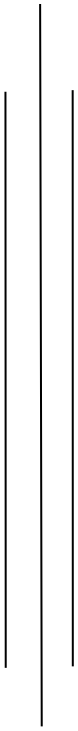
Compiled by: ***Er. Ganesh Ram Suwal***

(Affilated to Purbanchal University)

Khwopa Engineering College

Libali-2, Bhaktapur



A Complete Manual Of

Computer Graphics

BE Computer \ Electronics (Sixth Semester)

2008

Prepared By:

Er. Ganesh Ram Suwal

Lecturer

Khwopa Engineering College

Chapter 1

1.0 Introduction:

Computer graphics is one of the most exciting and rapidly growing computer fields. The computer graphics field is related to the generation of graphics using computers. It includes the creation, Storage, and manipulation of image objects. These objects come from diverse fields such as physical, mathematical, engineering, architectural, abstract structures and natural phenomenon. Computer graphics today is largely interactive that is, the user controls the contents, structure, and appearance of images of the objects by using input devices. Such as a keyboard, mouse, or touch sensitive panel on the screen.

Until 1980's computer graphics was a small specialized field, largely because the hardware was expensive and graphics-based application programs that were easy to use and cost-effective were few. Then, PC (Personal computers) with build in raster graphics displays such as the **Xerox Star Apple Macintosh** and **IBM PC**-popularized the use of bitmap graphics for users computer interactions. A bitmap is ones (1) and zeros (0) representation of the rectangular array points on the screen. Each point is called **Pixel** or **Pel (**shortened forms of **Picture Elements).** Once bitmap graphics became affordable, an explosion of easy to use and inexpensive graphics based applications soon followed. Graphics-based user interfaces allowed millions of new users to control simple, low-cost application programs, such as word processors, spreadsheets, and drawing programs.

The concept of a "*desktop*" now became a popular metaphor for organizing screen space. By means of a window manager, the user could create, position, and resize rectangular screen areas called windows. This allows user to switch among multiple activities just by pointing

and clicking at the desired window, typically with a mouse. Besides windows, icons which represent data files, application program, file cabinets, mailboxes, printers, recycle bin, and so on, made the user computer interaction more effective. By pointing and clicking icons, user could activate the corresponding programs or objectives, which replace much of the typing of the commands, used in earlier operating systems and computer applications. Today, almost all interactive application programs, even those for manipulating text (e.g. word processor) or numerical data (e.g. spreadsheet programs) use graphics extensively in the user interface and for visualizing and manipulating the application-specified objects.

Even people who do not use computers encounter computer graphics in TV commercial and cinematic special effects. Thus computer graphics is an integral part of all computer user interfaces, and is indispensable for visualizing 2D, 3D objects in almost all areas such as education, science, engineering, medicine, commerce, the military, research, advertising and entertainment. The theme is that learning how to program and use computers now includes learning how to use simple 2D graphics.

1.1 Early history of computer graphics:

We need to take a brief look at the historical development of computer graphics to place today's system in context. Crude plotting of hardcopy devices such as teletypes and line printers dates from the early days of computing. The **Whirlwind** computer developed in 1950 at the **Massachusetts Institute of Technology (MIT)** had computer driven CRT displays for output.

The **SAGE** air-defense system developed in middle 1950s was the first to use command and control CRT display consoles on which operators identified targets with light pens (hand held pointing devices that sense light emitted by objects on the screen).

Later on Sketchpad system by Ivan Sutherland came in light. That was the beginning

sys

and

At the same time, it was becoming clear to computer, automobile, and aerospace manufactures that CAD (Computer-Aided Design) and CAM (Computer-Aided Manufacturing) activities had enormous potential for automating drafting and other drawing-intensive activities. The General Motors DAC system for automobile designs and the Itek-Digitek System for lens designs were pioneering efforts that showed the utility of graphical interaction in the iterative design cycles common in engineering. By the mid-60s, a number of commercial products using these systems had appeared.

At that time only the most technology-intensive organizations could use the interactive computer graphics whereas others used punch cards, a non-interactive system.

Among the reasons for this were these:

- The high cost of the graphics hardware: at a time when automobiles cost a few thousand Dollars, computer cost several millions of Dollars, and the first commercial computer displays cost more than a hundred thousand Dollars.
- The need for large scale, expensive computing resources to support massive design database.
- The difficulty of writing large, interactive programs using batch-oriented FORTAN programming.
- One-of-a-kind non portable software, typically written for a particular manufacturer's display devices. When software is non portable, moving a new display device necessitates expensive and time consuming rewriting of working programs.

Thus interactive computer graphics had a limited use when it started in the early sixties. But it became very common once the Apple Macintosh and IBM PC appeared in the market with affordable cost.

1.2 Application of computer graphics:

Computer graphics is used today in many different areas of science, engineering, industry, business, education, medicine, art and training. All of these are included in the following categories.

1. User Interface.

Most applications have user interfaces that rely on desktop window systems to manage multiple simultaneous activities, and on point-and-click facilities to allow user to select menu items, icons, and objects on the screen. These activities fall under computer graphics. Typing is necessary only input text to be stored and manipulated. For example Word processor, Spreadsheet, and desktop-publishing programs are the typical examples where user-interface techniques are implemented.

2. Plotting

Plotting 2D, 3D graphics of mathematical, physical and economics functions use computer graphics extensively. The histogram, bar, and pi-chart; the task-scheduling charts are the most commonly used plotting. These all are used to present meaningful and concisely the trends and patterns of complex data.

3. Office automation and electronics publishing

The computer graphics has facilitated the office automation and electronics publishing, which is also popularly known as desktop publishing, giving more power to the organization to print the meaningful material in house. Office automation and electronics publishing can produce both traditional print (hardcopy) documents and electronics (softcopy) documents that contain text, tables, graphs, and other form of drawn or scanned in graphics.

4. Computer aided drafting and design

One of the major uses of computer graphics is to design component and systems of mechanical, electrical, electrochemical, and electronics devices, including structure such as building, automobile bodies, airplane and ship hull, very large-scale integrated (VLSI) chips, optical systems, and computer networks. These designs are more frequently used to test the structural, electrical, and thermal properties of the systems.

5. Scientific and business visualization

Generating computer graphics for scientific, engineering, and medical data set is termed as scientific visualization whereas business visualization is related with non scientific data set such as those obtained in economics. Visualization makes easier to understand the trends

and patterns inherent in the huge amount of data sets. It would otherwise be almost impossible to analyze those data numerically.

6. Simulation

Simulation is the imitation of the conditions like those, which is encountered in real life. Simulation thus helps to learn or feel the conditions; one might have to face in near future without being danger at the beginning of the course. For example, astronauts can exercise the feeling of weightlessness in a simulator. Similarly a pilot training can be conducted in a flight simulator. The military tank simulator, the naval simulator, driving simulator, air traffic control simulator, heavy duty vehicle, and so on are some of the mostly used simulator in practice. Simulators are also used to optimize the system. For example the vehicle, observing the reaction of the driver during the operation of the simulator.

7. Entertainment

Disney movies such as Lion Kings and The Beauty and the Best, and other scientific movies like Star Trek, are the best examples of the application of computer graphics in the field of entertainment. Instead of drawing necessary frames with slightly changing scenes for the production of cartoon film, only the key frames are sufficient for such cartoon-film where in-between frames are interpolated by the graphics system dramatically decreasing the cost of production while maintaining the quality. Computer and video games such as Fifa, Formula-1, Superbike, and Moto are few to name where computer graphics is used extensively.

8. Art and commerce

Here computer graphics is used to produce picture that express a message and attract attention such as a new model of a car moving alone the ring of the Saturn. These pictures are frequently seen at transportation terminals, supermarkets, hotels, etc. the slide production of commercial, scientific, or educational presentations is another cost effective use of computer graphics. One of such graphics package is "Power-point".

9. Cartography

Cartography is a subject, which deals with making map and charts. Computer graphics is used to produce both accurate and schematic representations of geographical and other

natural phenomenon from measurement data. Example includes geographic maps, oceanographic charts, weather maps, contour maps and population-density maps. Surfer is one such graphics packages, which is extensively used for cartography.

Image Processing

- Computer graphics is used to create a picture while image processing is used to modify or interpret existing pictures such as photographs and TV Scans. Two principle use in image processing are
    1. improving picture quality
    2. machine perception of visual information as used in robotics
- In image processing photograph is first digitize into an image file and then the rearrangement picture parts, to enhance color separations or to improve the quality of shading.
- In medical application image processing is used to enhance the photograph for example "tomography" and simulation operation. Tomography is a technique of X-ray photography that allows cross sectional views of physiology system to be displayed.

```
/* Program of Basic Graphical Shapes */
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int gd=DETECT,gm;
void main(){
  int a,b,c,d,e,f,x;
  initgraph( &gd, &gm," ");
  cleardevice();
  setbkcolor(BLACK);
  printf("\t\tGive the choise u want to make");
  printf("\n\t\t 1 pixel");
  printf("\n\t\t 2 line");
  printf("\n\t\t 3 rectangle");
```

```
printf("\n\t\t 4 circle");
printf("\n\t\t 5 Ellipse");
printf("\n\t\t 6 pieslice");
printf("\n\t\t 7 3 d bar");
scanf("%d",&x);
cleardevice();
switch(x)  {
  case 1:
   printf("\n\tGive the value x-axis");
   scanf("%d",&a);
   cleardevice();
   line(a,a,a,a);
   break;
  case 2:
   printf("\n\tGive the corodinate of the line x1,y1,x2,y2\t");
   scanf("%d %d %d %d",&a,&b,&c,&d);
   cleardevice();
   line(a,b,c,d);
   break;
  case 3:
   printf("\n\tGive the left, top, right and bottom value\t");
   scanf("%d %d %d %d",&a,&b,&c,&d);
   cleardevice();
   rectangle(a,b,c,d);
   break;
  case 4:
   printf("\n\tGive the x-axis and Y-axis");
   printf(" center valuse and radii\t");
   scanf("%d%d%d%d",&a,&b,&c);
   cleardevice();
   circle(a,b,c);
   break;
```

```
    case 5:
      printf("\n\tGive Center of ellipse X-axis");
      printf(" and Y-axis starting anlge\n");
      printf("end angle Horizontal axis X-axis and");
      printf(" Vertical axis Y- axis \t");
      cleardevice();
      scanf("%d%d%d%d%d%d",&a,&b,&c,&d,&e,&f);
      ellipse(a,b,c,d,e,f);
      break;
    case 6:
      printf("\n\tGive Center of pieslice X-axis ");
      printf("and Y-axis starting anlge");
      printf("\nend angle radii \t");
      scanf("%d%d%d%d%d%d",&a,&b,&c,&d,&e);
      cleardevice();
      pieslice(a,b, c,d,e);
      break;
    case 7:
      printf("\n\tGive left, top, right, bottom, ");
      printf("dept and topflag cordinates");
      scanf("%d%d%d%d%d%d",&a,&b,&c,&d,&e,&f);
      cleardevice();
      bar3d(a,b,c,d,e,f);
      break;
   }
  getch();
  closegraph();
}
```

Chapter 2

2.0 Hardware Concepts:

Since a computer is an electronic machine, so without any input to a computer it doesn't work anything. An input device is an electromechanical device, which accepts data from the outside world, and translates them into a form, which the computer can interpret. Data input devices like keyboards are used to provide additional data to the computers whereas pointing and selection devices like mouse, light pens, touch panels are used to provide visual and indication-input to the application.

2.1 Keyboard, Mouse, Light pen, Touch screen and Tablet

1. keyboard:

Keyboard is a device primarily used for entering text string, i.e. keyboard is an efficient device for inputting non graphics data. It is mainly used to input textual data. It generates a unique ASCII code corresponding to each key pressed or combination. It usually consists of alphanumeric keys, function keys, cursor-keys, and a separate numeric key pad. It is used to move cursor, select main item, enter screen coordinate and several other functions. In some keyboard (multimedia keyboard) there are extra multimedia keys. These keys are used to move the cursor, to select the menu items, predefined functions. The graphics keyboard is mainly used for entering screen coordinates and text, to invoke certain functions. Now-a-days ergonomically designed keyboard (Ergonomic keyboard) with removable palm rests is available. The slope of each half of the keyboard can be adjusted separately. Ergonomics is an applied science that coordinates the design of devices, systems and physical working conditions with the capacities and requirements of the workers. It is also called as human engineering.

Figure: Wireless Keyboard and Mouse

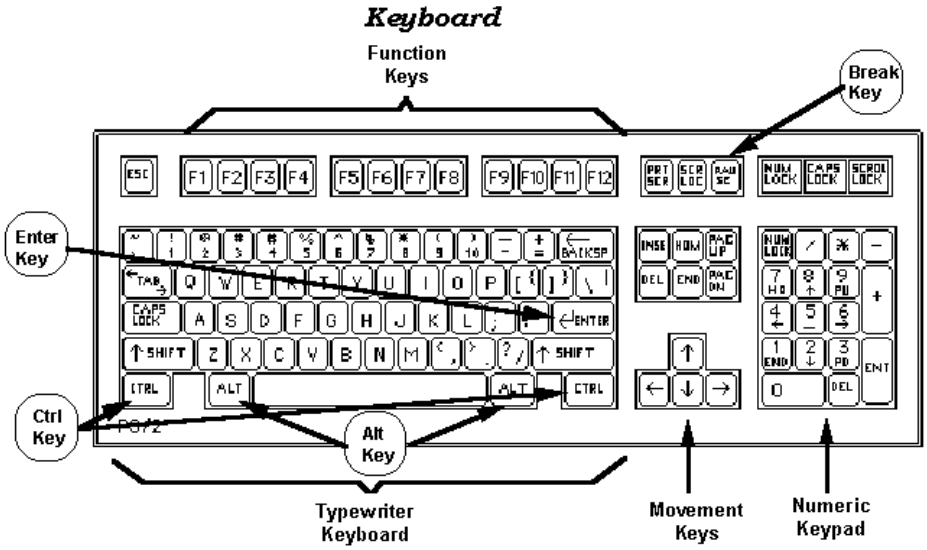


Fig: a simple keyboard.

2. Mouse:

Mouse is a small hand held device used to position the cursor on the screen. Wheels or rollers on the bottom of the mouse can be used to report the amount and direction of movement. Another method for detecting mouse motion is with an optical sensor. Mouse is a relative device, i.e. it can be picked up, moved in space, and put down again without any change in the reported position. For this computer maintains the current mouse position, which is incremented or decremented by the mouse movements. Following are the mostly used mice in computer graphics.

A. Mechanical Mouse

When a roller in the base of this mechanical mouse is moved, a pair of orthogonally arranged toothed wheels, each placed in between a LED and a photo detector, interrupts the light path. An optical detector counts the pulses. These pulses generated in the horizontal and vertical directions move the cursor on the screen. Hence, the numbers of interrupts so generated are used to report the mouse movements to the computer.

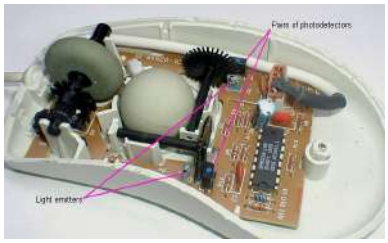


Fig: a mechanical mouse

B. Optical Mouse

Instead of rolling the ball, the optical mouse uses the light beam to detect the mouse movement across a specific patterned. It is used on a special pad having a grid of alternating light and dark lines. A LED on the bottom of the mouse directs a beam of light down onto the pad, from which it is reflected and sensed by the detectors on the bottom of the mouse. As the mouse is moved, the reflected light beam is broken each time a dark line is crossed generating the number of pulses which is proportional to the movement of mouse. Hence, the number of pulses so generated, which is equal to the number of lines crossed, is used to report mouse movements to the computer.

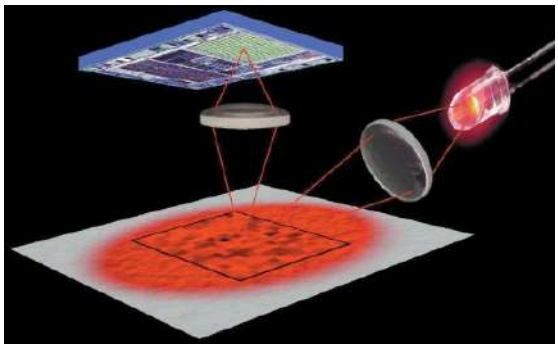


Fig: a *Fig: b*

Fig: a and b showing cross-sectional view of optical mouse

Advantages of optical mouse:

1. The cursor moves smoothly.
2. There are no moving parts; so there is less chance of failure.
3. No mechanical problem due to dusts, etc.

3. Light pen:

A **light pen** is a pencil shaped device and has a chord at the trailing end. It is a computer input device in the form of a light-sensitive wand used in conjunction with the computer's CRT monitor. It lets the user select screen position by detecting the light coming from point in the CRT screen. It also allows the user to point to displayed objects, or draw on the screen, in a similar way to a touch screen but with greater positional accuracy. They are sensitive to short bursts of light emitted from the phosphor coating at the instant electron beam strikes a particular point. An activated light pen pointed at a spot on a screen as the electron beam lights up the spot, generates an electrical pulse that causes the coordinate position of the electron beam to be recorded. The light pen when pointed to the screen detects the bright/dim effect and when light goes from dim to light, it sends a signal pulse to the video chip. The video chip sets a latch which feeds two numbers: X-location, Y-location into a memory location and can tell where the light pen is pointed on the screen by two numbers. A light pen can work with any CRT-based monitor, but not with LCD screens, projectors and other display devices.

A light pen is fairly simple to implement. The light pen works by sensing the sudden small change in brightness of a point on the screen when the electron gun refreshes that spot. By noting exactly where the scanning has reached at that moment, the X, Y position of the pen can be resolved. This is usually achieved by the light pen causing an interrupt, at which point the scan position can be read from a special register, or computed from a counter or timer. The pen position is updated on every refresh of the screen. Due to the fact that the user was required to hold his or her arm in front of the screen for long periods of time, the light pen fell out of use as a general purpose input device.

Fig: showing light pen

Drawbacks:

- Prolong use of the light pen can cause arm fatigue.
- It gives sometimes false reading due to background lightning in a room
- It cannot report the coordinates of a point that is completely black as a remedy one can display a dark blue field in place of the regular image for a single frame time.
- Light pen obscures the screen images as it is pointed to required spot.
- When a light pen is pointed at the screen, part of the screen image is obscured by the hand and pen and prolonged use of the light pen can cause arm fatigue.
- It requires special implementations for some applications because they cannot detect positions within black areas.

4. **Touch Panel:**

Touch panel allows the user to directly point to the screen with the touch of the finger to move the cursor or to select the menu item. Its application is in processing of options where graphical icons are given. The touch input can be recorded using optical, electrical or acoustical methods. There are three kinds of touch panels:

a) Optical Touch Panel

b) Electrical Touch Panel

c) Acoustic Touch Panel

A. Optical touch panel

It uses a series of infra-red emitting diodes (LED) along one vertical edge and along one horizontal edge of the panel, that is, infra-red lens are employed along the vertical edges and one horizontal edge of the frame. The opposite vertical and horizontal edges contain photo detectors or infra-red sensors to form a grid of invisible infra-red light beams over the display area. When the user touches the screen, it breaks one or two vertical and horizontal light beams falling on the screen thereby indicating the fingers positions which determines the point of contact. The cursor is then moved to this position or the icon at this position is selected. With closely spaced LEDS, it is possible to break two horizontal or two vertical beams simultaneously. In that case, the average position between two interrupted beams is recorded. The LEDS operate at infra-red frequency so that the light is not visible to the user. This is a low resolution panel which offers 10 to 50 positions in each direction.

An optical touch panel is provided which includes a plurality of light-emitting elements, reflectors for reflecting the emitted light, and a plurality of light-receiving elements for receiving the reflected light. The plurality of light-emitting elements and the plurality of light-receiving elements are arranged alternately along each of first and second adjacent sides of a rectangular position-detecting surface, and the reflectors are arranged along each of third and fourth adjacent sides of the position-detecting surface. A drive controller causes the plurality of light emitting elements to light in a predetermined order to thereby cause respective ones of the light-receiving elements arranged on opposite sides of each of the plurality of light-emitting elements to receive the light reflected by the reflectors. Hollow cylinders opposed to the respective light-receiving elements inhibit light other than the light reflected by the reflectors from being incident on the plurality of light-receiving elements.

Fig: Optical Touch Panel

Sonic panel

Bursts of high frequency sound waves travelling alternately horizontally and vertically are generated at the edge of the panel. When a user touches the screen, the finger causes part of the wave to be reflected back from the finger to the emitters/sources. The screen position at the point of contact is then calculated from a measurement of a time interval between the transmission of each wave and its reflection to the emitter i.e. the time between emission of the wave and its arrival at the source itself. This is also the high resolution touch panel having about 500 positions in each direction.

B. Electrical touch panel

It consists of slightly separated two transparent plates one is coated with a thin layer of conducting material and the other with resistive material. When the panel is touch with a finger, the two plates are forced to touch at the point of contact thereby creating the voltage drop across the resistive plate which is then used to calculate the coordinates of the touch position. This is also the high resolution touch panel having about 500 positions in each direction which is similar to that of sonic touch panel.

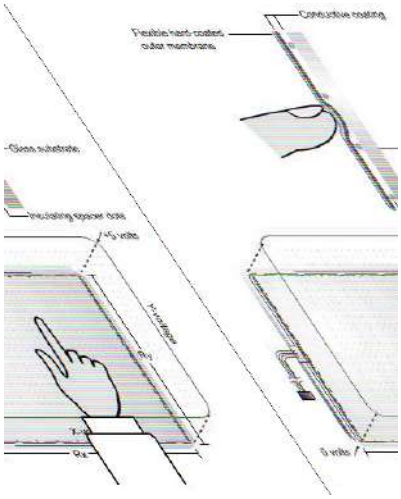


Fig: touch panel

5. Tablet:

A tablet is a *digitizer*. In general, a digitizer is a device which is used to scan over an object and to input a set of discrete coordinate positions. These positions can then be joined with straight line segments to approximate the shape of the original object. A tablet digitizes an object detecting the position of a movable stylus (pencil-shaped device) or puck (like mouse with cross hairs for sighting positions) held in the user's hand. It is a common device for drawing, painting or interactively selecting coordinate position on an object. One type of digitizer is the **graphic tablet**. It is used to enter two dimensional coordinates by activating a hand cursor or stylus at selected position on a flat surface. A tablet is a flat surfaces ranging from 6 x 6 inches to 48 x 72 inches which can detect the position of movable of stylus or pack held in user hand. The accuracy of tablets usually falls below 0.2 mm.

There are three types of digitizer:

a) Electrical Tablet

b) Sonic Tablet

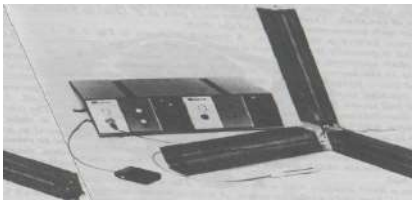
c) Resistive Tablet

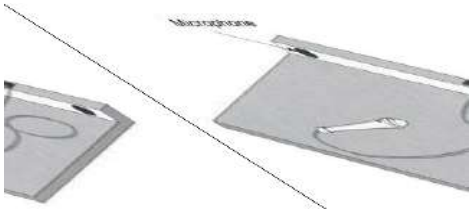
Electrical Tablet:

A grid of wires on ¼ to ½ inch center is embedded in the tablet surface, and electromagnetic signals generated by the electric pulses applied in the sequence to the wire in the grid induce electrical signals in a wire coil in the stylus (or puck). The strength of the signal induced by each pulse is used to determine the position of the stylus. The signal strength is also used to determine roughly how far the stylus is from the tablet. When the stylus is within ½ inch from the tablet, it is taken as "near" otherwise it is either "far" or "touching". When the stylus is "near" or "touching", a cursor is usually shown on the display to provide visual feedback to the user. A signal is sent to the computer when the tip of the stylus is pressed against the tablet, or when any button on the puck is pressed. The information provided by the tablet repeats 30 to 60 times per second.

Sonic Tablet

The sonic tablet use sound waves to couple the stylus to microphones positioned on the periphery of the digitizing area. An electric spark at the tip of the stylus creates sound bursts. The position of the stylus or coordinate value s is calculated using the delay between when the spark occurs and when its sound arrives at each microphone. The main advantage of sonic tablet is that it doesn't require the dedicated working area for the microphone can placed on any surface from the "tablet" work area. These facilitate digitizing drawing on thick books, because in an electrical tablet this is not convenient for the stylus cannot get closer to the tablet surface.





Resistive Tablet

The resistive tablet consists of a piece of glass coated with thin layer of conducting material. When a battery-powered stylus is activated at certain position, it emits high frequency radio signals, which induce the radio signal on the conducting layers. The strength of the signal received at the edges of the tablet is used to calculate the position of the stylus.

When the stylus is activated at certain point, it emits high frequency radio signal which induces radio signal into the conducting layer. The strength of the signal at the edges of the tablet determines the position of the stylus.

Several types of tablets are transparent and thus can be backlit for digitizing x-rays film and photographic negatives, the resistive tablet can be used to digitize the objects on CRT

because it can be curved to the shape of the CRT. The mechanism of sonic tablet or electrical tablets can be used to digitize the 3D objects, while the resistive tablets can be used to digitize on CRT because it can be curved to shape of the CRT.

2.1. Raster and Random display architecture

Raster Scan displays

Raster display technology is developed in early 70's based on TV (television technology) technology. The electron beam in raster scan display is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turn on as 1 and off as 0 to create a pattern of illuminated spots. Picture definition is stored in a memory called the **refresh buffer** or **frame buffer**. This memory holds the set of intensity values for all the screen points. The intensity values are retrieving from the buffer and repaint on the screen one row at a time. The intensity range for pixel positions depends on the capability of the raster system. In monochrome monitor frame buffer consists of one bit for each pixel (i.e. either 1 for on or 0 for off) and for color monitor or for variation of intensity of each pixel frame buffer require additional bits consists up to 24 bits for each pixel for high quality system. Which require several megabytes of storage for frame buffer, depending upon the resolution of the system.

For example:

If resolution of a display is 1024*1024,

24 bits are included for each pixel.

i.e. 8 bits for RED, 8 bits for GREEN, 8 bits for BLUE, then

Frame buffer require = (1024*1024*24) bits

= (1024*1024*24)/8 bytes

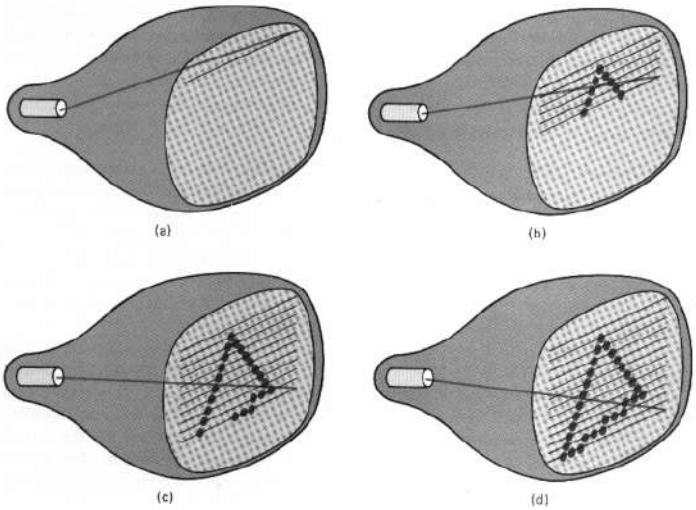
= (1024*1024*24)/(8*1024*1024) Mega bytes

=3 MB

And the capability of color combination of that display is 2^{24} = 16 millions of color combination

The refresh rate of raster scan display is typically 60 to 80 frames per seconds. That means the redrawn of image is 60 to 80 so that user view the non flickering image.

Some of the rater scans display system each frame is displayed in two pass using an interlaced refresh procedure. It is primarily used lower refresh rate for preserving the phosphor from burn out.



Raster Display Technology

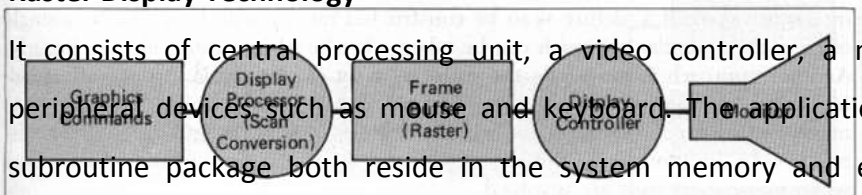


FIGURE 2–30
raster-scan system.

It consists of central processing unit, a video controller, a monitor, system memory, peripheral devices such as mouse and keyboard. The application program and graphical subroutine package both reside in the system memory and execute on CPU. When a particular command such as a line(x1, y1, x2, y2) is called by the application program, the graphics subroutine package sets the appropriate pixels in the *frame buffer*, a portion of the system memory. The *video controller* then cycles through the frame buffer, one scan line at a time typically 50 times per second. It brings a value of each pixel contained in the buffer and uses it to control the intensity of the CRT electron beam. So there exists a one to one relationship between the pixel in the frame buffer and that on the CRT screen.

Figure: Simplified Raster Display Architecture

A 640 pixels by 480 lines is an example of *medium resolution* raster display. A 1600 by 1200 is a *high resolution* one. A pixel in a frame buffer may be represented by one bit as in monochromatic system where each pixel on CRT screen is either ON '1' or OFF '0' or it may be represented by eight bits resulting 2^8 = 256 gray levels for continuous shades of gray on CRT screen. In color system, each of the three colors – red, green and blue is represented by eight bits producing 2^{24} = 16 million colors. A medium resolution color display having 640 x 480 pixels will thus require (640 x 480 x 24)/8 = 9 kb of RAM.

Advantages

- It has an ability to fill the areas with solid colors or patterns.
- The time required for refreshing is independent of complexity of an image
- Low cost

Disadvantages

- For Real-Time dynamics not only the end points are required to move but all the pixels in between the moved end points have to be scan converted with appropriate algorithms which might slow down the dynamic process.
- Required special algorithm to move all pixels.
- Due to scan conversion "*jaggies*" or "*stair-casing*" are unavoidable.

Video Controller:

It is a special purpose processor. It accesses the frame buffer to refresh the screen. It is given direct access to the frame buffer memory. Some transformation such as enlargement, reduction, or movement from one location to another can also be accomplished with the video controller. Some systems are designed to allow the video controller to mix the frame buffer image with an input from television camera or other input device.

Raster Display Processor:

The raster display with a peripheral display processor is a common architecture that avoids the disadvantage of simple raster display system. It includes a separate graphics processor to perform graphics functions such as scan conversion and raster operation and a separate frame buffer for image refresh. The display processor has its own separate memory called display processor memory. The purpose of the display processor is to free the CPU from the graphics chores. The display processor digitizes the picture definition given in an application program into a set of pixel intensity values for storage in the frame buffer. This digitization is called as scan conversion.

System memory holds data and those programs that execute on the CPU, and the application program, graphics package and OS. The display processor memory holds data plus the program that perform scan conversion and raster operations. The frame buffer

stores displayable image created by scan conversion and raster operations. The organization is given below in figure:

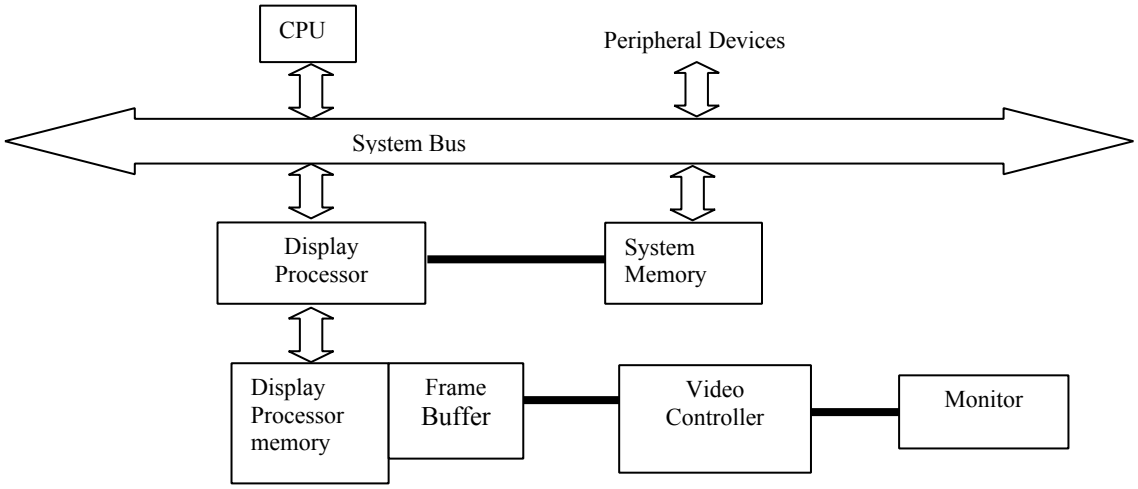
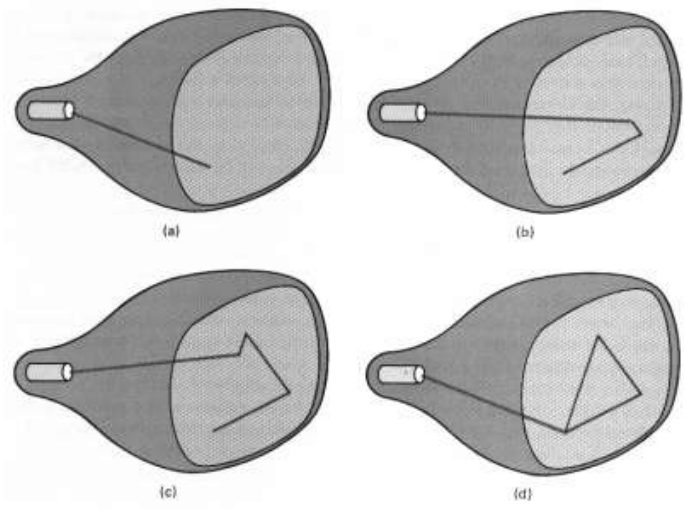


Figure: Raster Display Architecture with Display Processor

Random Scan Displays (vector display technology)

Random/Vector display technology was developed around 60's and used as a common display device until 80's. It is also called *a random scan, a stroke, a line drawing, or a calligraphic display*. The architecture of a simple vector display is shown in fig below.



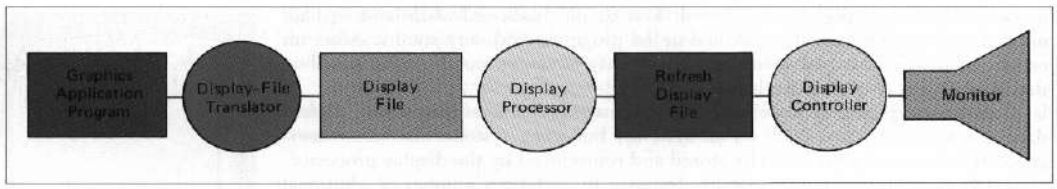


FIGURE 2-28
Block diagram of the functions performed by a random-scan system with a display controller.

Figure: The architecture of a simple vector display

It consists of a central processing unit (CPU), a display processor, a monitor, system memory and peripheral devices such as mouse and keyboard. A display processor is also called *a display processing unit (DPU), or graphics controller*. The application program and *graphics subroutine package* both reside in the system memory, and execute on CPU. A graphics subroutine packages create a display list, and stores in the system memory. A display list contains point and line plotting commands with end point coordinates as well as character plotting commands. The DPU interprets the commands in the display list and plots the respective output primitives such as point, line and characters. As a matter of fact, the DPU sends digital point coordinates to a vector generator that converts the digital coordinate values to analog voltage for circuits that display an electron beam hitting on the CRT's phosphor coating. Therefore the beam is deflected from endpoint to endpoint, as dictated by the arbitrary order of the commands in the display list, hence the name *Random Scan Display*. Since the light output of the phosphor decays in tens or at most hundreds of microseconds, the DPU must cycle through the display list to refresh the image around 50 times per second to avoid flicker. A portion of the system memory where the display list is resided is called a *refresh buffer.* This display technology is used with monochromatic CRTs, or beam-penetration color CRTs.

Advantages:

1. It can produce a smooth output primitive with higher resolution unlike the raster display technology.
2. It is better than raster display for real time dynamics such as animation.
3. For transformation, only the endpoints has to be moved to the new position in vector display, but in raster display it is necessary to move those endpoints, and at the same time all the pixels between the endpoints must be scan-converted using appropriate algorithm, no prior information on pixels can be reused.

Disadvantage:

1. A vector display cannot fill areas with patterns, and manipulate bits.
2. Time required for refreshing an image depends upon its complexity (more the lines, longer the time), the flicker may therefore appear as the complexity of the image increases. The fastest vector display can draw about 100,000 short vectors in a refresh cycle without flickering.

Comparison of Random and Raster Displays

Random	***Raster***
1. Restricted to engineering, line drawing applications.	1. Stores intensity information for each screen point, well suited for displaying shading and color areas.
2. Doesn't use interlacing.	2. Use interlacing.
3. Higher resolution.	3. Lower resolution.
4. More expensive.	4. Less expensive.
5. Uses monochrome or beam penetration type.	5. Uses monochrome or shadow mask type.
6. Image is displayed by steering the beam along the vectors.	6. Image is displayed by scanning the whole display area.
7. Editing is easy.	7. Editing is difficult.
8. Solid fill/pattern fill is difficult.	8. Solid fill/pattern fill is easy.
9. Refresh rate depends directly on picture complexity.	9. Refresh rate is independent of picture complexity.

Display device

1. Fluorescence / Phosphorescence

When electron beam strikes phosphor coated screen, the individual electron is moving with kinetic energy that is proportional to the acceleration voltage. Some of this energy is dissipated as heat and the rest of the energy is transferred to the electrons of phosphor atom, making them jump to higher quantum energy levels. In returning to their previous quantum levels the excited electrons give up their extra energy in the form of light, predicated by quantum theory. Any given phosphor has several quantum levels to which

electrons can be excited each corresponding to a color associated with return to an unexcited state. Further, the electrons at some levels are less stable and turn in an unexcited state more rapidly than others. A phosphor's fluorescence is the light emitted as these very unstable electrons lose their excess energy while the phosphor is being struck by electrons. Phosphorescence is the light given off by the return of the relatively more unstable excited electrons to their unexcited state once the electron beam excitation is removed. Since fluorescence usually last just a fraction of a microsecond, most of the light emitted is phosphorescence for a given phosphor.

2. Persistence

A phosphor's persistence is defined as the time from the removal of excitation to the moment when phosphorescence has decayed to 10% of the initial light output. The range of persistence of different phosphors can reach many seconds. The phosphors used for graphics display devices usually have persistence of 10 to 60 microseconds. A phosphor with low persistence is useful for animation; a high persistence phosphor is useful for highly complex, static pictures.

3. Refresh Rate

The refresh rate is the number of times per second the image is redrawn to give a feeling of un-flickering pictures, and it is usually 50 per second. As the refresh rate decreases flicker develops because the eye can no longer integrate the individual light impulses coming from a pixel. The refresh rate above which a picture stops flickering and fuse into a steady image is called critical fusion frequency (CFF).

The factors affecting the CFF are:

- Persistence: longer the persistence the lower the CFF, but the relation between the CFF and persistence is non linear.
- Image intensity: increasing the image intensity increase the CFF with nonlinear relationship.
- Ambient room light: Decreasing the ambient room light increase the CFF with nonlinear relationship.
- Wavelengths of emitted light

- Observer

4. Horizontal scan rate:

The horizontal scan rate is the number of scan lines per second. The rate is approximately the product of the refresh rate and the number of scan lines.

5. Resolution

Resolution is defined as the maximum number of points that can be displayed horizontally and vertically without overlap on display device. Factors affecting the resolution are follows.

- Spot profile: the spot intensity has a Gaussian distribution as depicted in fig a. So two adjacent spots on the display device appear distinct as long as their separation (D_2) is greater than the diameter of the spot (D_1) at which each spot has an intensity of about 60% of that at the center of the spot as shown in fig b

Fig. b

- Intensity: as the intensity of the electron beam increases, the spot size on display tends to increase because of spreading of energy beyond the point of bombardment. This phenomenon is also known as blooming. Consequently the resolution decreases.

Thus it is noted that resolution is not necessarily a constant, and it is not necessarily equal to the resolution of a pixmap which is allocated in buffer memory.

Cathode Ray Tube (CRT)

1. Monochromatic CRT
2. Color CRT
    2.1. Beam penetration
    2.2. Shadow mask
        2.2.1. Delta- Delta shadow mask
        2.2.2. Precision in line CRT

1. Monochromatic CRT

The electron gun emits a stream of electrons that is accelerated towards the phosphor coated screen by a high positive voltage applied inner side of the tube near the screen. The electrons are forced into a narrow beam by the focusing mechanism and directed towards a particular point on screen by the deflection mechanism. The mechanism may be electrostatic or magnetic. When electron hit the screen the phosphor emits visible light and then the phosphor's light output decays exponentially with time. The entire picture must be refreshed (redrawn) many times per-second so that the viewer sees an un-flickering pictures.

The stream of electrons from the heated cathode is accelerated towards the phosphor coated screen by a high voltage 15,000 to 20,000 volts. The control grid voltage determines how many electrons are actually in the electron beam. More negative the control grid voltage, fewer the electrons that pass through the grid. Since the light output of the phosphor depends upon the number of electrons in the beam the brightness of the screen is therefore controlled by varying the grid voltage.

Since electrons in electron-beam repel to each other and tends to diverge. The focusing mechanism employs an electron lens (electrostatic or magnetic) to concentrate the electrons in thin beam, and converge to thin small point when it hits phosphor coating. The cross-sectional electron density of the beam is Gaussian (normal) and the intensity spot on the phosphor has the same distribution as shown in fig. typically spots size of high resolution monochrome CRT is 0.005 inches.

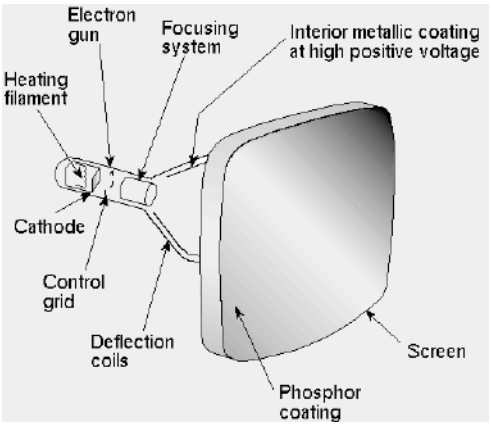


Fig cross-sectional view of monochromatic CRT

2. Color CRT

2.1. Beam penetration

There are two layers namely red and green in the display screen. A display color depends upon how far the electron beam penetrates into phosphor layer. Slow the electron excites only outer red layer and give red color as display while the fast electron excites inner green layers ad give green as display. And electron beam with intermediate speeds penetrate the middle of the screen and hence give the combination of red and green color. i.e color of the display is controlled by acceleration voltage. The quality of the display is not so good.

2.2. Shadow Mask

The inner side of the viewing surface of a color CRT consists of closely spaced groups of red, green and blue phosphor dots. Each group is called a triad. A thin metal plate perforated with many small holes is mounted close to the inner side of the viewing surface. This plate is called the shadow-mask. The shadow mask is mounted in such a way that each hole is correctly aligned with a triad. In color CRT, there are three electron guns one for each dot in triads. The electron beam from each gun therefore hits only the corresponding dot of a triad as the three electron beams deflects. A triad is so small that light emanating from the individual dots is perceived by the viewer as a mixture of the three colors. Thus, a wide range of color can be produced by each triad, depending on how strongly each individual phosphor dot is a triad is excited. There are two types of color CRTs.

2.2.1. A Delta-Delta CRT

A triad has a triangular (delta) pattern as are three electron guns. The beam of electron are deflected and focused onto the shadow mask, which contains the series of holes. When electron hit phosphor they activate a dot triangle. The phosphor dots triangle are arranged such that each electron beam activate only its corresponding color dots.

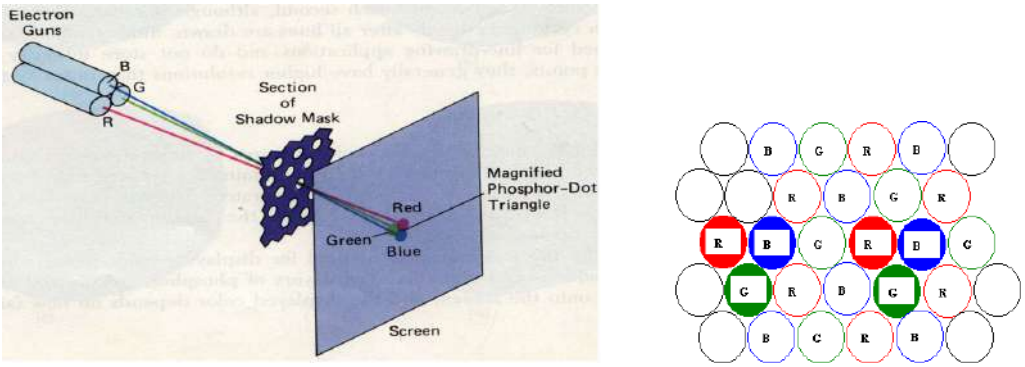


Fig Showing the parts of Delta Delta CRT

Drawback:

The drawback of this CRT is that a high precision display is very difficult to achieve because of technical difficulties involved in the alignment of shadow mask holes and the triads on one to one basis.

2.2.2. Precision in line

A triad has an *in-line pattern* as are three electron guns. The introduction of this type of CRT has eliminated the drawback of delta-delta CRT. But a slight reduction of image sharpness at the edge of the tubes has been noticed. Normally 1,000 scan line can be achieved.

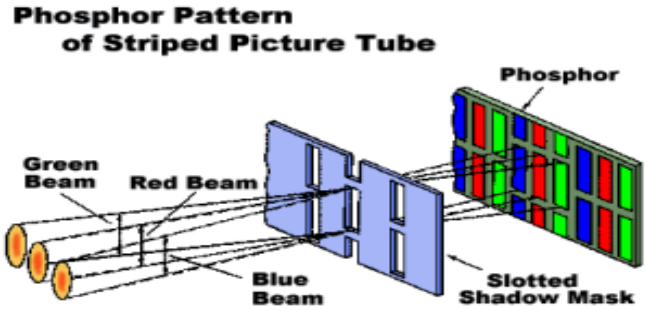


Fig: *Precision in line*

The necessity of triad has reduced the resolution of a color CRT. The distance between the centers of adjacent triads is called a pitch. In very high resolution tubes pitch measures 0.21

mm (0.61 mm for home TV tubes). The diameter of each electron beam is set at 1.75 times the pitch.

For example:

A color CRT (15.5 * 11.6) inches has pitch of 0.01 inches.

Then beam diameter = 0.01 * 1.75 =0.018 inches

Resolution per inch = 1/0.018 =55 lines

Hence resolution achievable for given CRT is 15.5 * 55 = 850 by 11.6 * 55 = 638 lines

Therefore the resolution of a CRT can be increased by decreasing the pitch. But small pitch CRT is difficult to manufacture because it is difficult to set small triads and the shadow mask is more fragile owing to too many holes on it. Besides the shadow mask is more likely to warp from heating by the electrons.

The shadow mask of color CRT also decreases the brightness because only 20% of electrons in the beam hit the phosphor and rest hits the shadow mask.

Flat Panel Display:

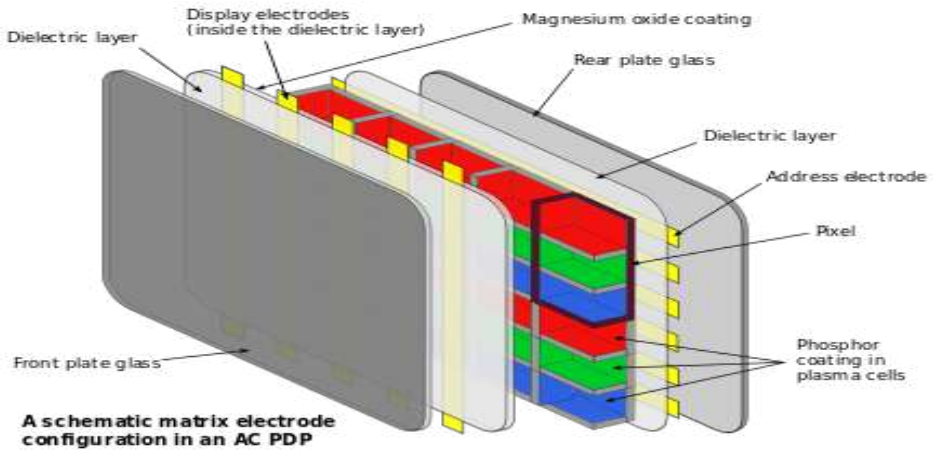
Flat Panel means reduction of volume, weight and power requirements compared to a CRT. Current applications are calculators, laptops computers, etc. It is classified into two types:

a) **Emissive Displays:** These are the device that converts electrical energy into light. E.g.: plasma panel, electro luminescent display, etc.
b) **Non-Emissive Display:** It uses optical effects to converts sunlight or other light sources into graphical patterns. E.g.: LCD

Plasma Panel Display (Gas-discharge display):

It consists of matrix of cells in a glass envelope. Each cell is filled with gas usually neon. In broader sense, region between two glass plates is filled with a mixture of gases such as neon, xenon and other inert gases. A series of vertical conducting ribbons is placed on one glass panel and a set of horizontal ribbons is built into other gas panel. Firing voltages applied to a pair of horizontal and vertical conductor causes gas at intersection of the two conductors to break down into glowing plasma of electrons and ions. By controlling the amount of voltage applied at various points on grid, each point acts as a pixel (intersection of conductors) to display an image. Picture definition is stored in a refresh buffer and firing voltages are applied to refresh pixel positions 60 times per second.

The cells in a glass envelop are luminous when they are electrified through "electrodes". With a sufficiently high voltage, some of the atoms in the gas of a cell lose electrons and become ionized creating electrically conducting plasma of atoms, free electrons, and ions. This disassociated gas is called as plasma. The collisions of the flowing electrons in the plasma with the inert gas atoms lead to light emission; such light-emitting plasmas are known as glow discharges. To turn on a bulb, system adjusts voltages on the corresponding lines. Once a glow discharge starts, it can be maintained by applying a low-level voltage between all the horizontal and vertical electrodes–even after the ionizing voltage is removed. When the electrons recombined energy is released in the form of protons, then the gas glows with orange-red color. To turn on a bulb, system adjusts voltages on the corresponding lines. Once the glow starts, a lower voltage is applied to sustain it. To turn off a bulb, the system momentarily decreases the voltage on the appropriate line than the sustaining voltage.



A schematic matrix electrode configuration in an AC PDP

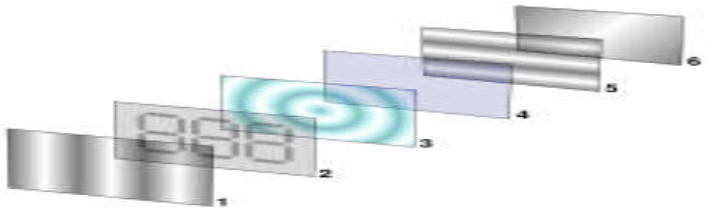
Electro Luminescent Display:

It consists of same grid line structure as used in plasma display. Between front and back panels is a thin layer of electro luminescent material such as zinc Sulphide doped with manganese that emits light when a light electric field is applied. Electrical energy is

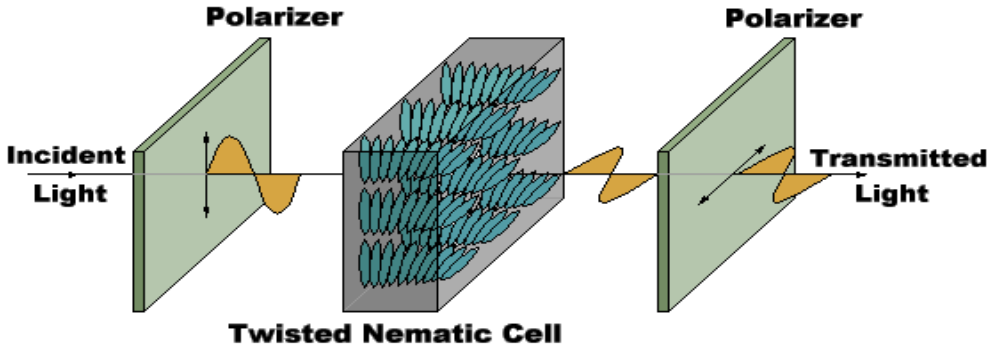
absorbed by a manganese atom which then release energy as a spot of light similar to the glowing plasma effect in the plasma panel.

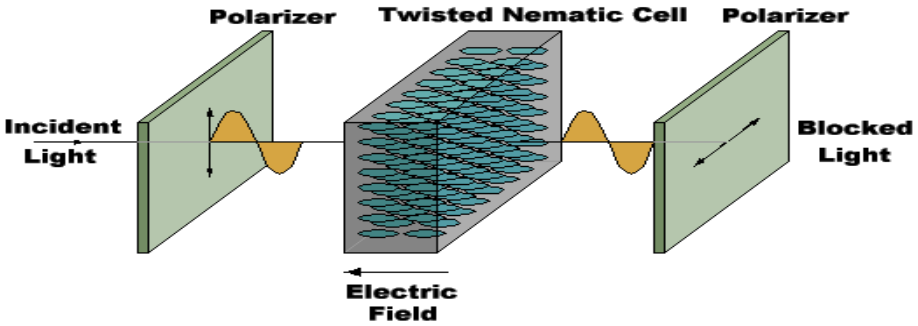
LCD (Liquid Crystal Display):

Liquid Crystal has a crystalline arrangement of molecules, and they get flow like liquids. Two glass plated sandwiched the liquid crystal material to present information on a screen. Rows of horizontal transparent conductors are built into one glass plate and columns of vertical transparent conductors are put into the other plate. The intersection of two conductors defines the pixel position. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is then reflected back to the viewer. This type of device is called passive matrix LCD and when we use transistors at each pixel position, it is called as active matrix displays. Depending upon how much they twist, some light waves are passed through while other light waves are blocked. This creates the variety of color that appears on the screen. LCD monitors produce color using either passive-matrix or active-matrix technology. Active matrix display, also known as a TFT (thin-film transistors) display, uses separate transistors to apply changes to each liquid crystal cell and this display high quality color that is viewable from all angles.



1. Polarizing filter film with a vertical axis to polarize light as it enters.
2. Glass substrate with electrodes with vertical ridges.
3. Twisted nematic liquid crystal.
4. Glass substrate with common electrode film with horizontal ridges to line up with the horizontal filter.
5. Polarizing filter film with a horizontal axis to block/pass light.
6. Reflective surface to send light back to viewer.





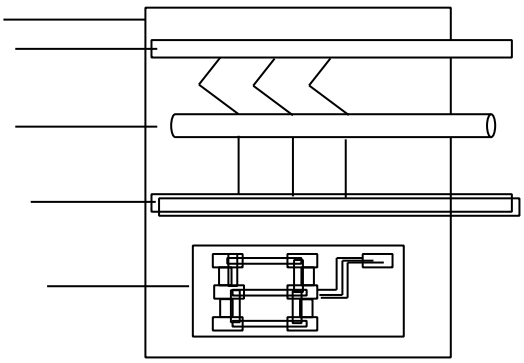
Voice System

The voice system consists of speech recognizer which analyzes the sound of each person. The voice system input can be used to initiate graphics operations or to enter data. These systems operate by matching an input against a predefined dictionary of words and phrases. The microphone is designed to minimize input of other background sounds. It must consist of dictionary of words (frequency pattern) spoken words are converted into frequency pattern.

Scanners

- Converts any printed image of an object into electronic form by shinning light onto the image and sensing the intensity of light's reflection at any point.
- Color scanners use filters to separate components of color into primary additive colors (red, green, blue) at each point.

- Red, Green, Blue are primary additive colors because they can be combined to create any other color.
- Image scanners translate printed images into electronic format that can be stored into a computer memory.
- Software is then used to manipulate the scanned electronic image.
- Images are enhanced or manipulated by graphics programs like Adode.



Optical Character Recognition (OCR)

- For text document, we use Optical Character Recognition software to translate image into text that is editable.
- When a scanner first created an image from a page, the image is stored to computer's memory as bitmap.
- A bitmap is a grid of dots, each dot represented by one or more bits.
- OCR software translates the array of dots into text that the computer can interpret as number and letters by looking at each character and trying to match the character with its own assumption about how the image should look like.

3.6 Data structure concepts and CAD packages

Data Structure

Data structure is representation of the logical relationship existing between individual elements of data. In other words, a data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.

Data may be a single or it may be a set of values. Whether it is a single value or a group of values to be processed must be organized in a particular fashion. This organization leads to structuring of data.

Data structure mainly specifies these four things.

- Organizing of data
- Accessing methods
- Degree of associativity
- Processing alternatives for information.

Data structures are the building blocks of a program.

- The data structure should be rich enough to reflect the relationship between data.
- The structure should be simple so that we can process data effectively whenever required.

Algorithm + Data structure = Program

Classification of data structure

- Primitive data structure

  Primitive data structures are directly operated upon by machine instructions. eg. Integer, floating-point numbers, character constants, string constants, pointers etc.

- Non primitive data structure

  These are more sophisticated data structure. These are derived from primitive data structures. The non-primitive data structures emphasize on structuring of a group of homogeneous or heterogeneous data items. Eg . array, list and files

Various operation perform on Data Structure

- CREATE: - this operation includes reserving memory for the program elements. This can be done by declaration statement. The creation of data structure may take place either during compile-time or during run-time. This operation is concern with dynamic memory allocation.
- DELETE:-this operation destroys the memory space allocated for the specified data structure. This operation is concern with de-allocation of memory.
- SELECTION: - It deals with accessing a particular data within a data structure.
- UPDATE:- This implies modifications or updates data in the data-structure.
- SEARCHING: - this operation concern with finding the desired data items from the given list. It may also find the locations of all elements that satisfy certain conditions.
- SORTING: - sorting is the process of arranging all data items in the data-structure either in ascending order or descending order.
- MERGING: - It is a process of merging or combining the data items of two different sorted lists into a single sorted list.

Description of various Data-Structure

ARRAY

Array is defined as a set of finite number of homogeneous elements or data items. It means it contains only one type of data type like integer, character, floating etc..

SYNTAX : **Int a[10];**

Where int specifies data type of elements array stores, “a” is the name of array, and the number specified inside the square brackets is number of elements an array can store. The index of an array begins with 0 and end with (n-1) where n is the maximum size of an array.

STACK

A stack is defined formally as a list (a linear data structure) in which all insertion and deletions are made at one end called the top of the stack(TOS). The fundamental operations which are possible on a stack are:

- push - insertion

- pop - deletion

The most recently pushed element can be checked prior to performing a pop operation. A stack is based on the last in first out algorithm (LIFO) in which insertion and deletion operations are performed at one end of the stack. The most accessible information in a stack is at the top of the stack and least accessible information is at the bottom of the stack. If someone wants to delete on item from an empty stack, it causes underflow and if someone wants to add an element to the stack (which is already full) then it is the case of overflow.

Implementation of Stack:

Stack can be implemented in two ways:

1. Pointer (Linked list)
2. Array

Push operation:

Step 1: Check for the overflow

If TOS ≥ = size

Output: “stack overflow and exit”.

Step 2: Increment the pointer value by one

TOS = TOS +1

Step 3: Perform insertion

S[TOS] = value

Step 4: Exit

POP operation:

Step 1: Check for the underflow

If TOS = 0

Output: “Stack underflow and exit”

Step 2: Decrement the pointer value by one

Value = S[TOS]

TOS = TOS-1

Step 3: Return the former information of the stack

Return [value]

Step 4: Exit.

Queue:

This is a non primitive linear data structure used to represent a linear list and permits deletion to be performed at one end of the list called front of a list and insertion on the other end of the list called rear of a list. The information is such a list is processed in the same order as it was received, i.e on First In First Out basis. (FIFO). A queue has two pointes; front and rear, pointing to the front and rear elements of the queue respectively. Consider a queue consisting of 'n' elements and an element value which we have to insert into the queue. The value NULL (0) of front pointer employees an empty queue.

Circular queue:

Suppose we have an array Q that contains n element in which Q1 comes after Qn in the array. When this technique is used to construct a queue then the queue is called the circular queue. In other words we can say that a queue is called circular when the last room comes just before the first room. Fig given below show the circular queue.

Linked List

Information part may consist of one or more than one fields. In other words a linked list consist of a series of a structure contains one or more then one contiguous information fields and a pointer to a structure containing its successor. The last node of the list contains NULL ('\0'). The pointer contains the address of the location where the next information is stored. A linked list also contains a list

pointer variable called start, which contains the address of the first node is the list. Hence , there is an arrow drawn from start to the node in the linked list. If there is no node in the list, the is called NULL list or EMPTY list. The different operations performed on the linked list are:

(1) Insertion:

a. Inset a node at the beginning of the list.

b. Inset a node at the end of the list.

c. Inset a node between the nodes.

(2) Deletion

a. Delete a node at the beginning of the list.

b. Delete a node at the end of the list.

c. Delete a node between the nodes.

(3) Traversing

Travelling from one node to another node in the list.

Doubly linked list:

A singly linked list is so named because each list element contain a pointer to the next element . In that list traveling is possible only in one direction. Some times it is required to transverse the list in either direction forward or backward. This involves the performance and efficiency of algorithms. So traversing of linked list in both the directions requires nodes having the links as given in figure below:

The links are used to denote the predecessor and successor of a node. The link denoting the predecessor of a node is called the left link and that denoting its successor is right link. A list with this type of arrangement is called Doubly Linked List as shown in the figure above. Now it is possible to define a doubly linked as a collection of nodes, each nodes having three fields.

- pointer to previous node (pointer to predecessor)
- Information field
- Pointer to next node (pointer to successor)

Graphs:

graphs G is defined as set of two tuples G = (V,E).

Where ,

V represents set of vertices of G and

E represents the set of edges of G.

There exists a mapping from the set of edges to a set of pairs of element of V. Fog eg. given below shows the different types of graphs.

Chapter 3:

3.1: line drawing

1. DDA (Digital Differential Analyzer) Algorithm

The basis of the DDA (Digital Differential Analyzer) method is to take unit steps along one of the coordinate assume **x co-ordinate** and compute the corresponding values along the other coordinate lets say **Y co-ordinate**. The unit steps Lets say **m** are always along the coordinate of greatest change, e.g. if we have **dx = 11** and **dy = 6**, then we would take unit steps **along x** and compute the steps **m along y**.

```
Slope = m =(dy/dx)

If (m <= 1.0) then                  //Means we have dx > dy
Let x_step = 1
{dx = 1, dy = 0 or 1}
Else {m > 1.0}                            //We have dx < dy
Let y_step = 1
{dy = 1, x_step = 0 or 1 }
```

- DDA stands for digital differential analyzer.
- The differential equation for a line is: m = dy / dx

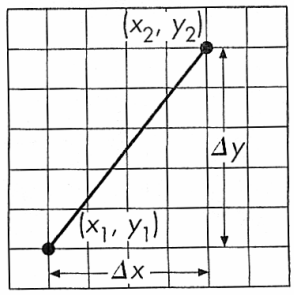


Figure 8.40 Line segment in window coordinates.

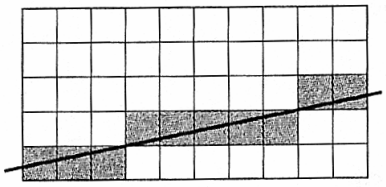


Figure 8.41 Pixels generated by DDA algorithm.

Advantages & disadvantages of DDA:-

- Faster than the direct use of the line equation and it does not do any floating point multiplication.
- Floating point Addition is still needed.
- Precision loss because of rounding off.
- Pixels drift farther apart if line is relatively larger.

#Program for plotting a line using DDA line drawing algorithm

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void DDALine(int, int, int, int);
int gd=DETECT,gm;
```

```
void main(){
  int x1,x2,y1,y2;
  initgraph(&gm,&gd,"c:\\tc\\bgi");
  printf("Enter the first point of a line");
  scanf("%d%d",&x1,&y1);
  printf("Enter the End point of a line");
  scanf("%d%d",&x2,&y2);
  DDALine(x1, y1, x2, y2);
  getch();
  closegraph();
}


void DDALine( int x1, int y1,int  x2,int  y2){
  int dx, dy, steps;
  float x_inc, y_inc, x, y;
  dx = x2 - x1;
  dy = y2 - y1;
  if (abs(dx) > abs(dy))
     steps = abs(dx);                                    //steps is larger of dx, dy
  else
     steps = abs(dy);
  x_inc = dx/steps;
  y_inc = dy/steps;
//either x_inc or y_inc = 1.0, the other is the slope
 x=x1;
 y=y1;
 putxy(round(x), round(y));
 for( i = 1 to steps) {
      x = x + x_inc;
      y = y + y_inc;
      putxy(round(x), round(y));
```

}

}//End DDALine Algorithm Function

2. Bresenham Line Drawing Algorithm

- Improve upon DDA algorithm to use integer arithmetic only.
- The Bresenham algorithm is incremental scan conversion algorithm
- The big advantage of this algorithm is that it uses only integer calculations
- Applicable to circle drawing too.

Move across the *x* axis in unit intervals and at each step choose between two different *y* coordinates

For example, from position (2, 3) we have to choose between (3, 3) and (3, 4)

We would like the point that is closer to the original line

At sample position x_k+1 the vertical separations from the mathematical line are labelled d_{upper} and d_{lower}

The *y* coordinate on the mathematical line at x_k+1 is:

$$y = m(x_k + 1) + b$$

So, d_{upper} and d_{lower} are given as follows:

$$d_{lower} = y - y_k$$

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

and:

$$d_{upper} = (y_k + 1) - y$$

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

We can use these to make a simple decision about which pixel is closer to the mathematical line

This simple decision is based on the difference between the two pixel positions:

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

Let's substitute *m* with Δ*y*/Δ*x* where Δ*x* and Δ*y* are the differences between the end-points:

$$\begin{aligned}\Delta x(d_{lower} - d_{upper}) &= \Delta x(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + 2\Delta y + \Delta x(2b - 1) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c\end{aligned}$$

So, a decision parameter p_k for the *k*th step along a line is given by:

$$\begin{aligned}p_k &= \Delta x(d_{lower} - d_{upper}) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c\end{aligned}$$

The sign of the decision parameter p_k is the same as that of $d_{lower} - d_{upper}$

If p_k is negative ($p_k < 0$), then we choose the lower pixel, otherwise we choose the upper pixel

Remember coordinate changes occur along the *x* axis in unit steps so we can do everything with integer calculations

At step *k*+1 the decision parameter is given as:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

Subtracting p_k from this we get:

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

But, x_{k+1} is the same as x_k+1 so:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

Where $y_{k+1} - y_k$ is either 0 or 1 depending on the sign of p_k

The first decision parameter p0 is evaluated at (x0, y0) is given as:

$$p_0 = 2\Delta y - \Delta x$$

Note: The algorithm and derivation above assumes slopes are less than 1. for other slopes we need to adjust the algorithm slightly

Let's have a go at this

Let's plot the line from (20, 10) to (30, 18)

First off calculate all of the constants:

- Δx: 10
- Δy: 8
- $2\Delta y$: 16
- $2\Delta y - 2\Delta x$: -4

Calculate the initial decision parameter p_0:

- $p0 = 2\Delta y - \Delta x = 6$

- Go through the steps of the Bresenham line drawing algorithm for a line going from (21,12) to (29,16)

The Bresenham line algorithm has the following advantages:

- An fast incremental algorithm
- Uses only integer calculations

Comparing this to the DDA algorithm, DDA has the following problems:

- Accumulation of round-off errors can make the pixelated line drift away from what was intended
- The rounding operations and floating point arithmetic involved are time consuming

```
/*Program for plotting a line using Bresenham's Line drawing Algorithm */
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main(){
clrscr();
int g=DETECT,d;
float x,y,dy,dx,p,dy2;
int k,x1,x2,y1,y2;
initgraph(&g,&d,"c:\\tc\\bgi");
cout<<"\n Enter the first points X1 and Y1\t";
```

```
cin>>x1>>y1;
cout<<"\n Enter the value for X2 and Y2\t";
cin >>x2>>y2;
cleardevice();
putpixel(x,y,RED);
dy=y2-y1;
dx=x2-x1;
p=dy2-dx;
for(k=0;k<=dx;k++){
    if(p>0)      {
        putpixel(x,y,RED);
        p=p+dy2;
        x++;
        }
    else         {
        putpixel(x,y,RED);
        p=p+dy2-2*dx;
        x++;
        y++;
        }
 }
getch();
closegraph();
}
```

Circle

A circle is a set of points that are all at a given distance *r* from a center position. If the centre of a circle at origin (0,0). Then the equation for a circle is:

$$x^2 \square y^2 \square r^2$$

Where

r is the radius of the circle

So, we can write a simple circle drawing algorithm by solving the equation for *y* at unit *x* intervals using:

$$y \square \pm\sqrt{r^2 \square x^2}$$

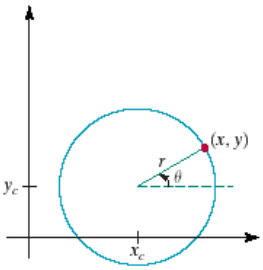
If the centre of a circle is at any point (*x c* , *y c*).

For any circle point (*x* , *y*), this distance relationship is expressed by the Pythagorean theorem in Cartesian coordinates as

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

We could use this equation to calculate the position of points on a circle circumference by stepping along the *x* axis in unit steps from *x c* - *r* to *x c* + *r* and calculating the corresponding *y* values at each position as

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



But this is not the best method for generating a circle.

One problem with this approach is that Firstly it involves considerable computation at each step. Moreover, the spacing between plotted pixel positions is not uniform.

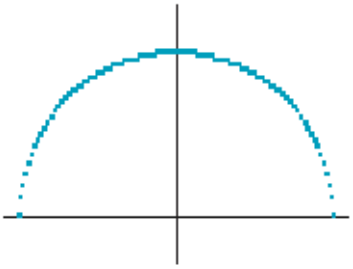
We could adjust the spacing by interchanging *x* and *y*. But this simply increases the computation and processing required by the algorithm.

Secondly the calculations are not very efficient

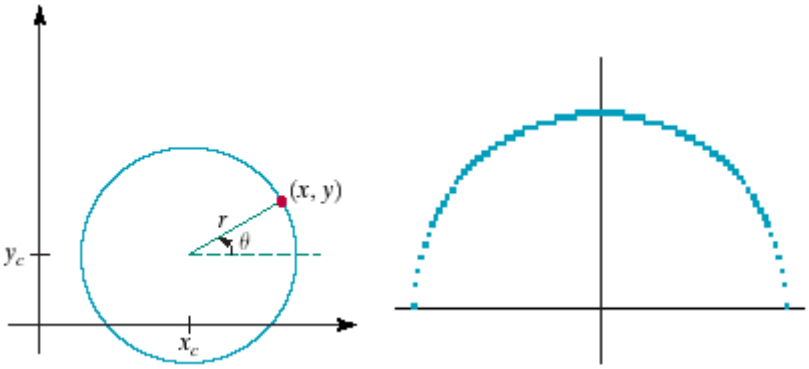
- The square (multiply) operations
- The square root operation – try really hard to avoid these!

We need a more efficient, more accurate solution

Upper half of a circle plotted with (*x c* , *y c*) = (0, 0).



Another way to eliminate the unequal spacing shown in Fig. is to calculate points along the circular boundary using polar coordinates *r* and Theta.



Expressing the circle equation in parametric polar form yields the pair of equations

$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

When a display is generated with these equations using a fixed angular step size, a circle is plotted with equally spaced points along the circumference.

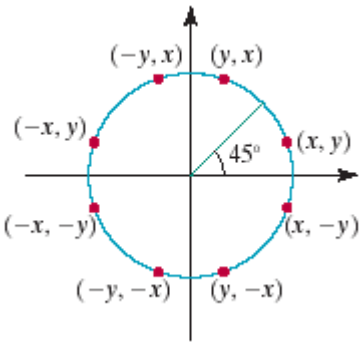
To reduce calculations, we can use a large angular separation between points along the circumference and connect the points with straight-line segments to approximate the circular path.

For a more continuous boundary on a raster display, set the angular step size at 1/*r*. This plots pixel positions that are approximately one unit apart. Although polar coordinates provide equal point spacing, the trigonometric calculations are still time consuming.

- Reduce computations by considering the symmetry of circles.
- The shape of the circle is similar in each quadrant. Determine the curve positions in the first quadrant; generate the circle section in the second quadrant of the *xy* plane by noting that the two circle sections are symmetric with respect to the *y* axis.
- And circle sections in the third and fourth quadrants can be obtained from sections in the first and second quadrants by considering symmetry about the *x* axis. We can take this one step further and note that there is also symmetry between octants. Circle sections in adjacent octants within one quadrant are symmetric with respect to the 45 line dividing the two octants.

Mid Point Circle Algorithm

The first thing we can notice to make our circle drawing algorithm more efficient is that circle centred at (*0, 0*) have *eight-way symmetry*



Similarly to the case with lines, there is an incremental algorithm for drawing circles – the *mid-point circle algorithm*

In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points

Let's re-jig the equation of the circle slightly to give us:

$$f_{circ}(x, y) = x^2 + y^2 - r^2$$

The equation evaluates as follows:

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

By evaluating this function at the midpoint between the candidate pixels we can make our decision

Assuming we have just plotted the pixel at (x_k, y_k) so we need to choose between (x_k+1, y_k) and (x_k+1, y_k-1)

Our decision variable can be defined as:

$$\begin{aligned} p_k &= f_{circ}(x_k + 1, y_k - \tfrac{1}{2}) \\ &= (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2 \end{aligned}$$

If $p_k < 0$ the midpoint is inside the circle and and the pixel at y_k is closer to the circle

Otherwise the midpoint is outside and y_k-1 is closer

To ensure things are as efficient as possible we can do all of our calculations incrementally

First consider:

$$\begin{aligned} p_{k+1} &= f_{circ}[x_{k+1} + 1, y_{k+1} - \tfrac{1}{2}] \\ &= [(x_k + 1) + 1]^2 + [y_{k+1} - \tfrac{1}{2}]^2 - r^2 \end{aligned}$$

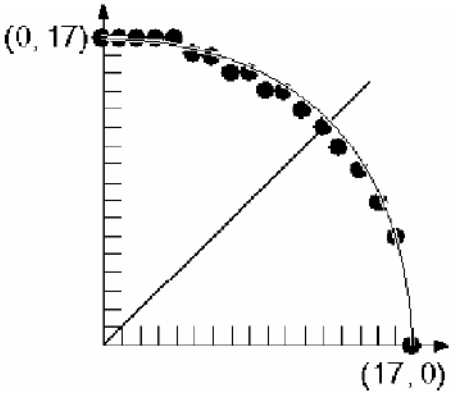
or:

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

where y_{k+1} is either y_k or y_k-1 depending on the sign of p_k

The idea in this approach is to test the halfway position between two pixels to determine if this midpoint is inside or outside the circle boundary.

This method is more easily applied to other conics; and for an integer circle radius, the midpoint approach generates the same pixel positions as the Bresenham circle algorithm.




```
/* Program for plotting a circle using mid point circle drawing algorithm */
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdio.h>
void main(){
 float xc,yc,x1,y1,pk,pk1,xk,yk,xk1,yk1,r;
 printf("Enter the centre of the circle");
 scanf("%f%f",&xc,&yc);
 printf("Enter the radius of the circle");
 scanf("%f",&r);;
 int driver,mode;
 driver=DETECT;
 initgraph(&driver,&mode,"..\\bgi");
 pk=5/4-r;
 xk=0;
 yk=r;
 while(xk<=yk)  {
```

```
  if(pk<0)    {
          xk1=xk+1;
          yk1=yk;
          pk=pk+2*xk1+1;
   }
  else    {
          xk1=xk+1;
          yk1=yk-1;
          pk=pk+2*xk1+1-2*yk1;
   }
  putpixel(xc+xk,yc+yk,1);
  putpixel(xc-xk,yc+yk,1);
  putpixel(xc+xk,yc-yk,1);
  putpixel(xc-xk,yc-yk,1);
  putpixel(xc+yk,yc+xk,1);
  putpixel(xc-yk,yc+xk,1);
  putpixel(xc+yk,yc-xk,1);
  putpixel(xc-yk,yc-xk,1);
  xk=xk1;
  yk=yk1;
 }
 getch();
 closegraph();
}
```

Ellipse

```
// program to draw the ellipse by mid point
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
```

```
int round(float);
void main(void){
 float rx2,ry2,x,y,xk,yk,p;
 int xc,yc,rx,ry;
  int m,d=DETECT;
 initgraph(&d,&m,"..\\bgi");
 printf("\n Enter the center coordinate (x,y) for ellipse:");
 scanf("%d%d",&x,&y);
 printf("\n Enter the radius to x and y axis:");
 scanf("%d%d",&rx,&ry);
 cleardevice();
 xk=0;
 yk=ry;
 x=rx*rx;
 y=ry*ry;
 p=(y-x*ry+(0.25)*x);
 while(2*y*xk<2*x*yk)  {
  xk=xk+1;
  if(p<0)    {
          yk=yk;
          p=(p+2*y*xk+y);
   }
  else    {
          yk=yk-1;
          p=(p+2*y*xk-2*x*yk+y);
   }
  putpixel(xc+xk,yc+yk,1);
  putpixel(xc-xk,yc+yk,1);
  putpixel(xc+xk,yc-yk,1);
  putpixel(xc-xk,yc-yk,1);
 }
p=(y*(xk+0.5)*(xk+0.5)+x*(yk-1)*(yk-1)-x*y);
```

```
 while(xk<rx)  {
  yk=yk-1;
  if(p>0)    {
          xk=xk;
          p=p+x-2*x*yk;
   }
  else    {
          xk=xk+1;
          p=p+2*y*xk-2*x*yk+x;
   }
  putpixel(xc+xk,yc+yk,1);
  putpixel(xc-xk,yc+yk,1);
  putpixel(xc+xk,yc-yk,1);
  putpixel(xc-xk,yc-yk,1);
 }
 getch();
}

int round(float value){
 int i;
 if(value-int(value)>=0.5)
  i=int(value)+1;
 else
  i=value;
  return i;
}
```

Clipping

Clipping refers to the removal of part of a scene. Internal clipping removes parts of a picture outside a given region; external clipping removes parts inside a region. We'll explore internal clipping, but external clipping can almost always be accomplished as a by-product.

A line clipping algorithms takes as input two endpoints of line segment and returns one (or more) line segments. A polygon clipper takes as input the vertices of a polygon and returns one (or more) polygons. There are several clipping algorithms. We'll study the Cohen-Sutherland line clipping algorithm to learn some basic concepts.

Cohen-Sutherland Line Clipping

The Cohen-Sutherland algorithm clips a line to an upright rectangular window. It is an application of *triage*, or makes the simple case fast. The algorithm extended window boundaries to define 9 regions:

Top-left, top-center, top-right, center-left, center, center-right, bottom-left, bottom-center, and bottom-right.

See figure 1 below. These 9 regions can be uniquely identified using a 4 bit code, often called an *outcode or Region code*. We'll use the order: left, right, bottom, top (LRBT) for these four bits. In particular, for each point $p = (x, y)$

- Left (first) bit is set to 1 when *p* lies to left of window (i.e $x < x_{wmin}$)
- Right (second) bit is set to 1 when *p* lies to right of window (i.e $x > x_{wmax}$)
- Bottom (third) bit is set to 1 when *p* lies below window (i.e $y < y_{wmin}$)
- Top (fourth) bit set is set to 1 when *p* lies above window (i.e $y > y_{wmax}$)

The LRBT (Left, Right, Bottom, and Top) order is somewhat arbitrary, but once an order is chosen we must stick with it. Note that points on the clipping window edge are considered inside (the bits are left at 0).

1001	0001	0101
1000	0000 Clip Window	0100
1010	0010	0110

Figure 1: The nine regions defined by an up-right window and their Region codes.

Given a line segment with end points

$p_0 = (x_0, y_0)$ And $p_1 = (x_1, y_1)$,

Here's the basic flow of the Cohen-Sutherland algorithm:

1. Compute 4-bit outcodes $LRBT_0$ and $LRBT_1$ for each end-point

2. If both outcodes are 0000, the *trivially visible* case, pass end-points to draw routine this occurs when the bitwise OR of outcodes yields 0000.

3. If both outcodes have 1's in the same bit position, the *trivially invisible* case, clip the entire line (pass nothing to the draw routine). This occurs when the bitwise AND of outcodes is not 0000.

4. Otherwise, the *indeterminate* case, - line may be partially visible or not visible. Analytically compute the intersection of the line with the appropriate window edges

Let's explore the indeterminate case more closely. First, one of two end-points must be outside the window, pretend it is $p_0 = (x_0, y_0)$.

1. Read P_1's 4-bit code in order, say left-to-right

2. When a set bit (1) is found, compute intersection point *I* of corresponding window edge with line from p_0 to p_1.

As an example, pretend the right bit is set so we want to compute the intersection with the right clipping window edge, also, pretend we've already done the homogeneous divide, so

the right edge is x=1, and we need to find y. The y value of the intersection is found by substituting x=1 into the line equation (from p_0 to p_1)

$$y - y_0 = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

and solving for y

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(1 - x_1).$$

Other cases are handled similarly.

Now this may not complete the clipping of the line, so we replace p_0 by the intersection point I and repeat Cohen-Sutherland algorithm. (Clearly we can save some state to avoid some computations)

- Define window by

$$\begin{aligned} x &= 0 \quad \text{Left edge} \\ x &= 1 \quad \text{Right edge} \\ y &= 0 \quad \text{Bottom edge} \\ y &= 1 \quad \text{Top edge} \end{aligned}$$

- End-points $p_0 = (1/2, 1/2)$ and $p = (1/4, 3/4)$ both have 4-bit codes 0000. Logical bitwise OR: $0000 \vee 0000 = 0000 \Rightarrow$ line is completely visible - draw line between them.
- End-point $p_2 = (3, 3)$ has 4-bit code 0101 and end-point $p_3 = (-2, 5)$ has 4-bit code 1001. Logical bitwise AND: $0101 \wedge 1001 = 0001 \neq 0000 \Rightarrow$ both end-points to right of window. Therefore line is invisible.
- End-point $p_4 = (3, 3)$ has 4-bit code 0101 and end-point $p_5 = (0, 1/2)$ has 4-bit code 0000.
    - Logical bitwise OR $0101 \vee 0000 = 0101 \Rightarrow$ no information.
    - Logical bitwise AND $0101 \wedge 0000 = 0000 \Rightarrow$ no information.

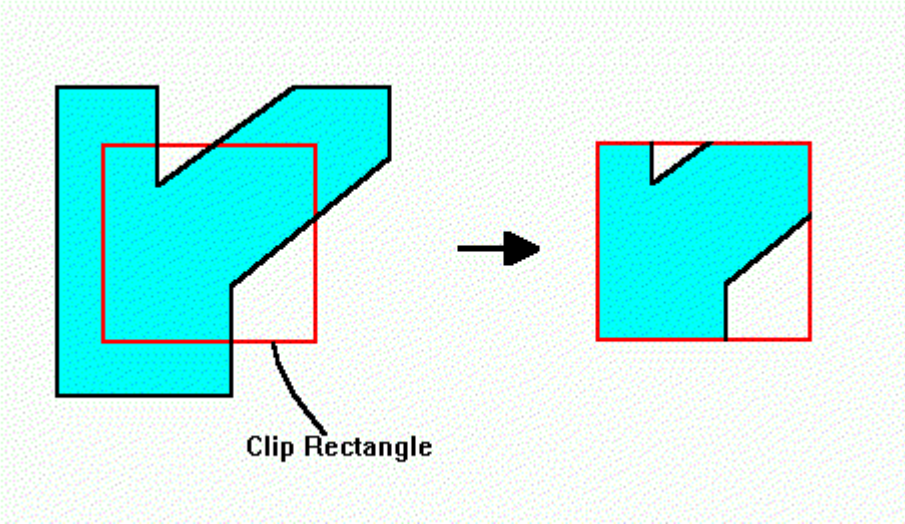
- $p_4 = (3, 3)$ is outside.
- Reading p_4's 4-bit code from left-to-right, ``right'' bit is set.
- Intersection with right edge is at $I = (1, 4/3)$
- I has 4-bit code . . .

The Cohen-Sutherland was one of, if not, the first clipping algorithm to be implemented in hardware. Yet the intersection was not computed analytically, as we have done, but by bisection (binary search) of the line segment.

Clipping Polygons

An algorithm that clips a polygon must deal with many different cases. The case is particularly note worthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of 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. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all these cases.

The following example illustrates a simple case of polygon clipping.



Sutherland and Hodgman's polygon-clipping algorithm uses a divide-and-conquer strategy: It solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clip edge. Four clip edges, each defining one boundary of the clip rectangle, successively clip a polygon against a clip rectangle.

Note the difference between this strategy for a polygon and the Cohen-Sutherland algorithm for clipping a line: The polygon clipper clips against four edges in succession, whereas the line clipper tests the out code to see which edge is crossed, and clips only when necessary.

Steps of Sutherland-Hodgman's polygon-clipping algorithm

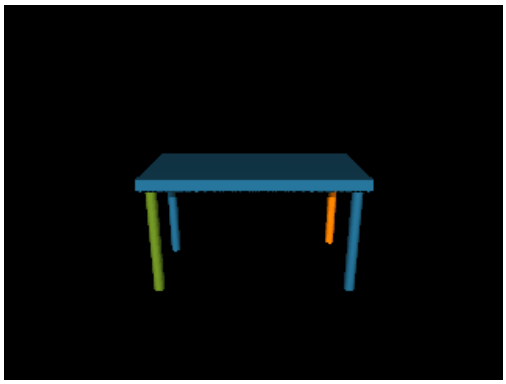
- Polygons can be clipped against each edge of the window one at a time. Windows/edge intersections, if any, are easy to find since the X or Y coordinates are already known.
- Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges.
- Note that the number of vertices usually changes and will often increases.
- We are using the Divide and Conquer approach.

2.3 Window to Viewport Coordinate Transformation

A world-coordinate area selected for display is called a window. An area on a Display device to which a window is mapped is called a viewport. The window defines what is to be viewed; the viewport defines where it is to be displayed.

2D Transformations

Transformations are a fundamental part of computer graphics. Transformations are used to position objects, to shape objects, to change viewing positions, and even to change how something is viewed (e.g. the type of perspective that is used).



In 3D graphics, we must use 3D transformations. However, 3D transformations can be quite confusing so it helps to first start with 2D.

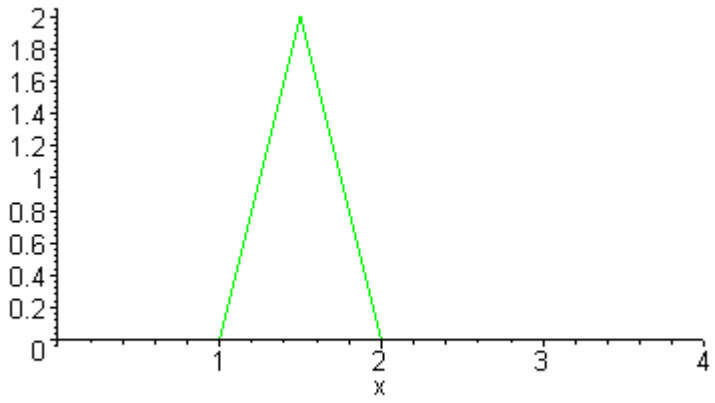
There are 2 main types of transformations that one can perform in 2 dimensions:

1. Basic Transformation
   - Translation
   - Scaling
   - Rotation
2. Composite Transformation
   - shearing
   - Reflection

These basic transformations can also be combined to obtain more complex transformations. In order to make the representation of these complex transformations easier to understand and more efficient, we introduce the idea of homogeneous coordinates.

Representation of Points/Objects

A point **p** in 2D is represented as a pair of numbers: **p**= (x, y) where x is the x-coordinate of the point **p** and y is the y-coordinate of **p** . 2D objects are often represented as a set of points (vertices), $\{p_1,p_2,...,p_n\}$, and an associated set of edges $\{e_1,e_2,...,e_m\}$. An edge is defined as a pair of points $e = \{p_i,p_j\}$. What are the points and edges of the triangle below?



We can also write points in vector/matrix notation as

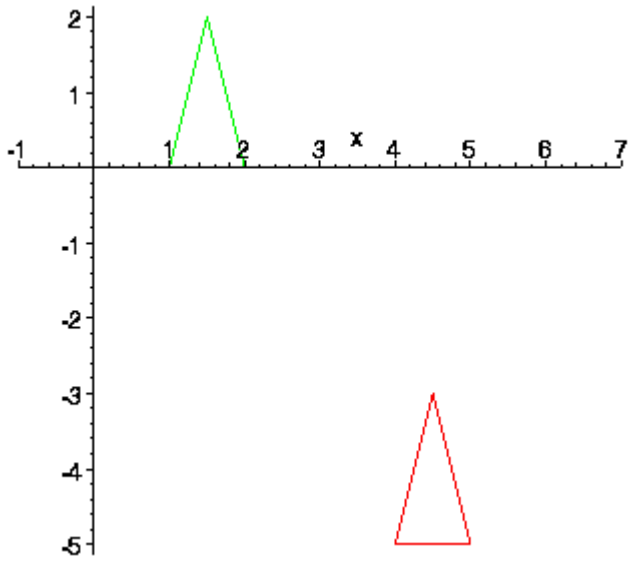
$$p = \begin{bmatrix} x \\ y \end{bmatrix}$$

Translations

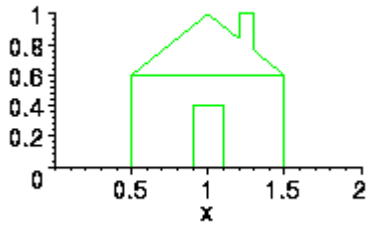
Translation is repositioning the object along the straight line path. Assume you are given a point at (x,y)=(2,1). Where will the point be if you move it 3 units to the right and 1 unit up? Ans: (x',y') = (5,2). How was this obtained? - (x',y') = (x+3,y+1). That is, to move a point by some amount *dx* to the right and *dy up*, you must add *dx* to the x-coordinate and add *dy* to the y-coordinate.

What was the required transformation to move the green triangle to the red triangle? Here the green triangle is represented by 3 points

triangle = { p1=(1,0), p2=(2,0), p3=(1.5,2) }



What are the points and edges in this picture of a house? What is the transformation is required to move this house so that the peak of the roof is at the origin? What is required to move the house as shown in animation?



click here for animation

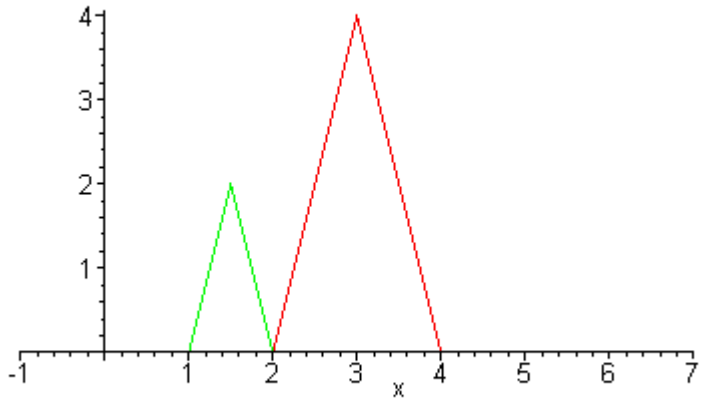
Matrix/Vector Representation of Translations

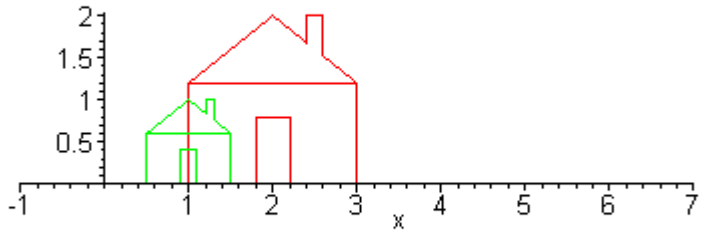
A translation can also be represented by a pair of numbers, $t=(t_x,t_y)$ where t_x is the change in the x-coordinate and t_y is the change in y coordinate. To translate the point p by t, we simply add to obtain the new (translated) point q = p + t.

$$q = p + t = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \end{bmatrix}$$

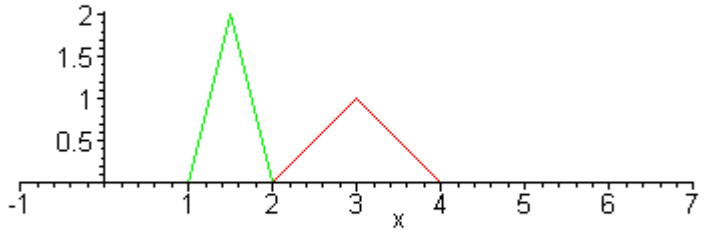
Scaling

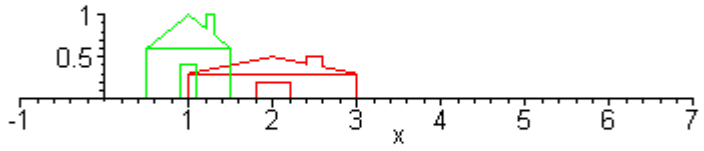
Suppose we want to double the size of a 2-D object. What do we mean by double? Double in size, width only, height only, along some line only? When we talk about scaling we usually mean some amount of scaling along each dimension. That is, we must specify how much to change the size along each dimension. Below we see a triangle and a house that have been doubled in *both* width and height (note, the area is more than doubled).


4
3
2
1
-1
1
2
3
4
5
6
7
x

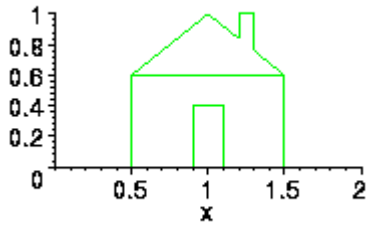

2
1.5
1
0.5
-1
1
2
3
4
5
6
7
x

The scaling for the x dimension does not have to be the same as the y dimension. If these are different, then the object is distorted. What is the scaling in each dimension of the pictures below?





And if we double the size, where is the resulting object? In the pictures above, the scaled object is always shifted to the right. This is because it is scaled with *respect to the origin*. That is, the point at the origin is left fixed. Thus scaling by more than 1 moves the object away from the origin and scaling of less than 1 moves the object toward the origin. This can be seen in the animation below.



click here for animation

This is because of how basic scaling is done. The above objects have been scaled simply by multiplying each of its points by the appropriate scaling factor. For example, the point p=(1.5,2) has been scaled by 2 along x and .5 along y. Thus, the new point is
q = (2*1.5,.5*2) = (1,1).

Matrix/Vector Representation of Translations

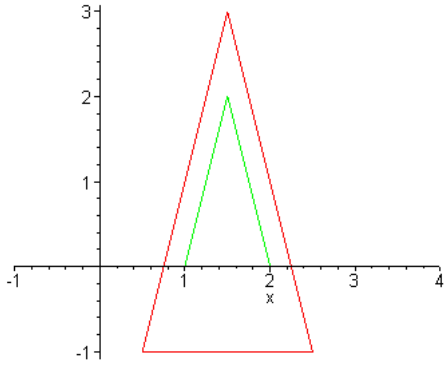
Scaling transformations are represented by matrices. For example, the above scaling of 2 and .5 is represented as a matrix:

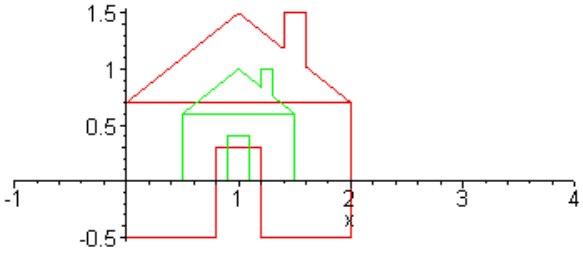
scale matrix: s = $\begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & .5 \end{bmatrix}$

new point: q = s*p = $\begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} sx\ x \\ sy\ y \end{bmatrix}$

Scaling about a Particular Point

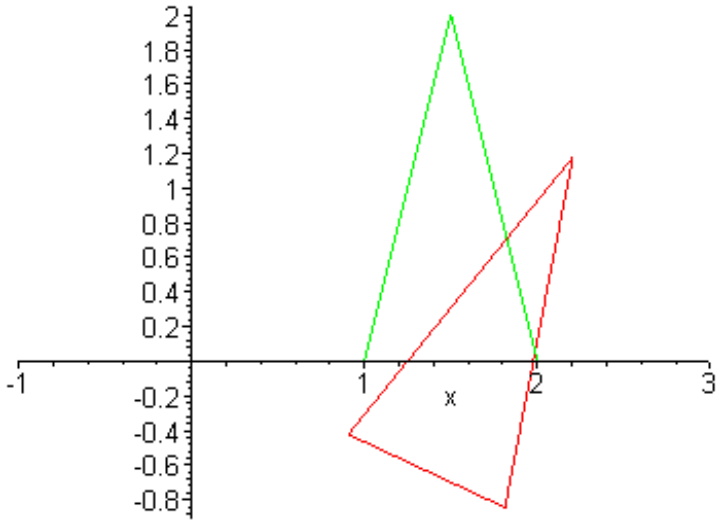
What do we do if we want to scale the objects about their center as show below? *Answer*

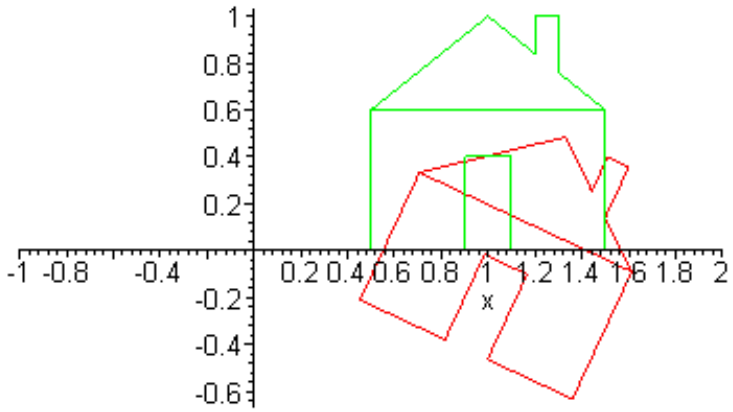




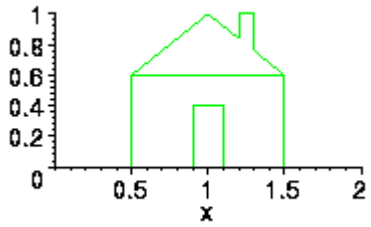
Rotation

Below, we see objects that have been rotate by 25 degrees.





Again, we see that basic rotations are with respect to the origin:

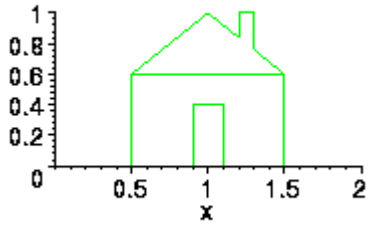


click here for animation

Matrix/Vector Representation of Translations

counterclockwise rotation of ***a*** degrees = $\begin{bmatrix} \cos(a) & \sin(a) \\ -\sin(a) & \cos(a) \end{bmatrix}$

Shear



click here for animation

Matrix/Vector Representation of Translations

shear along x axis = $\begin{bmatrix} 1 & shearx \\ 0 & 1 \end{bmatrix}$

shear along y axis = $\begin{bmatrix} 1 & 0 \\ sheary & 1 \end{bmatrix}$

Combining Transformations

We saw that the basic scaling and rotating transformations are always with respect to the origin. To scale or rotate about a particular point (the fixed point) we must first translate the object so that the fixed point is at the origin. We then perform the scaling or rotation and then

the inverse of the original translation to move the fixed point back to its original position. For example, if we want to scale the triangle by 2 in each direction about the point fp = (1.5,1), we first translate all the points of the triangle by **T = (-1.5,1)**, scale by 2 (**S**) , and then translate back by **-T=(1.5,1)**. Mathematically this looks like

$$\mathbf{q} = \begin{bmatrix} x2 \\ y2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \left(\begin{bmatrix} x1 \\ y1 \end{bmatrix} + \begin{bmatrix} -1.5 \\ -1 \end{bmatrix} \right) + \begin{bmatrix} 1.5 \\ 1 \end{bmatrix}$$

Order Matters!

Notice the order in which these transformations are performed. The first (rightmost) transformation is **T** and the last (leftmost) is **-T**. If you apply these transformations in a different order then you will get very different results. For example, what happens when you first apply **T** followed by **-T** followed by **S**? Here **T** and **-T** cancel each other out and you are simply left with **S**

.Sometimes (but be careful) order does not matter, For example, if you apply multiple 2D rotations, order makes no difference:

$\mathbf{R_1 R_2 = R_2 R_1}$

But this will not necessarily be true in 3D!!

Homogeneous Coordinates

In general, when you want to perform a complex transformation, you usually make it by combining a number of basic transformations. The above equation for **q**, however, is awkward to read because scaling is done by matrix multiplication and translation is done by vector addition. In order to represent all transformations in the same form, computer scientists have devised what are called homogeneous coordinates. Do not try to apply any exotic interpretation to them. They are simply a mathematical trick to make the representation be more consistent and easier to use.

Homogeneous coordinates (HC) add an extra virtual dimension. Thus 2D HC are actually 3D and 3D HC are 4D. Consider a 2D point p = (x,y). In HC, we represent p as p = (x,y,1). An extra coordinate is added whose value is always 1. This may seem odd but it allows us to now represent translations as matrix multiplication instead of as vector addition. A translation (dx, dy) which would normally be performed as q = (x,y) + (dx, dy) now is written as

$$\mathbf{q = T\ p} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+dx \\ y+dy \\ 1 \end{bmatrix}$$

Now, we can write the scaling about a fixed point as simply a matrix multiplication:

q = (-T) S T p = A p,

where A = (-T) S T

The matrix A can be calculated once and then applied to all the points in the object. This is much more efficient than our previous representation. It is also easier to identify the transformations and their order when everything is in the form of matrix multiplication.

The matrix for scaling in HC is

$$\mathbf{S} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and the matrix for rotation is

$$\mathbf{R} = \begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
/* Program for 2D Transformation */
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#include<stdlib.h>
float a[3][3], b[3][3], c[3][3];
float xr = 100 ,yr = 100,angle = 3.11416/2 , tx = 100 ,ty = 50,sx = 2,sy =2 ,sh =
0.5,x1=120,y1=120,x2=70,y2=170,x3=170,y3=140;
```

```
int i,j;
void ident(float a[3][3])        {
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)          {
                  if(i==j)
                    a[i][j] = 1;
                  else
                     a[i][j] = 0;
                  }
          }

void matb()    {
     b[0][0] =x1; b[0][1] =x2; b[0][2] =x3;
     b[1][0] =y1; b[1][1] =y2; b[1][2] =y3;
     b[2][0] =1;  b[2][1] =1;  b[2][2] =1;
   }

void matrixmult(float a[3][3], float b[3][3])    {
     int k;
     double temp;
     for(i=0;i<3;i++)     {
          for(j=0;j<3;j++)          {
              temp = 0;
              for(k=0;k<3;k++)             {
                    temp+=(a[i][k]*b[k][j]);
                  }
              c[i][j] = temp;
           }
         }
   }

void show()    {
```

```
    setcolor(WHITE);
    line(320+c[0][0],240-c[1][0],320+c[0][1],240-c[1][1]);
    line(320+c[0][1],240-c[1][1],320+c[0][2],240-c[1][2]);
    line(320+c[0][2],240-c[1][2],320+c[0][0],240-c[1][0]);
}

void trans() {
    ident(a);
    a[0][2] = tx;
    a[1][2] = ty;
    matb();
    matrixmult(a,b);
    show();
}

void rotate(){
   ident(a);
   a[0][2] = xr;
   a[1][2] = yr;
   ident(b);
   b[0][0] = cos(angle);
   b[1][0] = sin(angle);
   b[0][1] = -sin(angle);
   b[1][1] = cos(angle);
   matrixmult(a,b);
   for(i =0;i<3;i++)
     for(j=0;j<3;j++)
          a[i][j] = c[i][j];
   ident(b);
   b[0][2] = -xr;
   b[1][2] = -yr;
   matrixmult(a,b);
```

```
     for(i =0;i<3;i++)
      for(j=0;j<3;j++)
            a[i][j] = c[i][j];
    matb();
    matrixmult(a,b);
    show();
  }

void triangle()    {
      b[0][0] = 0; b[0][1] = 100; b[0][2] = 0;
      b[1][0] = 0; b[1][1] = 100; b[1][2] = 100;
      b[2][0] = 1; b[2][1] = 1;   b[2][2] = 1;
    }

void reflecty()    {
     ident(a);
     a[0][0] = -1;
     matb();
     matrixmult(a,b);
     show();
    }

void scale()    {
      ident(a);
      a[0][0] = sx;
      a[0][2] = xr*(1-sx);
      a[1][1] = sy;
      a[1][2] = yr*(1-sy);
      matb();
      matrixmult(a,b);
      show();
    }
```

```
void sheary()    {
    ident(a);
    a[1][0] = sh;
    triangle();
    setcolor(RED);
    line(320+b[0][0],240-b[1][0],320+b[0][1],240-b[1][1]);
    line(320+b[0][1],240-b[1][1],320+b[0][2],240-b[1][2]);
    line(320+b[0][2],240-b[1][2],320+b[0][0],240-b[1][0]);
    matrixmult(a,b);
    show();
   }

void first()    {
     line(0,240,640,240);
     line(320,0,320,480);
     line(320+x1,240-y1,320+x2,240-y2);
     line(320+x2,240-y2,320+x3,240-y3);
     line(320+x3,240-y3,320+x1,240-y1);
   }

void main()    {
     int gd=DETECT,gm;
     int choice;
     while(1)    {
          clrscr();
          printf("\n 1. Translation  \n 2. Rotation \n 3. Reflection along Y\n 4. Scaling \n 5.
Shearing along Y \n 6. Exit ");
          printf("\n\n Enter your choice :");
          scanf("%d",&choice);
          initgraph(&gd,&gm,"d:\\tc\\bgi");
          setcolor(BLUE);
```

```
switch(choice)          {
  case 1:
        first();
        trans();
        outtextxy(330,300,"Translation : tx = 100 , ty = 50");
        break;
  case 2:
        first();
        rotate();
        outtextxy(330,300,"Rotation : angle = 90");
        break;
  case 3:
        first();
        reflecty();
        outtextxy(330,300,"Reflection along Y");
        break;
  case 4:
        first();
        scale();
        outtextxy(330,300,"Scaling :  sx = 2 , sy = 2");
        break;
  case 5:
        line(0,240,640,240);
        line(320,0,320,480);
        sheary();
        outtextxy(330,300,"Shearing along Y : sh = 0.5");
        break;
  case 6:
        exit(0);
  default:
        printf("Wrong choice");
 }
```

```
        getch();
        closegraph();
       }
    }
```

Chapter 4

Graphical Software

Graphical software is used to create images as well as processing the images to make it realistic. There are two types of graphics software.

1. General Programming Package
2. Special Purpose Application Package.

1. General Programming Package:

This package provides set of graphics functions such as line, circle, polygon etc and these function can be used in high level programming language such as C, C++ etc. These packages are specially designed for programmers.

2. Special Purpose Application Package:

These programs are designed for non-programmers so that the user can generate displays without worrying about how graphic operation works. Example: Paintbrush, CAD (Computer Aided Design).

Need for Machine Independent Graphical Languages **[*PU-2010, PU-2011*]**

In Computer Science, a machine-independent program is any program that can be run by any computer, without regard to its architecture or even its operating system.

- Primary goal of standardized graphics software is **Portability.**
- When packages are designed with standard graphics function, software can be moved easily from one hardware to another.
- Without standards, program designed form one hardware often cannot be transferred to another system without extensive rewriting of the programs. Thus Machine Independence Graphical language is required.
- National and International Standards Organization in many countries has cooperated in an effort to develop generally accepted standards for computer graphics.

- **Graphical Kernel System (GKS)**:

It was adopted as the first graphics software standard by International Standardization Organization (ISO) and American National Standard Institute (ANSI).

Original GKS was 2D and later, 3D extension was developed.

- **Programmer's Hierarchical Interactive Graphics Standard (PHIGS):**

It is an extension of GKS with increased capabilities for object modeling, color specifications, surface rendering and picture manipulation.

An extension of PHIGS called PHIGS+ was developed to provide 3D surface rendering capabilities.

PHIGS doesn't provide a standard methodology for a graphics interface to output devices not for storing and transmitting pictures. So Separate Standards were developed.

- Standard graphics functions are defined as a set of specifications that is independent of any programming language.

Direct X

- Direct X is a set of development libraries for high performance games under windows.
- It consists of:
  a. Direct Draw: 2D graphics programming
  b. Direct Sound: Allows mixing of sounds.
  c. Direct Play: Allow multiplayer games to connect via modems, LANs etc.
  d. Direct Input: Allow joysticks and handles input from various peripherals.
  e. Direct 3D: 3D graphics programming.

Graphics file format

The command-based, GUI-based and scanned/digitized graphics can be stored in variety of file formats on a computer disk (hard disk). The format of a stored graphic image directly affects the way it can be used or revised later. For eg, when a file is stored as bit-mapped images, it only save "on/off" of pixel pattern. Bitmapped files are commonly known in some systems as paint files. **TIFF** (Tagged Image File Format) files also store bit-map graphics, but can include additional grayscale and color information. Other formats allow a graphics to be stored on disk as a collection of one or more individually defined and editable "objects". Instead of storing actual graphic as a bit map, the visual attributes of the graphic are stored as a group of mathematically defined objects. In this way, the graphic is simply redrawn by the computer every time it is retrieved from disk to RAM. Almost all of the latest graphics packages that use the GUI-based approach store the graphics in this way.

1. BMP (Window bitmap)
2. JPEG (Join Photographic Expert Group)
3. TIFF (Tagged Image File Format)
4. GIF (Graphics Interchange Format)

The following is the information coded in an image file:

1. *Format/Version Identifier:* Format being used including the version number.
2. *Image Width and Height in pixels:* It is given in number of pixels.
3. *Image Type:* Common type includes b and w, 8 bit color, 24 bit color etc.

4. *Image data format:* It specifies the order in which pixel values are stored in the image data selection. For eg: left to right, top to bottom. The values of image data may be compressed by using compression algorithm. Eg: Run Length Encoding (RLE).

BMP (Window bitmap)

- Commonly used by Microsoft Window program and windows OS (operating System) itself.
- Here, Lossless Compression is specified but some use only uncompressed files.
- It is a binary file.
- It is pixel based format that only supports RGB color.
- Bit depth can be 1, 4, 8 or 24 bits.

GIF (Graphics Interchange Format)

- Not used for prepress application. It is more suit for web design or to exchange images through e-mails or newsgroups.
- Most common on the web.
- It is limited to 256 colors.
- Since images with millions of colors are reduced to 256 colors, there is reduction in quality.
- GIF is best suited for images with few colors like chart, graphs, logos, icons etc.
- Photographic images with more than 256 colors should never be considered for GIF file type.
- GIF: uses lossless compression algorithm. There is no information loss.
- GIF : come in two flavors
    - GIF87a: no animation
    - GIF89a: supports animation

In summary GIF is: 8 bit + Compression + animation

JPEG (Join Photographic Expert Group)

- Stands for Join Photographic Expert Group which is a standardization committee.
- It is also stands for the compression algorithm that was invented by the committee.

- JPEG compressed images are often stored in a file format called JFIF (JPEF file interchange format) which most people refer to as JPEG.
- JPEG: can have millions of colors.
- It uses lossy compression to make images smaller in size. So there is loss of image quality.

TIFF (Tagged File Format)

- It is standard for images that will be placed in desktop publishing programs.
- It can be neatly transported across platforms and compressed to reduce the file size.
- It is device independent so that it can be used on PCs, MACs and UNIX workstation.

PNG (Portable Network Graphics)

- Portable Network Graphic.
- Lossless Compression Approach. In lossless approach, the decoded (decompressed) copy is exact replica of the original.
- It is used mostly in web application.

Graphics software:

1. Paint program: Paint program works with bit map images.

2. Photo manipulation program: It works with bit map images and is widely used to edit digitized photographs.

3. Computer Aided Design program:

CAD software is used in technical design fields to create models of objects that will be built or manufactured. CAD software allows user to design objects in 3 dimensions and can produce 3-D frames and solid models.

4. 3-D Modelling Programs:

Is used to create visual effects. 3-D modeling program works by creating objects like surface, solid, polygon etc.

5. Animation:

Computer are used to create animation for use in various fields, including games, and movies composing to allow game makers and film makers to add characters and objects to scenes that did not originally contain them.

Error Diagnostics:

Programming errors often remain undetected until an attempt is made to compile the program once the compile command has been issued. However, the presence of certain errors will become readily apparent, since these errors will prevent the program from being compiling successfully, some particular common errors of this types are declaring constants and variables improperly, a reference to an undeclared variable and incorrect punctuation. Such errors are referred as syntactical error or grammatical error. Most version of C program will generate a diagnostic message when a syntactical error has been detected (the compiler usually come to an interrupt when this happens). These diagnostic messages are not always completely straight forward in their meaning, but they are nevertheless helpful in identifying the nature and location of the error.

Logical Debugging:

Syntactical errors and certain types of logical errors will cause diagnostic messages to be generated when compiling or executing a program. Errors of these types are easy to find and correct. Some types of logical errors can be much more difficult to detect, however, since the output resulting from a logically incorrect program may appear to be error free. Moreover, logical errors are often hard to find even when they are known to exits (as for example, when the computed output is obviously incorrect)

Chapter 5 & 7

Project Development

Project

Project is a set of activities which has got definite starting time and final end time. It has a life cycle. All the projects operate within constrains of time, cost and quality performance.

#Definition

According to "**Harold Kerzner**"

"A project is any series of activities and tasks that

- Have a specific objectives to be completed within certain specifications
- Have defined start and end dates

- Have funding limits
- Consume resources"

According to **Trevor L. Young**

"A project is a collection of linked activities, carried out in an organized manner with a clearly defines start point and finish point to achieve some specific result that specify the needs of an organization as derived from the current business plans.**"**

Every project has specific objectives. It is project people who make or break the project. It works within constrains of time, cost and quality performance in a dynamic environment.

Project approach is based on sound planning, clear objectives logical coordination of resource utilization, and effective implementation with total responsibility to project manager and timely control and evaluation.

Characteristics of Project

1. Specific objectives
2. life span -> beginning and end of a project
3. constrains -> time, cost and quality
4. unique -> each project is different
5. Team work -> project manager as a leader
6. flexibility -> adapt to change
7. resource integration
8. planning and control
9. contracting and sub contracting
10. specific Beneficiaries

Project Life Cycle

A project has definite start and end point of dates. Project has a fixed life span. The life span of a project is divided into phases.

1. Formulation Phase
2. Planning Phase

3. Implementation Phase
4. Termination Phase

Project Management

Project management is the discipline of planning, organizing, securing and managing resources to bring about the successful completion of specific project goals and objectives.

Project Planning

Project planning is a decision-making task as it involves choosing among alternatives. One of the objectives of project planning is to completely define all works required, so that it will be readily identifiable to each project participant.

Definition:

- Project planning is defined as developing the plan in the required level of details with accompanying milestones and the use of available tools for preparing and monitoring the plan.
- It is a rational determination of how to initiate, sustain and terminate a project.

  Thus we can define project planning as a decision based on futurity.

According to Richard Steers

"*Planning is the process by which managers define goals and take necessary steps to ensure that these goals are achieves.*"

Reasons for Project Planning:

1. To eliminate or minimize uncertainty and risk.
2. To improve the efficiency of operations.
3. To acquire better understanding of the objectives.
4. To develop a sense of urgency and time consciousness.
5. To induce people to look ahead.

6. To provide means of communication and coordination between all those involved in the project.
7. To provide a bases for organizing the work in the project and allocating responsibility to the individuals.

Area of Project Planning:

Comprehensive project planning covers the following areas:

1. Planning the project work.
2. Planning the human resource and organization.
3. Planning the financial resources.
4. Planning the information system.

Project Planning Model:

A conceptual model depicting project planning is shown in the figure below.

1. Organization mission includes the business and organization is in.
2. Project objectives show the desired future position of project in terms of cost, schedule and technical performance.
3. Project goals refer to the milestones leading to the completion of project with package.
4. Project strategy is a plan of action with accompanying policies providing general direction of how resources will be used to accomplish project goals and objectives.
5. Organizational structures focus on the project driven functions and processes.
6. Project teamwork involves identification, negotiations and resolution of individual and collective authority and responsibility.
7. Style refers to project manager and project team member's manner, knowledge, skills and attitudes.
8. Project resources show the quality and human and non-human resources to support the project.

Project Planning Process

Planning aims at achieving the project completion, making the most effective use of time and resources. Project planning can be short term or long term as project may extend for many years. Thus, project planning requires both the operational and strategic thinking and decision making. It is characterized by creativity, innovation and ability to think rationally and prospectively. This forms the basis for project planning process which is a multi-stage process which can be defined as follows.

1. Analysis of the environment

Internal and external environment should be analyzed thoroughly. This analysis and forecasting of environment provides with critical factors such as planning premises. These lay down the boundary or limitations within which plans have to be formulated and implemented.

2. Establishment of objectives

On the basis of environmental conditions, goal of the project are laid down. It is formulated in all key areas of project operations. It should be laid down in precise and specific terms so that it can be comprehended by all the concerned of the project.

3. Identification of key factors of the project

This involves identification of various stages of project. It includes finding of activities, which pertain to major subdivision of project such as formulation, planning, implementation and termination. It usually requires imagination and foresight.

4. Identification of key elements of project

This identifies the key activities of project with the assistance of break down structure. It has to be done at each stage of the project.

5. Establishing the logical sequencing of activities

The sequencing of activities can be established by using the network analysis. The critical path is determined for each activities of the project.

6. Identification of time and resources

This involves estimation of time and resource requirements of the project. Three estimates of time are made. This includes optimistic (shortest possible time), pessimistic (longest time) and the normal time (most likely time).

7. Assignment of responsibility

Each activity of project should be allocated to responsibility centers: individual, unit and department.

8. Finalize project plan

These required resources are allocated to each activity of project. The project schedule is made for each activity and the project plan is finalized.

#Project planning techniques

1. Network analysis
2. Input / output analysis
3. Financial analysis

1. Network Analysis

It provides the framework for defining the activities to be done in the project, integrating the activities in a logical time sequence and dynamic control over the progress of the project plan. The best techniques for network analysis are

1) PERT (program Evaluation and Review Technique)
2) CPM (Critical Path Method)

PERT

- Various activities of a project are identified. Work break down structure is used for this purpose.
- The order of precedence is determined. The identified activities need to be done first before starting others. This is done through a diagram.
- Time estimates are made

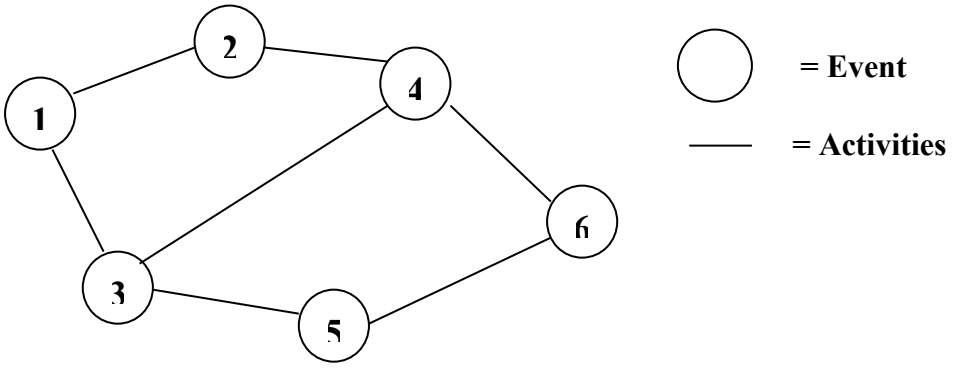


Figure: PERT Network

CPM

This technique is used for project planning, sequencing and control where the emphasis is on optimizing resource allocation and minimizing overall cost for a given execution time.

2. Input / Output Analysis

- Input analysis
- Human Resources
- Materials
- Machinery

- Money
- Information

3. Financial Analysis

It studies the financial sustainability of the project

- Capital requirements
- Source of funds
- Projected cash flow
- Account reporting system
- Project profitability

Gantt chart (Bar Chart)

It is a graphical representation of actual project progress Vs. planned progress within a time frame. It is a bar graph. A Gantt chart is name after its developer Henry Gantt and useful for scheduling bugging and resource planning. Each bar in a Gantt chart represents an activity. The bars are drawn along a time line. The length of each bar is proportional to the duration of time for the corresponding activity.

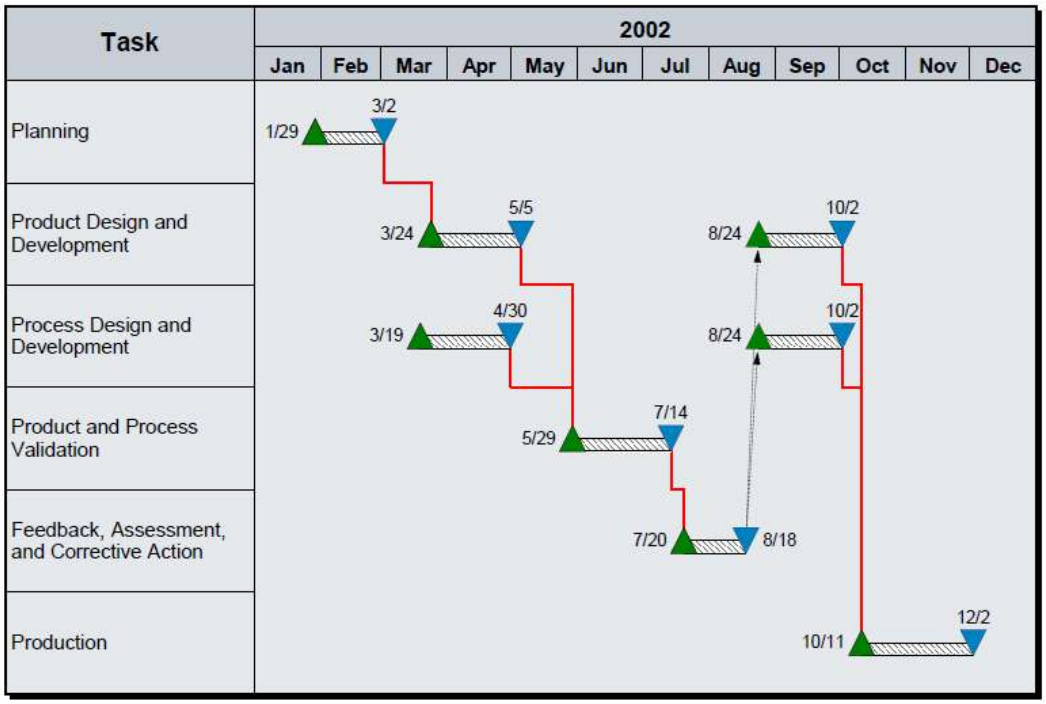


Figure: Gantt Chart of Product Development Schedule

#Element of project Management

There are three major elements

1) Project manager
2) Project team
3) Project management system

1. Project Manager

- Integrate people from various functional areas to achieve specific project goals
- PM provide motivation to the employee (project member and employee)
- PM perform the project compilation
- PM can provide necessary training to employee
- PM should have capability of leadership
- PM is an expert. He/She divide whole project into modules and assign that module to group of employee
- At the end of the project completion he/she integrate all the modules to form a complete package

2. Project Team

- Project team is a group of people from different field led by project manager
- Project member are the knowledgeable people who actually do the project.

3. Project Management system

- It consists of organizational structure and information system.
- It is a top level management system that defines relationship between project member and project manager
- It provides the interactive planning and control of the performance, cost or resources use, the inter related effects of schedule changes and the project times and cost of project completion

[For more detail please go through Project management and organization and management by Govinda Ram Agrawal]

Chapter 6:

Three Dimensional

Visible Surface Detection or Hidden surface

We must determine what is visible within a scene from a chosen viewing position For 3D worlds this is known as **visible surface detection** or **hidden surface elimination**

Visible surface detection algorithms are broadly classified as:

- **Object Space Methods:** Compares objects and parts of objects to each other within the scene definition to determine which surfaces are visible
- **Image Space Methods:** Visibility is decided point-by-point at each pixel position on the projection plane

Image space methods are by far the more common

Back-Face Detection

The simplest thing we can do is find the faces on the backs of polyhedra and discard them

We know from before that a point (*x, y, z*) is behind a polygon surface if: where *A, B, C* & *D* are the plane parameters for the surface This can actually be made even easier if we organise things to suit ourselves

Ensure we have a right handed system with the viewing direction along the negative *z*-axis

Now we can simply say that if the *z* component of the polygon's normal is less than zero the surface cannot be seen

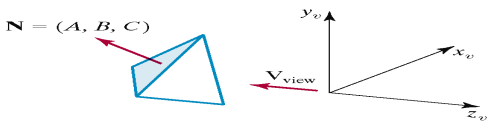


Figure 9-2
A polygon surface with plane parameter $C < 0$ in a right-handed viewing coordinate system is identified as a back face when the viewing direction is along the negative z_v axis.

In general back-face detection can be expected to eliminate about half of the polygon surfaces in a scene from further visibility tests More complicated surfaces though scupper us! We need better techniques to handle these kind of situations

Depth-Buffer Method

Compares surface depth values throughout a scene for each pixel position on the projection plane Usually applied to scenes only containing polygons As depth values can be computed easily, this tends to be very fast Also often called the z-buffer method

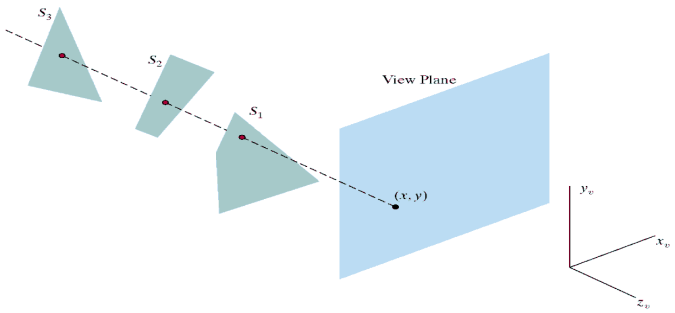


Figure 9-4
Three surfaces overlapping pixel position (x, y)on the view plane. The visible surface, S_1, has the smallest depth value.

1. Initialise the depth buffer and frame buffer so that for all buffer positions (*x, y*)

   depthBuff(x, y) = 1.0

   frameBuff(x, y) = bgColour
2. Process each polygon in a scene, one at a time
   - For each projected (*x, y*) pixel position of a polygon, calculate the depth *z* (if not already known)
   - If z < depthBuff(x, y), compute the surface colour at that position and set

     depthBuff(x, y) = z

     frameBuff(x, y) = surfColour(x, y)

After all surfaces are processed depthBuff and frameBuff will store correct values

$$z = \frac{-Ax - By - D}{C}$$

At any surface position the depth is calculated from the plane equation as:

For any scan line adjacent *x* positions differ by ±1, as do adjacent *y* positions

$$z' = \frac{-A(x-1) - By - D}{C} \qquad z' = z + \frac{A}{C}$$

The depth-buffer algorithm proceeds by starting at the top vertex of the polygon Then we recursively calculate the *x*-coordinate values down a left edge of the polygon The *x* value for the beginning position on each scan line can be calculated from the previous one

$$x' = x + \frac{1}{m}$$

where *m* is the slope

Depth values along the edge being considered are calculated using

$$z' = z + \frac{A/m + B}{C}$$

A-Buffer Method

The A-buffer method is an extension of the depth-buffer method The A-buffer method is visibility detection method developed at Lucasfilm Studios for the rendering system REYES (**R**enders **E**verything **Y**ou **E**ver **S**aw)

The A-buffer expands on the depth buffer method to allow transparencies The key data structure in the A-buffer is the *accumulation buffer*

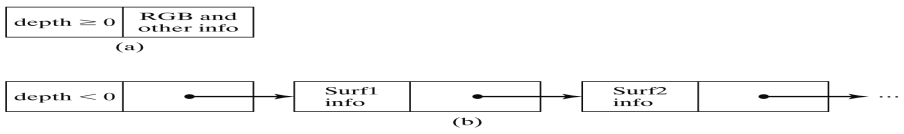


Figure 9-9
Two possible organizations for surface information in an A-buffer representation for a pixel position. When a single surface overlaps the pixel, the surface depth, color, and other information are stored as in (a). When more than one surface overlaps the pixel, a linked list of surface data is stored as in (b).

If depth is >= 0, then the surface data field stores the depth of that pixel position as before

If depth < 0 then the data filed stores a pointer to a linked list of surface data

Surface information in the A-buffer includes:

- RGB intensity components
- Opacity parameter
- Depth
- Percent of area coverage
- Surface identifier
- Other surface rendering parameters

The algorithm proceeds just like the depth buffer algorithm

The depth and opacity values are used to determine the final colour of a pixel

Scan-Line Method

An image space method for identifying visible surfaces Computes and compares depth values along the various scan-lines for a scene

Two important tables are maintained:

- The edge table
- The surface facet table

The edge table contains:

- Coordinate end points of reach line in the scene

- The inverse slope of each line
- Pointers into the surface facet table to connect edges to surfaces

The surface facet tables contains:

- The plane coefficients
- Surface material properties
- Other surface data
- Maybe pointers into the edge table

To facilitate the search for surfaces crossing a given scan-line an active list of edges is formed for each scan-line as it is processed

The active list stores only those edges that cross the scan-line in order of increasing x

Also a flag is set for each surface to indicate whether a position along a scan-line is either inside or outside the surface

Pixel positions across each scan-line are processed from left to right At the left intersection with a surface the surface flag is turned on At the right intersection point the flag is turned off We only need to perform depth calculations when more than one surface has its flag turned on at a certain scan-line position

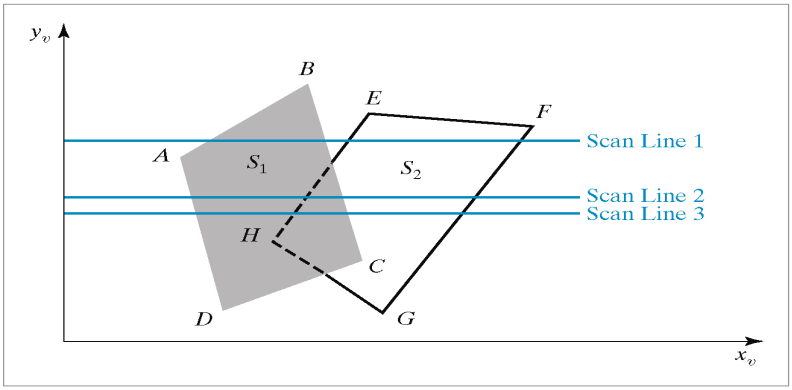


Figure 9-10

Scan lines crossing the view-plane projection of two surfaces, S_1 and S_2. Dashed lines indicate the boundaries of hidden surface sections.

Scan-Line Method Limitations

The scan-line method runs into trouble when surfaces cut through each other or otherwise cyclically overlap

Such surfaces need to be divided

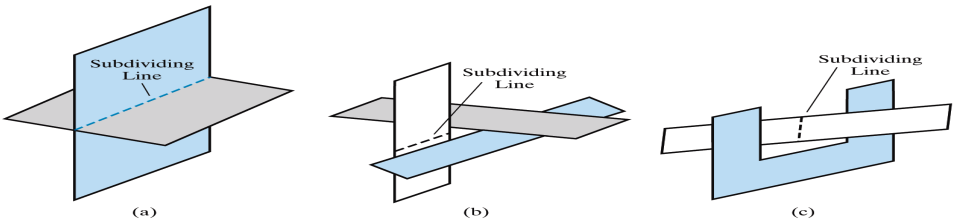


Figure 9-11
Intersecting and cyclically overlapping surfaces that alternately obscure one another.

We need to make sure that we only draw visible surfaces when rendering scenes

There are a number of techniques for doing this such as

- Back face detection
- Depth-buffer method
- A-buffer method
- Scan-line method

Next time we will look at some more techniques and think about which techniques are suitable for which situations

Illumination Models & Surface-Rendering Methods

Illumination model or a ***lighting model*** is the model for calculating light intensity at a single surface point.

Surface rendering is a procedure for applying a lighting model to obtain pixel intensities for all the projected surface positions in a scene.

Illumination models

Given the parameters:

- the optical properties of surfaces (opaque/transparent, shiny/dull, surface-texture);
- the relative positions of the surfaces in a scene;
- the color and positions of the light sources;
- The position and orientation of the viewing plane.

Illumination models calculate the intensity projected from a particular surface point in a specified viewing direction.

Light Sources

- When we view an opaque nonluminous object, we see reflected light from the surfaces of the object.
- The total reflected light is the sum of the contributions from ***light sources*** and other reflecting surfaces in the scene.
- Light sources = ***light-emitting sources****.*
- Reflecting surfaces = ***light-reflecting sources****.*

Light Sources

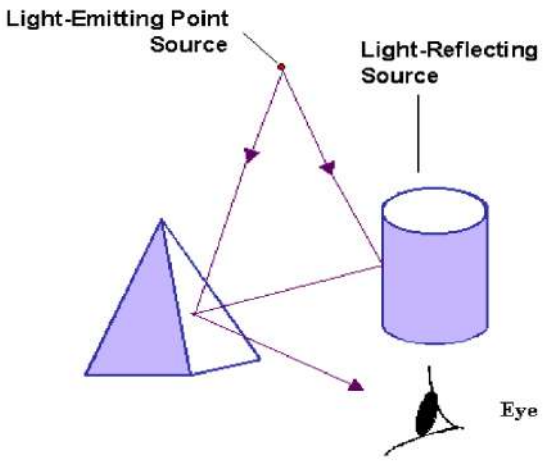


Fig. 1 Light viewed from an opaque surface is in general a combination of reflected light from a light source and reflections of light reflections from other surfaces.

Point Light Source

- The rays emitted from a point light radially diverge from the source.
- Point sources are abstraction of real-world sources of light such as light bulbs, candles, or the sun.
- The light originates at a particular place; it comes from a particular direction over a particular distance.
- Surfaces facing towards and positioned near the light source will receive more light than those facing away from or far removed from the source following radially diverging paths as shown in fig. 2.

Distributed Light Source

- A nearby source, such as the long fluoresent light.
- All of the rays from a directional/distributed light source have the same direction, and no point of origin.

- It is as if the light source was infinitely far away from the surface that it is illuminating.
- Sunlight is an example of an infinite light source.

Materials

- When light is incident on an opaque surface, part of it is reflected and part is absorbed.
- Shiny materials reflect more of the incident light, and dull surface absorb more of the incident light.

For an illuminated transparent surface, some of the incident light will be reflected and some will be transmitted through the material.

Diffuse reflection

- Grainy surfaces scatter the reflected light in all directions. This scattered light is called ***diffuse reflection***.
- The surface appears equally bright from all viewing directions.

What we call the color of an object is the color of the diffuse reflection of the incident light.

Specular reflection

Light sources create highlights, bright spots, called ***specular reflection***. More pronounced on shiny surfaces than on dull.

Basic Illumination Models

Lighting calculations are based on:

- Optical properties of surfaces, such as glossy, matte, opaque, and transparent. This controls the amount of reflection and absorption of incident light.
- The background lighting conditions.
- The light-source specifications. All light sources are considered to be point sources, specified with a coordinate position and intensity value (color).

Ambient Light

- A surface that is not directly exposed to a light source will still be visible if nearby objects are illuminated due to light reflecting from nearby objects.
- Ambient light has no spatial or directional characteristics.

- The amount of ambient light incident on each object is a constant for all surfaces and over all directions.
- The amount of ambient light that is reflected by an object is constant for all surfaces and over all direction, but the intensity of reflected light for each surface depends only on the optical properties of the surface.

Ambient Light

- The level of ambient light in a scene is a parameter I_a , and each surface illuminated with this constant value.
- Illumination equation for ambient light is

$$I = k_a I_a$$

where

I is the resulting intensity

I_a is the incident ambient light intensity

k_a is the object's basic intensity, ***ambient-reflection coefficient or ambient reflectivity***.

Diffuse Reflection

- Diffuse reflections are constant over each surface in a scene, independent of the viewing direction.
- The amount of the incident light that is diffusely reflected can be set for each surface with parameter k_d, the ***diffuse-reflection coefficient***, or ***diffuse reflectivity***.

$0 \square k_d \square 1$;

k_d near 1 – highly reflective surface;

k_d near 0 – surface that absorbs most of the incident light;

k_d is a function of surface color;

Even though there is equal light scattering in all direction from a surface, the brightness of the surface does depend on the orientation of the surface relative to the light source:

Diffuse Reflection

As the angle between the surface normal and the incoming light direction increases, les of the incident light falls on the surface.

We denote the ***angle of incidence*** between the incoming light direction and the surface normal as □. *Thus, the amount of illumination depends on cos□. If the incoming light from the source is perpendicular to the surface at a particular point, that point is fully illuminated.*

If I_l is the intensity of the point Light source, then the diffuse reflection equation for a point on the surface can be written as

$$I_{l,diff} = k_d I_l \cos\square$$

or

$$I_{l,diff} = k_d I_l (\mathbf{N} \cdot \mathbf{L})$$

where **N** is the unit normal vector to a surface and **L** is the unit direction vector to the point light source from a position on the surface.

Figure 10 illustrates the illumination with diffuse reflection, using various values of parameter k_d between 0 and1.

We can combine the ambient and point-source intensity calculations to obtain an expression for the total diffuse reflection.

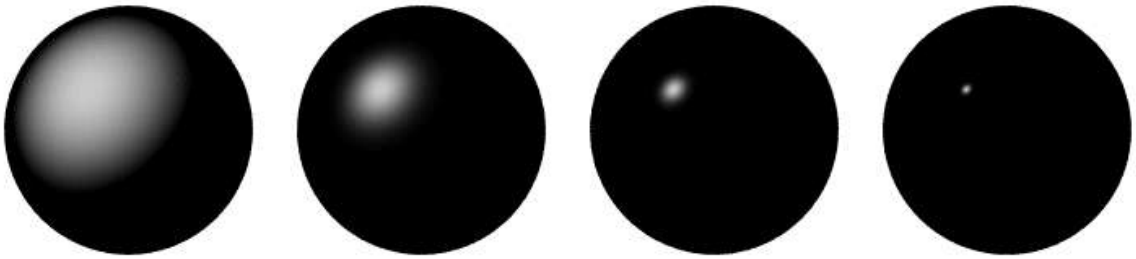
$$I_{diff} = k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L})$$

where both k_a and k_d depend on surface material properties and are assigned values in the ange from 0 to 1.

Specular Reflection and the Phong Model

- ***Specular reflection*** is the result of total, or near total, reflection of the incident light in a concentrated region around the ***specular-reflection angle.***
- Shiny surfaces have a narrow specular-reflection range.

Dull surfaces have a wider reflection range.



Specular Reflection

Figure 13 shows the specular reflection direction at a point on the illuminated surface. In this figure,

- **R** represents the unit vector in the direction of specular reflection;
- **L** – unit vector directed toward the point light source;
- **V** – unit vector pointing to the viewer from the surface position;

Angle □ *is the viewing angle relative to the specular-reflection direction* ***R****.*

Phong Specular-Reflection Model

Phong specular-reflection model is an empirical model for calculating the specular-reflection range:

- Sets the intensity of specular reflection proportional to $\cos^{ns}\phi$;
- Angle □ *assigned values in the range* 0^o *to* 90^o*, so that* cos□ *values from 0 to 1;*
- ***Specular-reflection parameter*** n_s is determined by the type of surface,
- ***Specular-reflection coefficient*** k_s equal to some value in the range 0 to 1 for each surface.
- The intensity of specular reflection depends on the materials properties of the surface and the angle of incidence, as well as other factors such as the polarization and color of the incident light.
- According to *Fresnel's Laws of Reflection*,

  Phong specular-reflection model is given as:

$$I_{spec} = W(\Theta) I_l \cos^{ns}\phi$$

Where W(Θ) is a monochromatic specular-reflection coefficient and *Θ=0°* to *Θ=90° or W(Θ)=0 to W(Θ)=1*

- Very shiny surface is modeled with a large value for n_s (say, 100 or more);
- Small values are used for duller surfaces.
- For perfect reflector (perfect mirror), n_s is infinite;

Phong specular-reflection model:

$$I_{spec} = k_s I_l \cos^{ns}\phi$$

Since **V** and **R** are unit vectors in the viewing and specular-reflection directions, we can calculate the value of $\cos^{ns}\phi$ *with the dot product* ***V·R****.*

$$I_{spec} = k_s I_l (\mathbf{V}\cdot\mathbf{R})^{ns}$$

$$\mathbf{R} + \mathbf{L} = (2\mathbf{N}\cdot\mathbf{L})\mathbf{N}$$

$$\mathbf{R} = (2\mathbf{N}\cdot\mathbf{L})\mathbf{N} - \mathbf{L}$$

$$\phi = \alpha/2$$

$$\mathbf{H} = (\mathbf{L} + \mathbf{V})/|(\mathbf{L} + \mathbf{V})|$$

$$I_{spec} = k_s I_l (\mathbf{N}\cdot\mathbf{H})^{ns}$$

Combine Diffuse & Specular Reflections

For a single point light source, we can model the combined diffuse and specular reflections from a point on an illuminated surface as

$$I = I_{diff} + I_{spec}$$

$$= k_a I_a + k_d I_l(\mathbf{N}\cdot\mathbf{L}) + k_s I_l(\mathbf{N}\cdot\mathbf{H})^{ns}$$

Combine Diffuse & Specular Reflections with Multiple Light Sources

If we place more than one point source in a scene, we obtain the light reflection at any surface point by summering the contributions from the individual sources:

$$I = k_a I_a + \sum_{i=1}^{n} I_{li} [k_d (N \cdot L_i) + k_s (N \cdot H_i)^{ns}]$$

Intensity Attenuation

As radiant energy from a point light source travels through space, its amplitude is attenuated by the factor $1/d^2$, where d is the distance that the light has traveled.

A surface close to the light source (small d) receives higher incident intensity from the source than a distant surface (large d).

- Problem in using the factor $1/d^2$ to attenuate intensities:

  The factor $1/d^2$ produces too much intensity variations when d is small, and it produces very little variation when d is large.

- We can compensate for these problems by using inverse linear or quadratic functions of d to attenuate intensities.
- A general inverse quadratic ***attenuation function***:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

- The value of the constant term a_0 can be adjusted to prevent $f(d)$ from becoming too large when d is very small.

Surface rendering

- Surface rendering can be performed by applying the illumination model to every visible surface point, or the rendering can be accomplished by interpolating intensities across the surface from a small set of illumination-model calculations.
- Scan-line algorithms use interpolation schemes.
- Surface-rendering procedures are termed ***surface-shading methods***.

Shadows

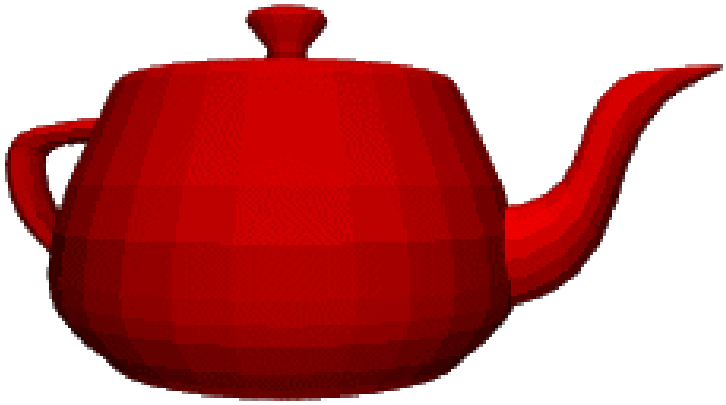
- Hidden-surface methods can be used to locate areas where light sources produce shadows.
  - Apply a hidden-surface method with a light source at a view position.
  - Shadow patterns generated by a hidden-surface method are valid for any selected viewing position, as long as the light-source positions are not changed.
- In polygon-based system, we can add surface-detail polygons that correspond to shadow areas of surface polygons.
- We can display shadow areas with ambient light intensity only, or we can combine the ambient light with specified surface texture.

Polygon Rendering Methods

- The application of an illumination model to the rendering of standard graphic objects;
- Three methods, each of them treats a single polygon independent of all others (non-global):
  - Constant-Intensity Shading / Flat Shading;
  - Intensity-Interpolation Shading / Gourad Shading;
  - Normal-vector Interpolation Shading / Phong Shading
- Each polygon can be rendered with a single intensity, or the intensity can be obtained at each point of the surface using an interpolation scheme.

Constant-Intensity Shading Flat Shading

- Fast & simple.
- A single intensity is calculated for each polygon.
- All points over the surface of the polygon are displayed with the same intensity value.
- Useful for quickly displaying the general appearance of a curved surface.



- Flat shading provides an accurate rendering for an object if all of the following assumptions are valid:
  - The object is a polyhedron and is not an approximation of an object with a curved surface;
  - All light sources illuminating the object are far from the surface so that **N**·**L** and the attenuation function are constant over the surface;
  - The viewing position is also far from the surface so that **V**·**R** is constant over the surface;

Gouraud Shading

- ***Intensity-interpolation*** scheme, referred to as ***Gouraud shading***, renders a polygon surface by linearly interpolating intensity values across the surface.
- Intensity values for each polygon are matched with the values of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with Gouraud

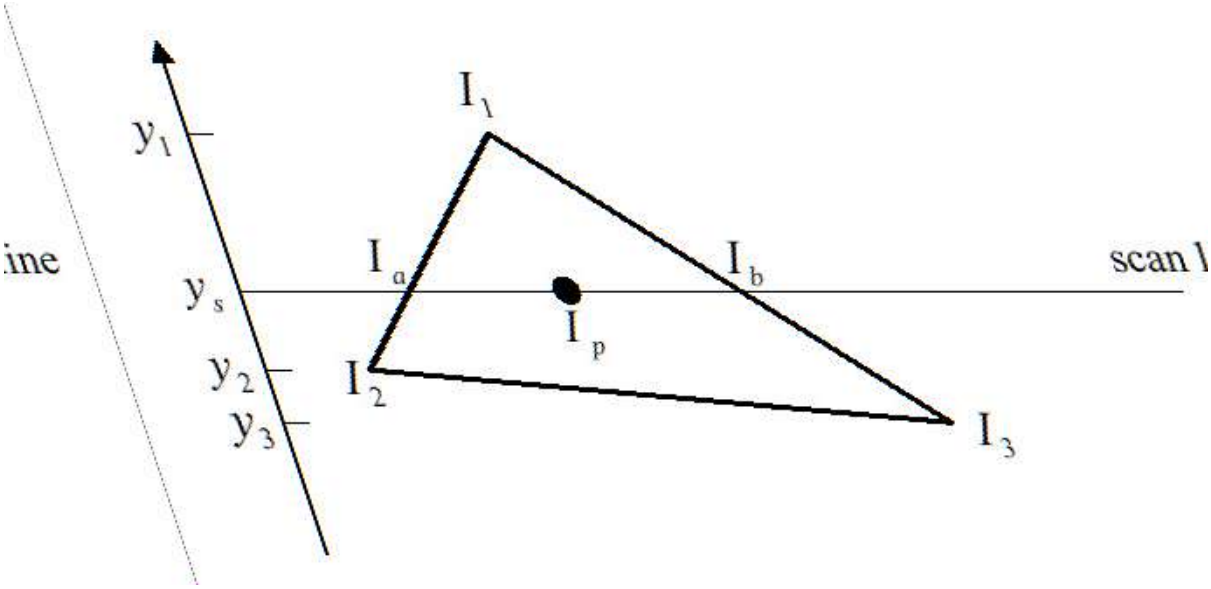
shading by performing the following calculations:

- Determine the average unit normal vector at each polygon vertex;

- Apply an illumination model to each vertex to calculate the vertex intensity;
- Linearly interpolate the vertex intensities over the surface of the polygon;

Gouraud Shading - Step 1

For each scan line, the intensity at the intersection of the scan line with a polygon edge is linearly interpolated from the intensities at the edge endpoints.

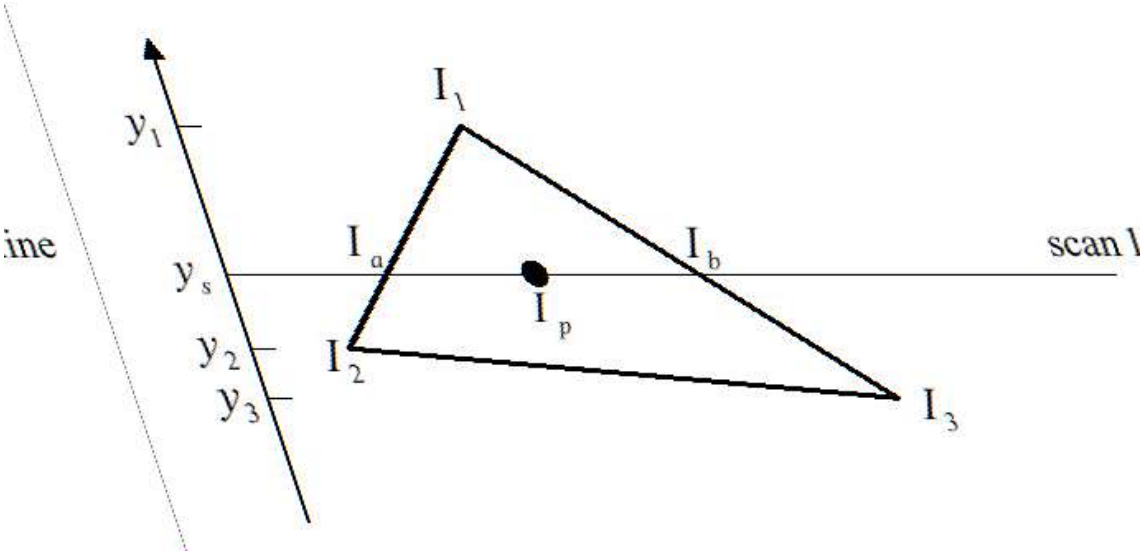


A fast method for obtaining this intensity is to interpolate between intensities of endpoints by using only the vertical displacement of the scan line:

$$I_a \approx I_1 \frac{y_s - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_s}{y_1 - y_2}$$

AND

$$I_b \approx I_1 \frac{y_s - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_s}{y_1 - y_3}$$



Once these bounding intensities are established for a scan line, an interior point (such as *p*) is interpolated from the bounding intensities at points *a* and *b* as

$$I_p \approx I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}$$

Incremental calculations:

If the intensity at edge position (*x,y*) is interpolated as

$$I \approx I_1 \frac{y - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y}{y_1 - y_2}$$

then we can obtain the intensity along this edge for the next scan line, *y*-1, as

$$I' \approx I + \frac{I_2 - I}{y_1 - y_2}$$

Similar calculations are used to obtain intensities at horizontal pixel positions along each scan line.

- Highlights on the surface are sometimes displayed with anomalous shapes.

- Can cause bright or dark intensity streaks to appear on the surface (***Mach-band effect***).

Dividing the surface into a greater number of

polygon faces can reduce these effects.

Phong Shading

- A more accurate method for rendering a polygon surface.
- Interpolates normal vectors, and then applies the illumination model to each surface point.
- Method developed by Phong Bui Tuong.
- Called ***Phong shading,*** or ***normal-vector interpolation shading***.
- More realistic highlights.
- Greatly reduces the Mach-band effect.

Phong Shading Steps:

- Determine the average unit normal vector at each polygon vertex.
- Linearly interpolate the vertex normals over the surface of the polygon.
- Apply illumination model along each scan line to calculate projected pixel intensities for the surface points.

The normal vector **N** for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals:

$$N \approx N_1 \frac{y - y_2}{y_1 - y_2} + N_2 \frac{y_1 - y}{y_1 - y_2}$$

Incremental methods are used to evaluate normals between scan lines and along each individual scan line.

- Produce more accurate results.
- Trade-off: Phong shading requires a lot of calculations.
- Bishop & Weimer developed **Fast Phong Shading** approximation using Taylor series expansion.

Khwopa Engineering College

BE Computer / Electronics (3rd Year)

Computer Graphics

Lab 1: Introduction of Graphics and its implementation using c-programming / C++;
Lab 2: Drawing Lines in Graphics (Horizontal, Vertical, Right-Left diagonal and Left-Right Diagonals)
Lab 3: Implementing Line animations and implementing on Games;
Lab 4: Drawing Circle and ellipse
Lab5: Implementing Circle and ellipse animations
Lab6: Implementing DDA
Lab7: Implementing Bresenham Line drawing algorithm
Lab8: Circle drawing using Mid-point circle drawing algorithm
Lab9: Ellipse drawing using Mid-point ellipse drawing algorithm
Lab10: 2DTranslation and scaling
Lab11: 2D Rotation Transformation
Lab12: 2D Shear Transformation
Lab13: Concept of 3D

~ The End ~

For Any Suggestion:

URL: *www.ganeshramsuwal.com.np*

Email: *suwalganesh@gmail.com*

ganesh@wiseexist.com

Cell: *+977-9841790021*