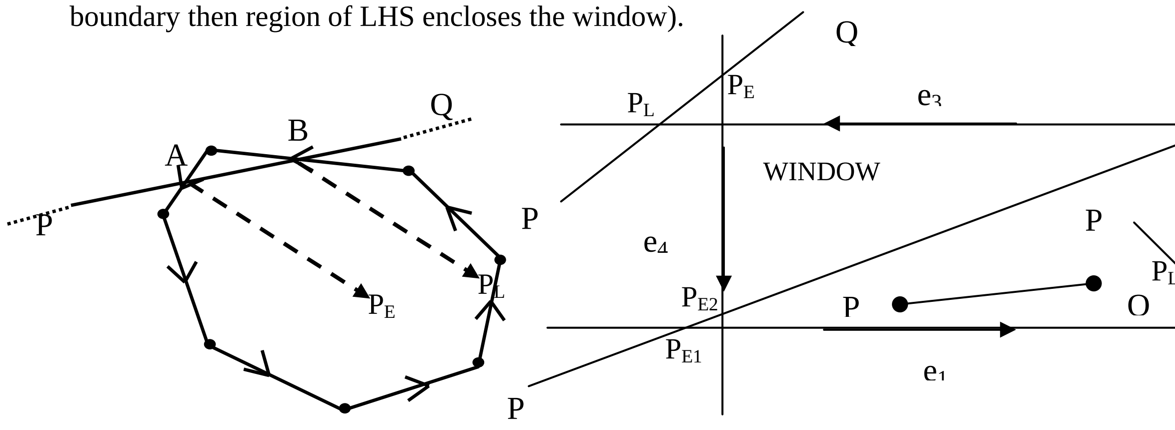


Figure 10(a): Potentially Entering P_E and Potentially Leaving P_L points

Say, we are moving from point P to point Q, and we know that while traversing the windows edges in anticlockwise manner the region lying on the left hand side is referred to as the window region. So, from the *Figure 10 (a)*, you may notice that at point P_E we are moving from P to Q we are entering the window region , thus this point of intersection should be referred to as the Potentially entering point(P_E). Now, refer to other intersection shown in the *Figure 10 (a)*, here, also the movement of points on the line are determined by varying value of parameter t is from point P to Q and w.r.t ., the orientation of window boundary, we are exiting the window region. So, this point of intersection is known as potentially leaving point (P_L).

Potentially entering point (P_E) while moving towards the window, the point of intersection between line PQ and the window edges faced are known as **P_E**.

Potentially leaving point (P_L) These are the point of intersection encountered while we move away from window.

OR

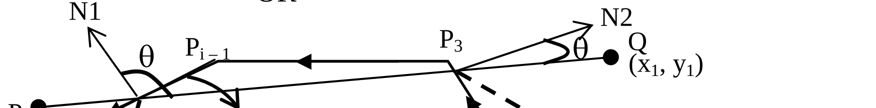


Figure 10 (b): Potentially Entering P_E and Potentially Leaving P_L points

Other way round you may detect the point of intersection as Potentially leaving point (P_L) or Potentially entering point (P_E) by studying the angle between the line to be clipped from the outward normal to the intersecting edge, at the point of intersection. From *Figure 10 (b)*, it is the angle between PQ and N1 or N2. You may notice that, while moving from P to Q the angle between line PQ and N1 is obtuse whereas the angle between PQ and N2 is acute; so, you may say, if the angle between line PQ and N1 is obtuse the point of intersection is potentially entering (P_E) whereas, if the angle between PQ and N2 is acute, then the point of intersection is potentially leaving (P_L).

OR

P_E => **$N_i \cdot PQ < 0$** **;** **(angle θ greater than 90° or Obtuse)**
$N_i \cdot (Q-P) < 0$

P_L => **$N_i \cdot PQ > 0$** **;** **(angle θ is less than 90° or Acute)**
$N_i \cdot (Q-P) > 0$

- **Where N_i is the outward normal to the ith edge.**

Note: when $(\overline{Q} - \overline{P}) \cdot \overline{N_i} = 0$ then it implies that:

1) $\overline{Q} - \overline{P} = 0$	2) $\overline{N_i} = 0$	3) $\theta = 90°$
$\downarrow$	$\downarrow$	$\downarrow$
not possible $\because$ if $\overline{Q} - \overline{P} = 0$ then $\overline{PQ}$ is a point and not a line	not possible	possible i.e. $(\overline{Q} - \overline{P}) \perp \overline{N_i}$

$\Downarrow$

line segment PQ is || to ith edge then only $\overline{N_i}$ will be $\perp$ to both PQ and ith edge.

- **How to find the normal :**

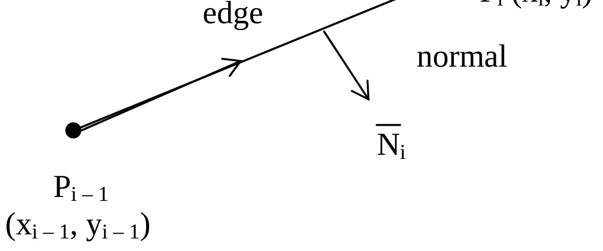


Figure 11 : Normal Determination

$$\overrightarrow{P_{i-1}P_i} = (x_1 - x_{i-1}, y_i - y_{i-1}) = (s_1, s_2)$$

if $\overrightarrow{N_i} = (n_{1i}, n_{2i})$, then

$$\overrightarrow{N_i} \cdot \overrightarrow{P_{i-1}P_i} = 0 \Rightarrow (s_1, s_2).(n_{1i}, n_{2i}) = 0$$

$$\Rightarrow s_1 n_{1i} + s_2 n_{2i} = 0$$
$$\Rightarrow s_1 n_{1i} = - s_2 n_{2i} = 0$$
$$\Rightarrow n_{1i} = \frac{-s_2}{s_1} n_{2i}$$

If $n_{2i} = 1$ then, $n_{1i} = \frac{s_2}{s_1}$

Therefore, $\overrightarrow{N_i} = (n_{1i}, n_{2i}) = (\frac{-s_2}{s_1}, 1) \longrightarrow$ NORMAL

if $\left(\frac{-s_2}{s_1}, 1\right)$ is outward normal then $- \left(\frac{-s_2}{s_1}, 1\right)$ i.e. $\left(\frac{s_2}{s_1}, -1\right)$ is inward normal.

Now, we have learned some basic things which will be required in the working of the algorithm, so, we can start using the concepts learned so far to clip the line segment passing through the window.

We know that the parametric equation of line PQ is

$$P + t\,(Q - P) \;;\quad 0 \le t \le 1$$

Now, to find the point of intersection of line PQ and ith edge we need to find the appropriate value of parameter t, for that we firstly find normal to the ith edge. Say $\overrightarrow{Ni}$ is the normal to ith edge (P_{i-1} to P_i), then to find the appropriate parameter t, we should determine the dot product of the vector from P_{i-1} to the point of intersection and the normal Ni. Let us do this exercise.

The vector joining edge point P_{i-1} and point of intersection of the line PQ with the ith edge for some t will be given by:

$$\{ [\overline{P} + t(\overline{Q} - \overline{P})] - \overline{P}_{i-1} \}$$

As $\overrightarrow{Ni}$ is normal to the ith edge , so, the dot product of the vector joining edge point P_{i-1} and point of intersection of the line PQ with the ith edge for some t and Ni will be given by:

$$\{[\overline{P} + t(\overline{Q} - \overline{P})] - \overline{P}_{i-1}\}.\overrightarrow{N_i} = 0 \qquad \text{---------------(1)}$$

Rearranging (1) we get,

$$t = \frac{-(\overline{P} - \overline{P}_{i-1}).\overline{N_i}}{(\overline{Q} - \overline{P}).\overline{N_i}} \qquad \text{---------------(2)}$$

Using this value of t from (2) in parametric form of line we will get the point of intersection of line PQ and ith edge.

Note: By the above discussion of P_E and P_L we came to know that if $N_i . (Q-P) < 0 \Rightarrow P_E$ point and $N_i . (Q-P) > 0 \Rightarrow P_L$ point. If the value of t calculated using (2) lies in the interval 0 to 1 (in magnitude) then, the point of intersection will be considered otherwise it will not be considered.

Steps to clip a line segment PQ:

- Firstly, find all the points of intersections of the line segment PQ with the edges of the polygonal window and classify them either as P_E and P_L points. Also determine the value of parmeter t , using equation (2) for respective PE's and PL's.

Or

If value of the normals to respective edges are not given then, find the value of normal to every edge of the window, then, determine the value of parmeter t , using equation (2) for respective point of intersection between line segment and window edge then on the basis of the vaue of parameter t mark them as $\mathbf{P_E}$ and $\mathbf{P_L}$ provided the value of t lies from 0 to 1 in magnitude.

- Secondly, out of the various values of t for $\mathbf{P_E}$'s determine the maximum value of t say it be t_{max}. Similarly, out of the various values of t for $\mathbf{P_L}$'s determine the minimum value of t say it be t_{min} . Note that for clipping to be possible $t_{min} > t_{max}$.

- Finally, vary the parameter t from t_{max} to t_{min} and determine the clipped line as outcome.

For better understanding of steps consider *Figure 12.*

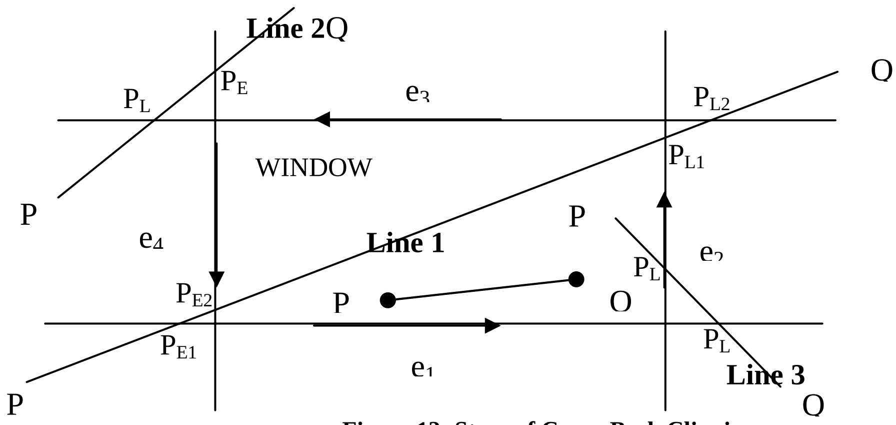


Figure 12: Steps of Cyrus Beck Clipping

In case of Line 1: (PQ)

1) Point 1 is potentially entering (P_{E1}) because as we move along PQ, the LHS of e_1 is window and hence, it seems that we are entering the window.
2) Point 2 is again P_{E2} similarly as point 1.
3) Point 3 is potentially leaving (P_4) $\because$ as we move along PQ, the LHS of e_2 is window and hence, it seems that we are leaving the window.
4) Similarly, point 4 is also PL_2.

Line 2 and 3 (PQ):

Using same logic as for line 1 we find $\overline{P}_L$ and P_E. Now, it is to be noted that for each point of intersection we have some value of t.

say t_1 is value of t for P_{E1}
say t_2 is value of t for P_{E2}
say t_3 is value of t for P_{L1}
say t_4 is value of t for P_{L2}

As the portion visible to us is image what laying inside the window for line 1 the line visible is from P_{E2} to P_{L1} and for these points there is some value of t. With this initialisation let us study the following cases:

Case 1: Say we get a new value of t_E (i.e value of parameter t for any potentially entering (P_E) point) we choose t_{max} as: $t_{max} = \max \{t_{max}, t_E\}$. The initial value of $t_{max} = 0$ is taken.

Case 2: Say we get a new value of t_L (i.e value of parameter t for any potentially leaving(P_L) point) then t_{max} value is updated by $t_{min} = \min\{t_{min}, t_L\}$. The initial value of $t_{min} = 1$ is taken.

Finally when value of t for all edges are found and if $t_{max} < t_{min}$ then line is visible else not. And line is visible from

$P + t_{max}(Q - P)$ to $P + t_{min}(Q - P)$ *i.e.*

$P + t(Q - P)$ such that $t_{max} \le t \le t_{min}$

$t_{max} < t_{min}$ (draw line)

$t_{max} \nless t_{min}$ (reject line)

Example: How does the Cyrus Beck line clipping algorithm, clip a line segment if the window is non convex?

Solution: Consider the *Figure 13*, here, the window is non convex in shape and PQ is a line segment passing through this window .Here too the condition of visibility of the line is $t_{max} < t_{min}$ and the line is visible from $P + t_{max}(Q - P)$ to $P + t_{min}(Q - P)$, if $t_{max} \nless t_{min}$ then reject the line segment. Now, applying this rule to the *Figure 13*, we find that when PQ line segment passes through the non convex window, it cuts the edges of the window at 4 points. $1 \rightarrow P_E$; $2 \rightarrow P_L$; $3 \rightarrow P_E$; $4 \rightarrow P_L$. In this example, using the algorithm we reject the line segment PQ but it is not the correct result.

Condition of visibility is satisfied in region 1-2 and 3-4 only so the line exists there but in region 2-3 the condition is violated so the line does not exists.

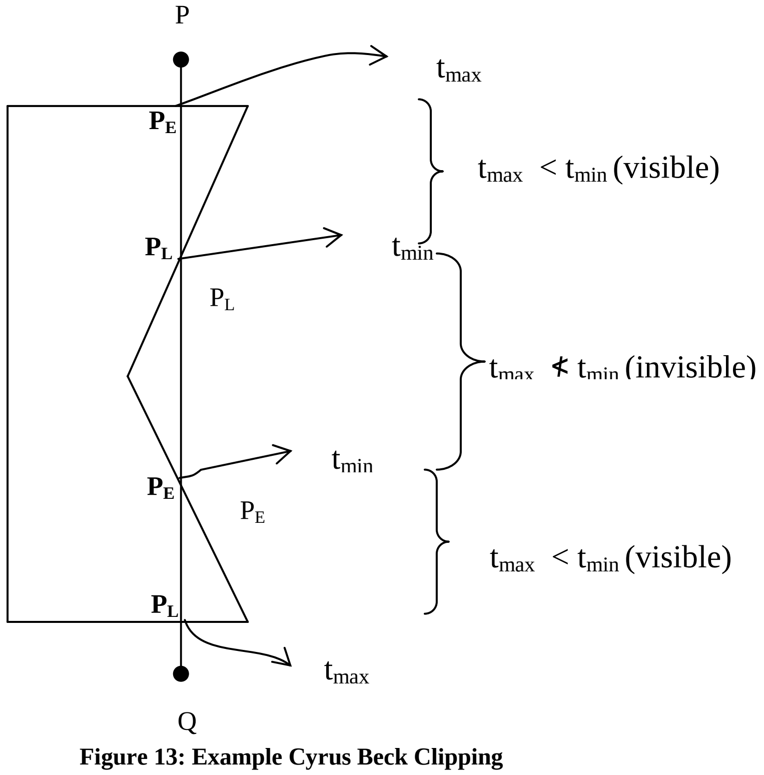

P
t_{max}
P_E
$t_{max} < t_{min}$ (visible)
P_L
t_{min}
P_L
$t_{max} \nless t_{min}$ (invisible)
t_{min}
P_E
P_E
$t_{max} < t_{min}$ (visible)
P_L
t_{max}
Q

Figure 13: Example Cyrus Beck Clipping

☞ Check Your Progress 1

1) Suppose a rectangular window ABCD is defined such that A = (–1, –2) and C(3, 1) using generalised geometrical approach , clip the line segment joining the points P(–20, 0) and Q(20, 30).

……………………………………………………………………………………

……………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………

2) A clipping window is given by $P_0(10,10)$, $P_1(20,10)$, $P_2(25,15)$, $P_3(20,20)$, $P_4(10,15)$, $P_5 = P_0$. Using Cyrus Beck algorithm clip the line A(0,0) and B(30,25).

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

3) How will you compute the outward normal to a clipping in the case of Cyrus-Back algorithm?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

4) Compare Cohen Sutherland Line clipping with Cyrus Beck Line clipping algorithm?

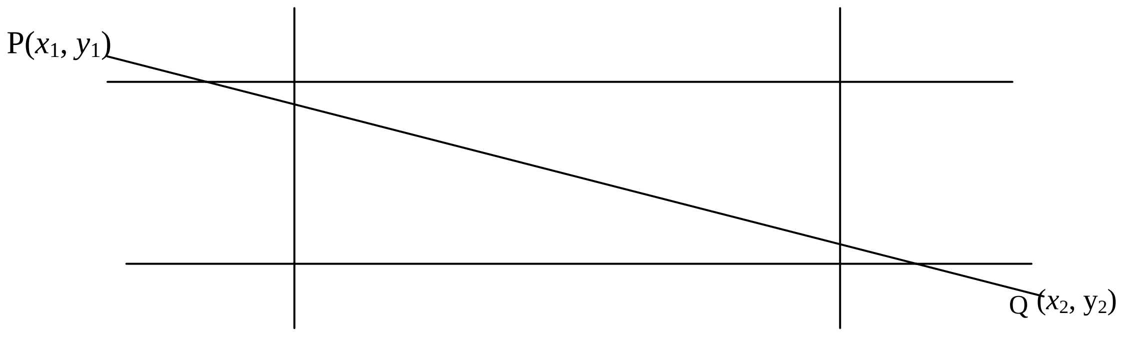
…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

5) Clip the line shown in Figure below using Cohen Sutherland Line clipping algorithm.



…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

6) Which type of clipping windows can't be handled by Cyrus Beck clipping algorithm, how can such cases be handled?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………

3.5 POLYGON CLIPPING

After understanding the concept of line clipping and its algorithms, we can now extend the concept of line clipping to polygon clipping, because polygon is a surface enclosed by several lines. Thus, by considering the polygon as a set of line we can divide the problem to line clipping and hence, the problem of polygon clipping is simplified. But it is to be noted that, clipping each edge separately by using a line clipping algorithm will certainly not produce a truncated polygon as one would expect. Rather, it would produce a set of unconnected line segments as the polygon is exploded. Herein lies the need to use a different clipping algorithm to output truncated but yet bounded regions from a polygon input. Sutherland-Hodgman algorithm is one of the standard methods used for clipping arbitrary shaped polygons with a rectangular clipping window. It uses divide and conquer technique for clipping the polygon.

3.5.1 Sutherland-Hodgman Algorithm

Any polygon of any arbitrary shape can be described with the help of some set of vertices associated with it. When we try to clip the polygon under consideration with any rectangular window, then, we observe that the coordinates of the polygon vertices satisfies one of the four cases listed in the table shown below, and further it is to be noted that this procedure of clipping can be simplified by clipping the polygon edgewise and not the polygon as a whole. This decomposes the bigger problem into a set of subproblems, which can be handled separately as per the cases listed in the table below. Actually this table describes the cases of the Sutherland-Hodgman Polygon Clipping algorithm.

Thus, in order to clip polygon edges against a window edge we move from vertex V_i to the next vertexV_{i+1} and decide the output vertex according to four simple tests or rules or cases listed below:

Table: Cases of the Sutherland-Hodgman Polygon Clipping Algorithm

Case	V_i	V_{i+1}	Output Vertex
A	Inside window)	Inside	V_{i+1}
B	Inside	Outside	V'_i i.e intersection of polygon and window edge
C	Outside	Outside	None
D	Outside	Inside	$V'_I\ ;V_{i+1}$

In words, the 4 possible Tests listed above to clip any polygon states are as mentioned below:

1) If both Input vertices are inside the window boundary then only 2nd vertex is added to output vertex list.
2) If 1st vertex is inside the window boundary and the 2nd vertex is outside then, only the intersection edge with boundary is added to output vertex.
3) If both Input vertices are outside the window boundary then nothing is added to the output list.
4) If the 1st vertex is outside the window and the 2nd vertex is inside window, then both the intersection points of the polygon edge with window boundary and 2nd vertex are added to output vertex list.

So, we can use the rules cited above to clip a polygon correctly. The polygon must be tested against each edge of the clip rectangle; new edges must be added and existing edges must be discarded , retained or divided. Actually this algorithm decomposes the problem of polygon clipping against a clip window into identical subproblems, where a subproblem is to clip all polygon edges (pair of vertices) in succession against a single infinite clip edge. The output is a set of clipped edges or pair of vertices that fall in the visible side with respect to clip edge. These set of clipped edges or output vertices are considered as input for the next sub problem of clipping against the second window edge. Thus, considering the output of the previous subproblem as the input, each of the subproblems are solved sequentially, finally yielding the vertices that fall on or within the window boundary. These vertices connected in order forms, the shape of the clipped polygon.

For better understanding of the application of the rules given above consider the *Figure 14*, where the shaded region shows the clipped polygon.

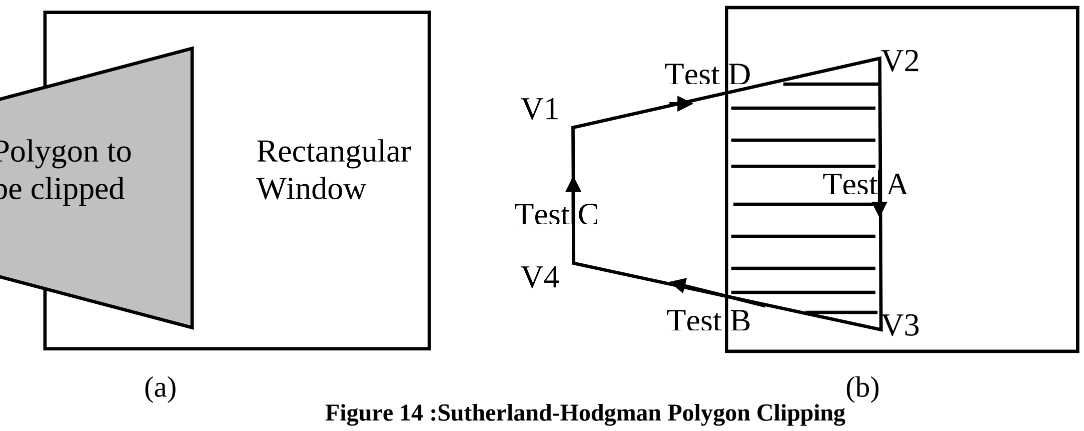


Figure 14 :Sutherland-Hodgman Polygon Clipping

Pseudocode for Sutherland – Hodgman Algorithm

Define variables

inVertexArray is the array of input polygon vertices
outVerteArray is the array of output polygon vertices
Nin is the number of entries in inVertexArray
Nout is the number of entries in outVertexArray
n is the number of edges of the clip polygon
ClipEdge[x] is the xth edge of clip polygon defined by a pair of vertices
s, p are the start and end point respectively of current polygon edge
i is the intersection point with a clip boundary
j is the vertex loop counter

Define Functions

AddNewVertex(newVertex, Nout, outVertexArray)
: Adds newVertex to outVertexArray and then updates Nout

InsideTest(testVertex, clipEdge[x])
: Checks whether the vertex lies inside the clip edge or not; retures TRUE is inside else returns FALSE

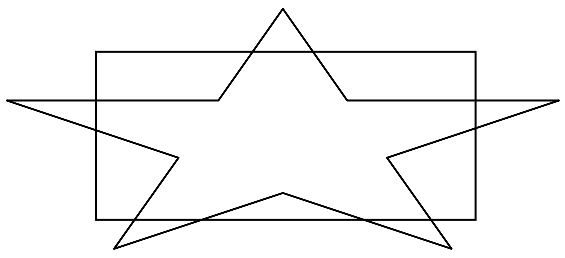
Intersect(first, second, clipEdge[x])
: Clip polygon edge(first, second) against clipEdge[x], outputs the intersection point

```
{                           :  begin main
x = 1
while (x ≤ n)               : Loop through all the n clip edges
{
Nout = 0                    : Flush the outVertexArray
s = inVertexArray[Nin] : Start with the last vertex in inVertexArray
for j = 1 to Nin do         : Loop through Nin number of polygon vertices (edges)
    {
        p = inVertexArrray[j]
        if InsideTest(p, clipEdge[x] = = TRUE then                : Case A
and D
        if InsideTest(s, clipEdge[x] = = TRUE then
        AddNewVertex(p, Nout, outVertexArray)                     :  Case A
          else
        i = Intersect(s, p, clipEdge[x])                      :  Case D
        AddNewVertex(i, Nout, outVertexArray)
        AddNewVertex(p, Nout, outVertexArray)
        end if
        else                 :  i.e. if InsideTest(p, clipEdge[x] = = FALSE
                             (Cases 2 and 3)
        if InsideTest(s, clipEdge[x]) = =TRUE then                : Case B
        {
          Intersect(s, p, clipEdge[x])
          AddNewVertex(i, Nout, outVertexArray)
          end if                              : No action for case C

          s = p                               : Advance to next pair of vertices
          j = j + 1
         end if                               :  end {for}
           }
        x = x + 1                             : Proceed to the next ClipEdge[x +1]
        Nin = Nout
        inVertexArray = outVertexArray            :  The ouput vertex array for
                                          the current clip edge becomes the input
                                          vertex array for the next clip edge
        }                                 : end while
        }                                 : end main
```

☞ **Check Your Progress 2**

1) Using Sutherland-Hodgeman polygon clipping algorithm clip on the polygon given below.



3.6 WINDOWING TRANSFORMATIONS

From section 3.1,we understood the meaning of the term window and viewport which could again be understood as:

Window: A world coordinate area selected for display (i.e. area of picture selected for viewing).

Viewport: An Area or a display device to which a window is mapped.

Note:
- Window defines what is to be viewed and viewpoint defines where it is to be displayed.
- Often window and viewpoints are rectangles in standard position with edges parallel to coordinate axes. Generalised shapes like polygon etc., take long to

process, so we are not considering these cases where window or viewport can have general polygon shape or circular shape.

The mapping of a part of a world coordinate scene to device coordinates is referred to as Viewing Transformation. In general 2D viewing transformations are referred to as window to viewport transformation or windowing transformation.

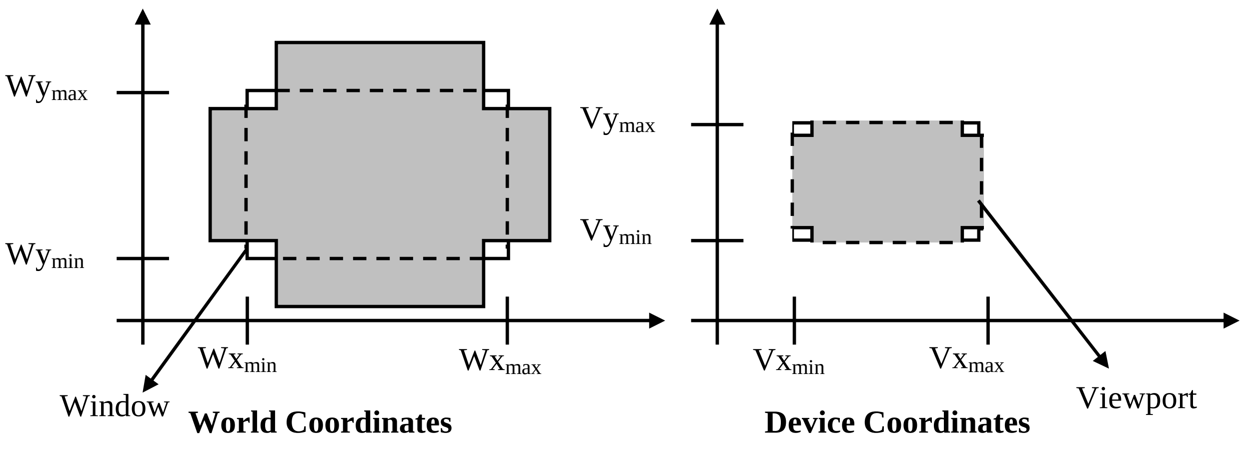


Figure 15: Windowing Transformation

You may notice in *Figure 15*, that all parts of the picture that lie outside the window are clipped and the contents which lie inside the widow are transferred to device coordinates. Secondly, you may also notice that while window selects a part of the scene, viewport displays the selected part at the desired location on the display area. When window is changed we see a different part of the scene at the same portion (viewport) on display. If we change the viewport only, we see the same part of the scene drawn at a different scale or at a different place on the display. By successively increasing or decreasing the size of the window around a part of the scene the viewport remain fixed, we can get the effect of zoom out or zoom in respectively on the displayed part. Mathematically, viewing transformation can be expressed as V=W.N

Where,

- **V** refers Viewing transformation which maps a part of world coordinate scene to device coordinates;
- **W** refers to workstation transformation which maps normalised device coordinates to physical device coordinates;
- **N** refers to Normalisation transformation used to map world coordinates to normalized device coordinates.

Window to Viewpoint Coordinates transformation:

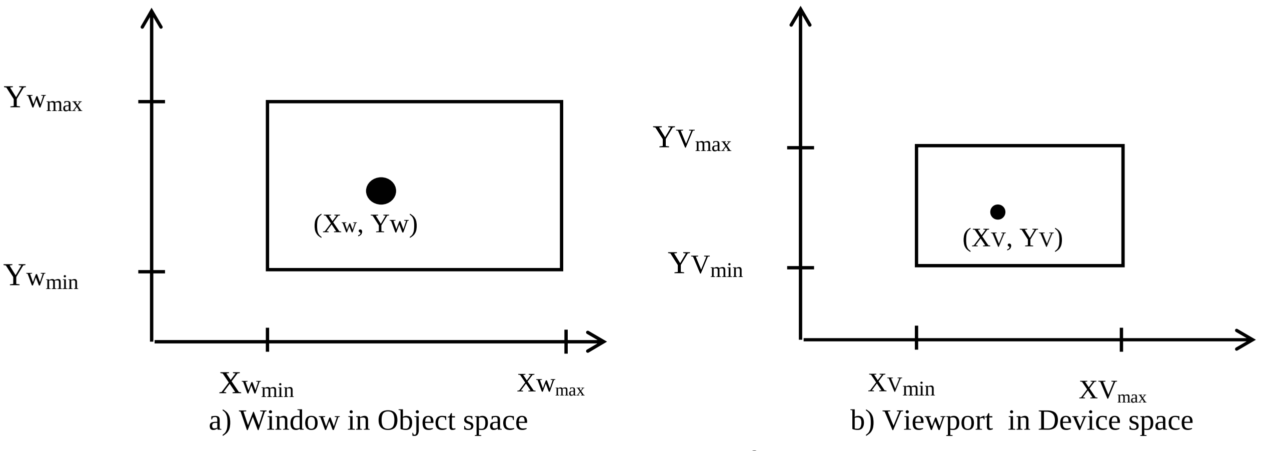


Figure 16: Window to Viewport Transformation

Figure 16, illustrates window-viewpoint mapping. Here, it is depicted that a point at position (X_w, Y_w) in window is mapped into position (X_v, Y_v) in the associated viewpoint.

So, as to maintain the same relative placement in the viewpoint as in the window we require

$$\left.\begin{aligned}\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} &= \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} \quad \text{-------------(a)}\\ \frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} &= \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}} \quad \text{-------------(b)}\end{aligned}\right\} \quad \text{-------------(1)}$$

rearranging equation (a) and (b) of (1) we get viewpoint position (x_v, y_v) i.e.,

$$\left(\begin{aligned}xv &= xv_{\min} + (xw - xw_{\min})\, S_x\\ yv &= yv_{\min} + (yw - yw_{\min})\, S_y\end{aligned}\right) \quad \text{------------(2)}$$

where

$$\left.\begin{aligned}S_x \text{ scaling factor along } x \text{ axis} &= \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}\\ S_y \text{ scaling factor along y axis} &= \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}\end{aligned}\right\} \quad \text{----------(3)}$$

Note, if $S_x = S_y$ then the relative proportions of objects are maintained else the world object will be stretched or contracted in either *x* or *y* direction when displayed on output device.

In terms of Geometric transformations the above relation can be interpreted through the following two steps:

Step 1 Scaling the window area to the size of the viewport with scale factors s_x and s_y w.r.t a fixed point ($xw_{\min}$, $yw_{\min}$).

$$\Rightarrow [T_1] = \begin{pmatrix} s_x & 0 & xw_{\min}(1 - s_x)\\ 0 & s_y & yw_{\min}(1 - s_y)\\ 0 & 0 & 1\end{pmatrix}$$

Step 2 Translating the scaled window to the position of the viewport so that,

$$\Delta x = xv_{\min} - xw_{\min} \qquad \text{and}$$
$$\Delta y = yv_{\min} - yw_{\min}$$

$$\Rightarrow [T_2] = \begin{pmatrix} 1 & 0 & \Delta x\\ 0 & 1 & \Delta y\\ 0 & 0 & 1\end{pmatrix}$$

Concatenating [T_1] and [T_2] we get,

$$[T] = [T_1]\,[T_2] = \begin{pmatrix} s_x & 0 & \Delta x + xw_{\min}(1 - s_x)\\ 0 & s_y & \Delta y + yw_{\min}(1 - s_y)\\ 0 & 0 & 1\end{pmatrix}$$

Replacing the values of Δx and Δy in the above transformation matrix we finally get,

$$[T] = \begin{pmatrix} s_x & 0 & -s_x xw_{\min} + xv_{\min}\\ 0 & s_y & -s_y yw_{\min} + yv_{\min}\\ 0 & 0 & 1\end{pmatrix} \qquad (1)$$

The aspect ratio of a rectangular window or viewport is defined by

$$a = \frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}}$$

If $s_x = s_y$ then $\frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}} = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$

$$\Rightarrow \frac{xv_{\max} - xv_{\min}}{yv_{\max} - yv_{\min}} = \frac{xw_{\max} - xw_{\min}}{yw_{\max} - yw_{\min}} \Rightarrow a_v = a_w$$

So, it can be said that if the aspect ratio a_v of the viewport equals the aspect ratio a_w of the window, then $s_x = s_y$ and no distortion of displayed scene occurs other than uniform magnification or compression. If $a_v \neq a_w$ then the displayed scene in the viewport gets somewhat distorted w.r.t the scene captured by the window.

Example Find the normalisation transformation *N* which uses the rectangle W (1, 1), X (5, 3), Y (4, 5) and Z (0, 3) as a window and the normalised device screen as the viewport.

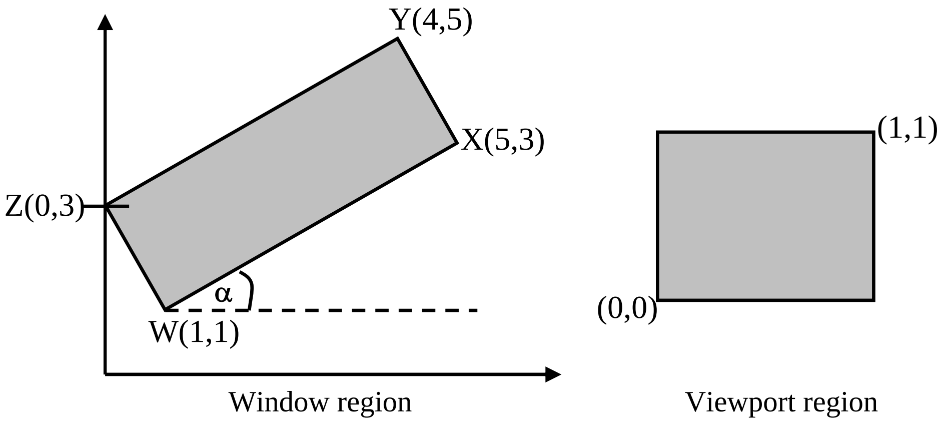


Figure 17: Example Transformations

Here, we see that the window edges are not parallel to the coordinate axes. So we will first rotate the window about W so that it is aligned with the axes.

$$\text{Now, } \tan\alpha = \frac{3-1}{5-1} = \frac{1}{2}$$

$$\Rightarrow \sin\alpha = \frac{1}{\sqrt{5}} \quad \cos\alpha = \frac{2}{\sqrt{5}}$$

Here, we are rotating the rectangle in clockwise direction. So α is (–)ve i.e., – α

The rotation matrix about W (1, 1) is,

$$[T_{R.\theta}]_W = \begin{bmatrix} \mathbf{Cos}\ \alpha & \mathbf{-Sin}\ \alpha & \mathbf{(1-Cos}\ \alpha\mathbf{)}\ x_p + \mathbf{Sin}\ \alpha\ y_p \\ \mathbf{Sin}\ \alpha & \mathbf{Cos}\ \alpha & \mathbf{(1-Cos}\ \alpha\mathbf{)}\ y_p - \mathbf{Sin}\ \alpha\ x_p \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$[T_{R.\theta}]_W = \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \left(\frac{1-3}{\sqrt{5}}\right) \\ \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & \left(\frac{1-1}{\sqrt{5}}\right) \\ 0 & 0 & 1 \end{pmatrix}$$

The *x* extent of the rotated window is the length of WX which is $\sqrt{(4^2 + 2^2)} = 2\sqrt{5}$

Similarly, the y extent is length of WZ which is $\sqrt{(1^2+2^2)}=\sqrt{5}$

For scaling the rotated window to the normalised viewport we calculate s_x and s_y as,

$$s_x = \frac{\text{viewport } x \text{ extent}}{\text{window } x \text{ extent}} = \frac{1}{2\sqrt{5}}$$

$$s_y = \frac{\text{viewport } y \text{ extent}}{\text{window } y \text{ extent}} = \frac{1}{\sqrt{5}}$$

As in expression (1), the general form of transformation matrix representing mapping of a window to a viewport is,

$$[T] = \begin{pmatrix} s_x & 0 & -s_x xw_{\min} + xv_{\min} \\ 0 & s_y & -s_y yw_{\min} + yv_{\min} \\ 0 & 0 & 1 \end{pmatrix}$$

In this problem $[T]$ may be termed as N as this is a case of normalisation transformation with,

$xw_{\min} = 1$	$xv_{\min} = 0$
$yw_{\min} = 1$	$yv_{\min} = 0$
$s_x = \frac{1}{2\sqrt{5}}$	$s_y = \frac{1}{\sqrt{5}}$

By substituting the above values in $[T]$ i.e. N,

$$N = \begin{pmatrix} \frac{1}{2\sqrt{5}} & 0 & \left(\frac{-1}{2}\right)\frac{1}{\sqrt{5}} + 0 \\ 0 & \frac{1}{\sqrt{5}} & \left(\frac{-1}{\sqrt{5}}\right)1 + 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Now, we compose the rotation and transformation N to find the required viewing transformation N_R

$$N_R = N\,[T_{R.\theta}]_W = \begin{pmatrix} \frac{1}{2\sqrt{5}} & 0 & \frac{-1}{2\sqrt{5}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{5} & 1-\frac{3}{\sqrt{5}} \\ \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 1-\frac{1}{\sqrt{5}} \\ 0 & 0 & 1 \end{pmatrix}$$

3.7 SUMMARY

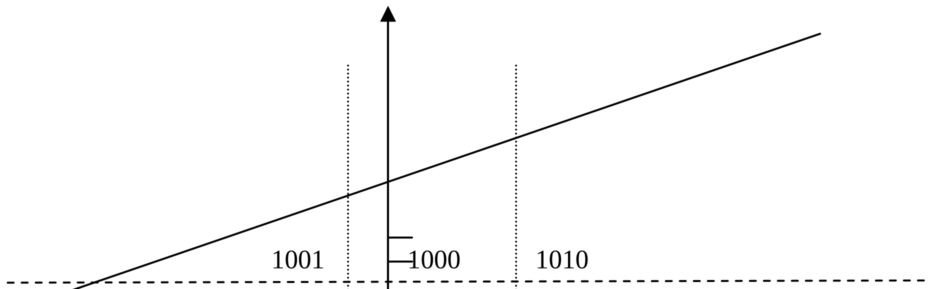
In this unit, we have discussed the concept of clipping, along with the concept of clipping we have also discussed various clipping types. The unit is quite important from the point of view of achieving realism through computer graphics.

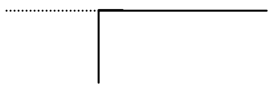
This unit contains algorithms, which are easy to implement in any programming language.

3.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Rectangular window ABCD is defined such that A = (–1, –2) and C (3, 1). So, to clip the line segment joining the points P(–20, 0) and Q(20, 30) using generalised geometrical approach we do the following tasks.





Equations of line segment PQ is:

y = mx + c where $m = \dfrac{y_2 - y_1}{x_2 - x_1}$ i.e $m = \dfrac{30-0}{20-(-20)} = \dfrac{30}{40} = 3/4$

$\therefore$ equation of line PQ $\Rightarrow$ y = 3/4x + c -------------(1)

As point P (-20, 0) lies on the line given by equation (1) so it should satisfy equation (1) i.e.,

0 = ¾ * (-20) + c $\Rightarrow$ c = 15 ------------------(2)

Using (2) in (1) we get the complete equation of line as

y = 3/4x + 15 ----------------------(3)

Steps to clip the line segment PQ are:

1) Point P is lying as LHS of window ABCD so by finding the intersection of line PQ with y = y_{max} = 1 we will get co-ordinates of P' using equation (3) put y = y_{max} = 1 in (3) we get,

   1 = 3/4x + 15 $\Rightarrow$ $x = \dfrac{-14*4}{3} = -18.66$

   i.e., P'(-18.66, 1), which is lying outside the window region ABCDhence reject portion PP' of line segment PQ.

2) Now consider altered line segment P'Q

   By finding the pt. of intersection of P'Q with x = x_{min} = -1 we will get p'',

   so put x = x_{min} = -1 in (3) we get

   Y = ¾(-1) + 15 = -3/4 + 15 = 57/4 = 14.25

   $\therefore$ $\Rightarrow$ P''(-1, 14.25) is also lying out of window region ABCD so reject portion P'P'' of line segment.

3) Now consider line segment P''Q

   We will get P''' if we will find the pt. of intersection of P''Q with x = x_{max} = 3

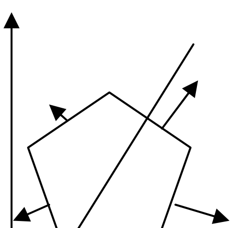
   $\therefore$by using x = x_{max} = 3 in (3) we get

   Y = ¾ * 3 + 15 = 9/4 + 15 = 69/4 = 17.25

   $\therefore$ $\Rightarrow$ P'''(3, 17.25) is also lying out of window region ABCD hence Q (20,30) is also lying out $\therefore$Reject whole line segment P'''Q.

   That is how a whole line segment PQ is rejected in small segments.

2) The clipping window is given by P_0(10,10), P_1(20,10), P_2(25,15), P_3(20,20), P_4(10,15), P_5 = P_0 and we need to clip the line A(0,0) and B(30,25). Using Cyrus Beck algorithm, we proceed as follows:



Steps to find clipping are:

1) Find $\overline{N}$ and using that find t:

$$t = \frac{-(\overline{P} - \overline{P_{i-1}}).\overline{N}}{(\overline{Q} - \overline{P}).\overline{N}}$$, $\overline{P}, \overline{Q}$ end pts. of line segment $\overline{P_{i-1}}$ starting pt of edge check t lies in the interval (0,1)

2) Find pt. of intersection $\overline{P} + t(\overline{Q} - \overline{P})$

3) If $(\overline{Q} - \overline{P}).\overline{N} < 0$ then pt. of intersection is potentially entering (P_E) if $(\overline{Q} - \overline{P}).\overline{N} > 0$ then pt. of intersections potentially leaving (P_L)

4) Find (max) t_{max} out of the value of t_{max} for P_E's and find(min) t_{mix} out of the value of t_{min} for P_L's condition of clip is $t_{max} < t_{min}$

5) Clipping is available from $\overline{P} + t_{\max}(\overline{Q} - \overline{P})$ to $\overline{P} + t_{\min}(\overline{Q} - \overline{P})$.

Now, let us do the above steps so as to clip the line $\overline{AB}$ in clipping windows p_0p_1, p_2, p_3, p_4.

edge 0: (i.e. edge P_0P_1)

Say $\overline{N_0} = (n_{01}, n_{02})$ be normal to this edge $\overline{p_0 p_1}$. So $\overline{N_0}.\overline{p_0 p_1} = 0$

$$\overrightarrow{p_0 p_1} = \overline{p_1} - \overline{p_0} = ((20,10) - (10,10) = (10,0) \qquad \text{--------------(1)}$$

$$\overline{N_0}.\overline{p_0 p_1} = (n_{01}, n_{02}).(10,0) = 0$$

$$= 10n_{01} + 0n_{02} = 0 \quad \Rightarrow \quad n_{01} = \frac{-0}{10} n_{02}$$

If we take $n_{02} = -1$ then $n_{01} = 0$ so, $\overline{N_0} = (0,1)$ --------------(2)

Now, to check that the point of intersection of line segment $\overline{AB}$ and edge is potentially entering or leaving find $\overline{AB}.\overline{N}$

$$\overline{AB} = \overline{B} - \overline{A} = ((30-0), (25-0)) = (30,25)$$

$$\overline{AB}.\overline{N_0} = (30,25).(0,1) = -25 < 0 \qquad so\ pt.us\ PE$$

$$t_{\max} = \frac{-(\overline{A} - \overline{P_0}).\overline{N_0}}{(\overline{B} - \overline{A}).N_0} = \frac{-((0,0)-(10,10)).(0,1)}{-25} = \frac{(10,10).(0,1)}{-25} = \frac{2}{5}$$

edge 1: (i.e. edge $\overline{P_1 P_2}$)

Say, $\overline{N_1} = (n_{11}, n_{12})$ be normal to this edge $\overline{p_1 p_2}$. So $\overline{N_1}.\overline{p_1 p_2} = 0$

$$\overrightarrow{p_1 p_2} = \overline{p_2} - \overline{p_1} = ((25,20) - (15,10) = (5,5) \qquad \text{--------(1)}$$

$$(n_{11}, n_{12}).\overline{p_1 p_2} = (n_{11}, n_{12}).(5,5) = 5n_{11} + 5n_{12} = 0 \quad \Rightarrow \quad n_{11} = -n_{12}$$

If $n_{12} = -1$ then $n_{11} = 1$ $\therefore \overline{N_1} = (n_{11}, n_{12}) = (1,-1)$ -------------(2)

Now let's find $\overline{AB}.\overline{N_1}$ to check that point of intersection is potentially entering or leaving

$\overline{AB}.\overline{N_1}$ = (30,25).(1,-1) = 30-25 = 5 > 0 so pt. is P_L

So, find t_{min};

$$t_{\min} = \frac{-(\overline{A}-\overline{P_1}).\overline{N_1}}{(\overline{B}-\overline{A}).N_1} = \frac{-((0,0)-(20,10)).(1,-1)}{5} = \frac{(20,10).(1,-1)}{5} = \frac{20-10}{5} = 2\,;$$

reject this point of intersection

Edge 2: (i.e. edge $\overline{P_2P_3}$)

Say $\overline{N_2} = (n_{21}, n_{22})$ be normal to this edge $\overline{p_2p_3}$. So $\overline{N_2}.\overline{p_2p_3} = 0$

$$\overrightarrow{p_2p_3} = \overline{p_3} - \overline{p_2} = ((20-25),(20-15) = (-5,5) \qquad \text{--------(1)}$$

$$\overline{p_1p_2}.\overline{N_2} = (-5,5).(n_{21}, n_{22}) = 0 \Rightarrow -5n_{21} + 5n_{22} = 0 \Rightarrow n_{21} = n_{22}$$

If $n_{21} = 1$ then $n_{22} = 1 \therefore \overline{N_2} = (n_{21}, n_{22}) = (1,1)$ -------------(2)

To check that pt. of intersection in P_L or P_E, angle sign of $\overline{AB}.\overline{N_2}$

$\overline{AB}.\overline{N_2}$ = (30,25).(1,1) = 30+25 > 0 so P_L

So, find t_{min};

$$t_{\min} = \frac{-(\overline{A}-\overline{P_2}).\overline{N_2}}{(\overline{B}-\overline{A}).N_2} = \frac{-((0,0)-(25,15)).(1,1)}{55} = \frac{(25,15).(1,1)}{55} = \frac{25-15}{55} = \frac{40}{55} = \frac{8}{11}$$

Edge 3: (i.e. edge $\overline{P_3P_4}$)

Say $\overline{N_3}$ be normal to $\overline{p_3p_4}$, then $\overline{N_3}.\overline{p_3p_4} = 0$

$$\overrightarrow{p_3p_4} = \overline{p_4} - \overline{p_3} = ((10,15)-(20,20) = (-10,-5) \qquad \text{--------(1)}$$

$$\overline{p_1p_2}.\overline{N_3} = (-10,-5)(n_{31}, n_{32}) = 10n_{31} + 5n_{32} = 0 \quad \Rightarrow \quad n_{31} = -1/2\, n_{32}$$

if $n_{32} = 1$ then $n_{31} = -1/2 \therefore \overline{N_3} = (-1/2, -1)$ -------------(2)

Now, let us find that pt. of intersection is P_E or P_L, lets check sign of $\overline{AB}.\overline{N_3}$

$\overline{AB}.\overline{N_3}$ = (30,25).(-1/2,-1) = 30(-1/2)+25(1) = -15+25 =10 > 0 so P_L

$$t_{\min} = \frac{-(\overline{A}-\overline{P_3}).\overline{N_3}}{(\overline{B}-\overline{A}).N_3} = \frac{-((0,0)-(20,20)).(-1/2,1)}{10} = \frac{(20,20).(-1/2,1)}{10} = \frac{-10+20}{10} = 1$$

Edge 4: (i.e. edge $\overline{P_4P_5}$ or $\overline{P_4P_0}$ ($\therefore$ $P_5 = P_0$)

Say $\overline{N_4}(n_{41}, n_{42})$ be normal to the edge $\overline{p_4p_0}$, so $\overline{N_4}.\overline{p_4p_0} = 0$

$$\overrightarrow{p_4p_0} = \overline{p_0} - \overline{p_4} = ((10,10)-(10,15) = (0,-5) \qquad \text{--------(1)}$$

$$\overline{p_1p_2}.\overline{N_4} = (0,-5)(n_{41}, n_{42}) = 0n_{41} - 5n_{42} = 0 \quad \Rightarrow \quad n_{42} = \frac{0n_{41}}{5}$$

If $n_{41} = -1$ then $n_{42} = 0 \therefore \overline{N_4} = (-1,0)$ -------------(2)

Now to find whether that point of intersection is potentially entering or leaving find and check sign of $\overline{AB}.\overline{N_4}$

$\overline{AB}.\overline{N_4}$ = (30,25).(-1,0) = -30+0 < 0 so P_E

Now find that (t_{max});

$$t_{\max} = \frac{-(\overline{A}-\overline{P_4}).\overline{N_4}}{(\overline{B}-\overline{A}).N_4} = \frac{-((0,0)-(10,15)).(-1,0)}{-30} = \frac{(10,15).(-1,0)}{-30} = \frac{-10}{-30} = \frac{1}{3}$$

Now out of all t_{max} find max. value of t_{max} and
out of all t_{min} find min. value of t_{min} and
$\max[t_{max}] = 2/5 = t_{max}$, $\min[t_{min}] = 8/11 = t_{min}$.

as $[t_{max}] < [t_{min}]$. So the line segment AB is clipped from

A + t_{max}(B-A) to A + t_{min}(B-A)

$\overline{A} + t_{\max}(\overline{B} - \overline{A})$

= (0,0) + 2/5[(30,25) - (0,0)] = (12, 10)

$\overline{A} + t_{\min}(\overline{B} - \overline{A})$

= (0,0) + 8/11[(30,25) - (0,0)] = (240/11, 200/11)

i.e., line is dipped from (12,10) to (240/11, 200/11), means

line AB is visible from (12,10) to (240/11, 200/11)

3) Let us compute the outward normal to a clipping in the case of Cyrus-Back algorithm. Consider a convex window, say it is hexagonal in shape; through which a line AB is passing and say $\overline{N_i}$ is normal at the edge of the hexagon where the line AB intersects i.e.,

$\overrightarrow{P_{i-1}P_i} = \overline{P_i} - \overline{P_{i-1}}$ is the edge with which the line segment AB intersects so,

$\overrightarrow{P_{i-1}P_i} = \overline{P_i} - \overline{P_{i-1}} = (x_i, y_i) - (x_{i-1}, y_{i-1})$ ----------------(1)

$= (x_i - x_{i-1}, y_i - y_{i-1}) = (s_1, s_2)$

If normal $\overline{N_i} = (n_{1i}, n_{2i})$ -------------------(2)

then, $\overrightarrow{P_{i-1}P_i}.\overrightarrow{N_i} = 0$

$(s_1, s_2).(n_{1i}, n_{2i}) = (s_1 n_{1i} + s_2 n_{2i}) = 0$

$n_{1i} = \dfrac{-s_2}{s_1} n_{2i}$

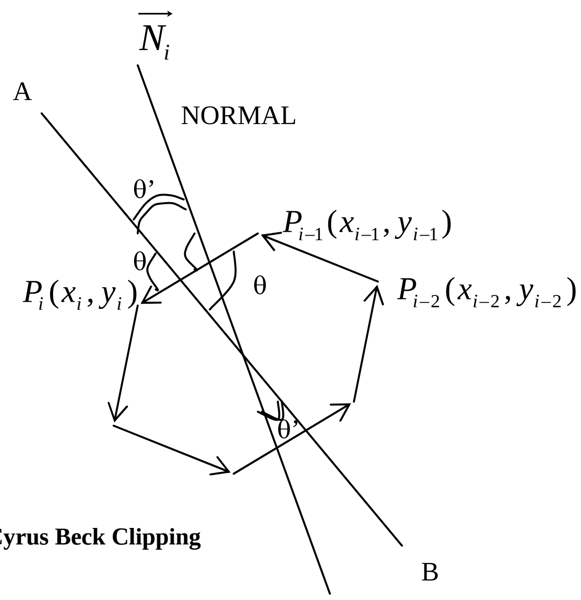


Figure 18: Example Cyrus Beck Clipping

Case1: If $n_{2i} = 1$ then $n_{1i} = \dfrac{-s_2}{s_1}$ $\quad \therefore \overline{N_i} = (-s_2/s_1, 1)$

Case2: If $n_{2i} = -1$ then $n_{1i} = \dfrac{s_2}{s_1}$ $\quad \therefore \overline{N_i} = (s_2/s_1, -1)$

Normal $\overline{N_i}$ in case 1 is opposite of the Normal in case 2 i.e. the direction is opposite.

So, if $\overline{N_i}$ in case is OUTWARD NORMAL then $\overline{N_i}$ in case2 will be INWARD NORMAL

In CYRUS-BECK Algorithm the point of intersection is either potentially entering on potentially leaving, lets study the two with inward/outward normal.

Case A: POTENTIALLY ENTERING CASE (*Figure 19*)

In *Figure19* the point of intersection given by $\overline{AB}$ is potentially entering

i) $\overline{N_i}$ is outward normal then,

$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos(\theta + 90°) = -|\overline{AB}||\overline{N_i}|\sin\theta < 0$$

ii) $\overline{N_i}$ is inward normal then,

$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos\theta' = |\overline{AB}||\overline{N_i}|\cos(90° - \theta) = -|\overline{AB}||\overline{N_i}|\sin\theta$$

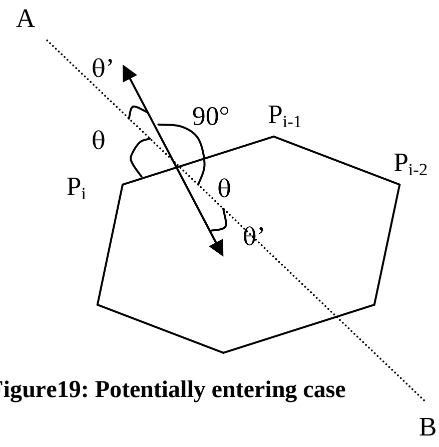


Figure19: Potentially entering case

Similarly ;

Case B: POTENTIALLY LEAVING CASE

When the point of intersection of line AB with edge $\overrightarrow{P_{i-1}.P_i}$ is potentially leaving.

i) $\overline{N_i}$ is outward normal then,

$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos\theta' = |\overline{AB}||\overline{N_i}|\cos(90° - \theta) = |\overline{AB}||\overline{N_i}|\sin\theta > 0$$

ii) $\overline{N_i}$ is inward normal then,

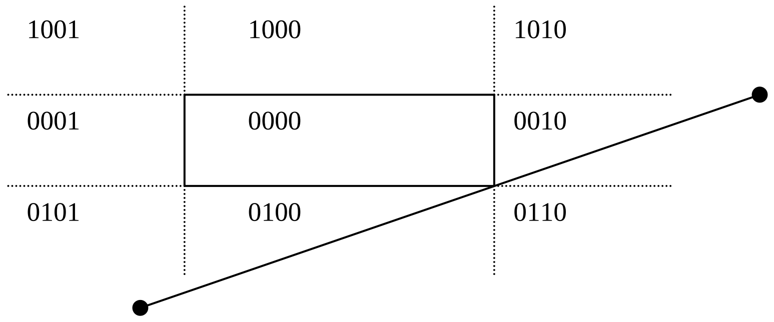
$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos(90° + \theta) = -|\overline{AB}||\overline{N_i}|\sin\theta < 0$$

i.e., by analyzing the sign of dot product of $\overline{AB}$ with $\overline{N_i}$ in case of potentially entering/leaving pt. we can find that normal $\overline{N_i}$ is outward normal or inward normal.

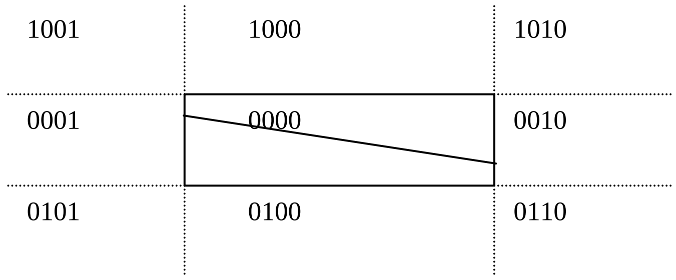
4) Limitations of Cohen-Sutherland clipping algorithm are:

1) Clipping window region can be rectangular in shape only and no other polygonal shaped window is allowed.
2) Edges of rectangular shaped clipping window has to be parallel to the x-axis and y-axis.
3) If end pts of line segment lies in the extreme limits i.e., one at R.H.S other at L.H.S., and on one the at top and other at the bottom (diagonally) then, even

if the line doesn't pass through the clipping region it will have logical intersection of 0000 implying that line segment will be clipped but infact it is not so.



5) Using Cohen Sutherland Line clipping algorithm when we clip the line we get the line shown in *Figure* below as output.



6) Window regions which are Non convex can't be handled by the CYRUS-BECK ALGORITHM without modification.

As we know that in Cyrus Beck algorithm the line is to draw if tmax < tmin else it is to be rejected and as per equation (1) and Figure we find that the whole line LM will be rejected but some portion of LM has to be visible, to achieve this, the algorithm has to be modified so as to handle new convex window regions.

Modifications to be applied to Cyrus Beck algorithm to handle the non convex region case are:

1) The non convex region is to be divided into many convex sub regions.

2) tmax and tmin are to be found w.r.t individual convex window region hence line is to be clipped for different convex window regions.

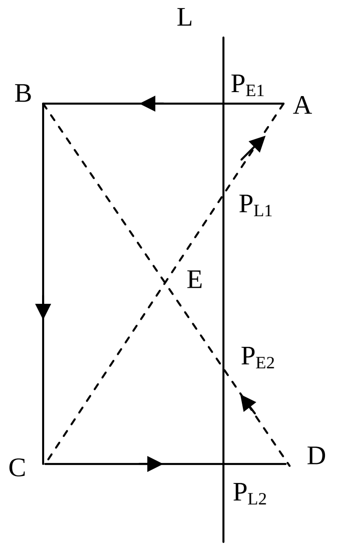
As per the modification the window region ABCDE is to be divided into two convex windows.

AEBA and BCDEB

Or

CDEC and ABCEA

ABCDE → Non convex window3
LM → line to be clipped
P_E → Potentially entering
P_L → Potentially leaving



Check Your Progress 2

1) Clipping the given polygon using Sutherland Hodgeman polygon clipping algorithm we will get the result given in the *Figure* below as output.