

LECTURE20: DIGITAL IMAGE PROCESSING

-IMAGE PROCESSING BASICS

-What is an image?

An image is a two-dimensional function $f(x,y)$, where x and y are the spatial(plane) coordinates, and the amplitude of f at any pair of coordinates (x,y) is called the intensity of the image at that level.

-What is a digital image?

If x,y and the amplitude values of f are finite and discrete quantities, we call the image a digital image. A digital image is composed of a finite number of elements called pixels, each of which has a particular location and value.

-Image Processing

is the processing of image data that is the processing of the light intensity of the image or a set of mathematical and logical functions to analyze and manipulate the image. Since ancient times, man has been interested in processing digital images through the following two main items:

1 - **Improving image information** : This item enables users to interpret images and know their content well. The aim of the improvement of images is to create a new image of the original image by increasing the amount of information that can be interpreted visually from the data. The process of improvement is a science and a reflection of the role of the human element in terms of invention and type of improvement.

2 - **Data processing**: It is intended to deal with form data and this treatment is done by machine. The program is programmed to input the information in an equation. Then, for example, the Fourier transform and inverse to improve it and produce an improved image.

-APPLICATIONS OF DIGITAL IMAGE PROCESSING

1- Image sharpening and restoration It refers to the process in which the user can enhance the look of an image. It basically manipulates the images and achieves the desired output. It includes conversion, sharpening, blurring, detecting edges, retrieval, and recognition of images.

2- Medical Field There are several applications under medical field such as Gamma-ray imaging, PET scan, X-Ray Imaging, Medical CT scan and UV imaging

3-Robot vision There are several robotic machines which work on the digital image processing. Through image processing technique robot finds their ways, for example, hurdle detection robot and line follower robot.

4-Pattern recognition It involves the study of image processing, it is also combined with artificial intelligence such that computer-aided diagnosis, handwriting recognition and images recognition can be easily implemented.

5-Video processing A collection of frames or pictures are arranged in such a way that it makes the fast movement of pictures. It involves frame rate conversion, motion detection, reduction of noise and colour space conversion.

-IMAGE REPRESENTATION

Images are mainly classified into two types

1-Analog images.

2-Digital images.

Digital images: Images are derived from analog images but are in spatial domain while analog images are within a frequency domain for easy handling and analysis in the computer. (x, y) This point $f(x, y)$ is used to indicate that the point in Spatial coordinate and its value is proportional to the intensity of the brightness. The digital image can be represented by a matrix determined in rows and columns. Image is converted by **Sampling** process from continuous signal x to comb function as shown in figure 1. Then is quantized by **Quantization** process that converts continuous analog signal into its digital representation as shown in figure 2.

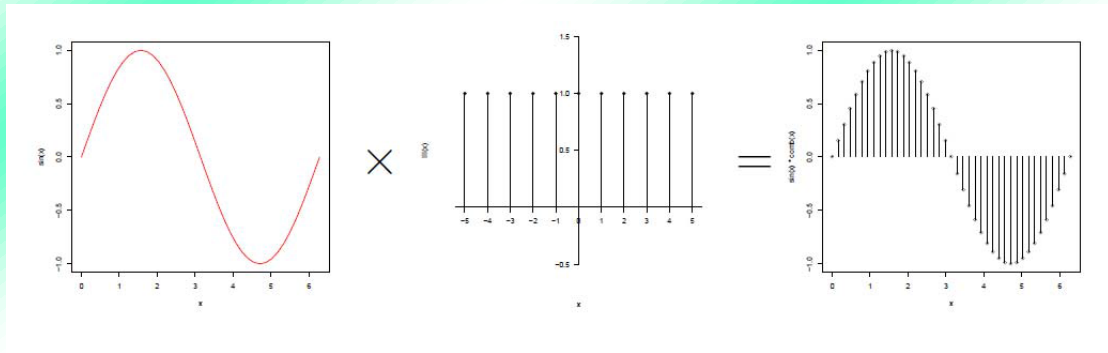


Figure 1 sampling process

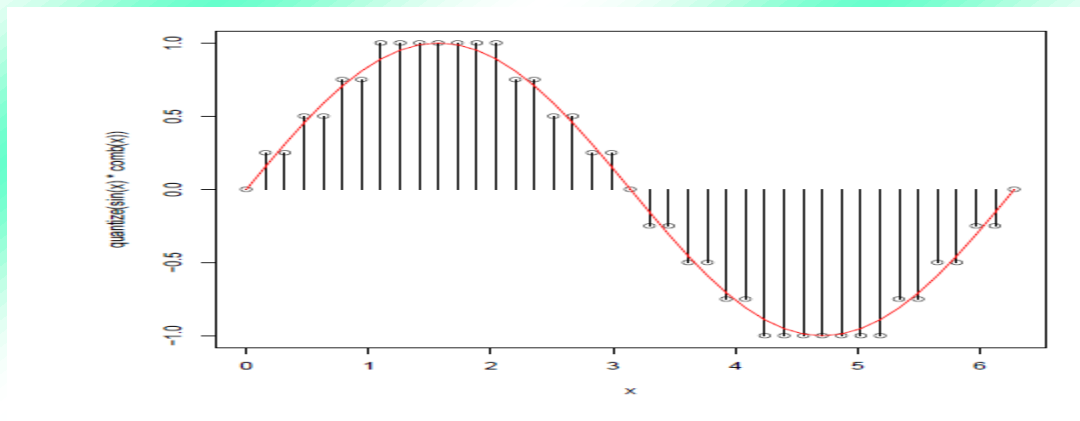


Figure 2 quantization process

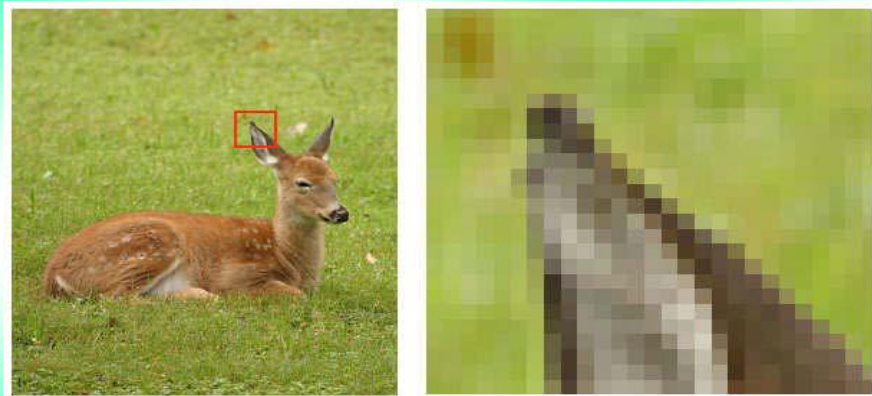


Figure 3 analog to digital conversion image via sampling and quantization process

-Analog Image Processing vs. Digital Image Processing

There are following differences between Analog Images Processing and Digital Image Processing:

Analog Image Processing	Digital Image Processing
The analog image processing is applied on analog signals and form just 2D signals.	The advanced image handling is connected to computerized signals that work on examining and controlling the images.
Analog signal is time-varying signals so the analog images varied under analog image processing.	It enforce digital image accuracy and intensity distribution.
Analog image processing is a slower and pricier process.	Digital image processing is a cheaper , as well as fast process in image storage and retrieval.
Analog signal is a real-world but not good quality of images.	It uses good image compression techniques that provide better reduction and images quality
The advanced picture handling is connected to computerized signals that work on examining and controlling the images.	It uses an image segmentation technique which is used to detect discontinuity which occurs due to a broken connection path.

LECTURE 1 : COMPUTER GRAPHICS INTRODUCTION

INTRODUCTION

Computer graphics It is often abbreviated as CG, is an art of drawing pictures, lines, shape, charts, etc. Using computers with the help of mathematical algorithms programmed. It is the science that merge between theory and technology to produce an image with lines, or an number of composed shapes. Computer graphics image is made up of number of pixels. **Pixel** is an abbreviation of (picture element) it is the smallest addressable graphical unit represented on the computer screen . CG developed since the last 20 years. Computer is information processing machine and communication. The computer graphics is one of the most effective and commonly used ways of communication with the user. It displays the information in the form of graphical objects such as pictures, charts, diagram and graphs. In computer graphics picture or graphics objects are presented as a collection of discrete pixels. The user can control intensity and color of pixel which decide how picture look like. The process of representing continuous picture or graphics object as a collection of discrete pixels is called **Scan Conversion**.



Photo: It is any image captured by digital camera or by mobile or with the classical camera.

Picture: is the old name of image.

Image: addressed all type of image (still or video).

Pixel: pixel can be specified by its row and column numbers. It can be simply black and white system (monochrome) or color system. In B/W system each pixel is either ON or OFF, so only one bit per pixel is needed.

-ADVANTAGES of COMPUTER GRAPHICS

1.It provides tools for producing picture of “real-world” as well as

synthetic objects such as mathematical surfaces in 4D and 3D of data that have no inherent geometry such as survey result.

2. It is capability to show moving pictures and video thus possible to produce animations with computer graphics. As well as controls the animation by adjusting the speed, portion of picture in view the amount of detail shown and so on.

3.CG provides tools called motion dynamics. In which user can move objects as well as observes as per requirement for example walk throw made by builder to show flat interior and surrounding.

4.It provides facility called update dynamics. With this we can change the shape color and other properties of object.

5.recently the development of digital signal processing and audio synthesis chip provide the interface between the sound with graphics.

-APPLICATION OF COMPUTER GRAPHICS

Computer Graphics has various applications, some of which are listed below

1. **Computer graphics user interfaces *GUIs*** – A graphic, mouse-oriented paradigm which allows the user to interact with a computer.
2. **Business presentation graphics** – "A picture is worth a thousand words".
3. **Cartography** – Drawing maps.
4. **Weather Maps** – Real-time mapping, symbolic representations.
5. **Satellite Imaging** – Geodesic images.
6. **Photo Enhancement** – Sharpening blurred photos.
7. **Medical imaging** – MRIs, CAT scans, etc. - Non-invasive internal examination.
8. **Engineering drawings** – mechanical, electrical, civil, etc. - Replacing the blueprints of the past.
9. **Typography** – The use of character images in publishing - replacing the hard type of the past.
10. **Architecture** – Construction plans, exterior sketches - replacing the blueprints and hand drawings of the past.
11. **Art** – Computers provide a new medium for artists.
12. **Training** – Flight simulators, computer aided instruction, etc.

13. **Entertainment** – Movies and games.
14. **Simulation and modeling** – Replacing physical modeling and enactments
15. **Image Enhancement** - Sharpening blurred photos.

-COORDINATE REPRESENTATIONS

1. Cartesian coordinate specifications were used in Computer Graphics.
2. Any other type of coordinate values for a picture are must be converted to Cartesian coordinate before giving input to graphics package.
3. The conversion process of the World-coordinates description of the scene to one or more output device display are referred to as “**Device Coordinates**” or “**Screen Coordinates**”.
4. Generally a graphic system first converts the world-coordinates position to normalized device coordinates. In the range from 0 to 1 before final conversion to specific device coordinates.

-DISPLAY DEVICES

Display devices are also known as output devices. Most commonly used output device in a graphics system is a video monitor.

1-CRT DISPLAY

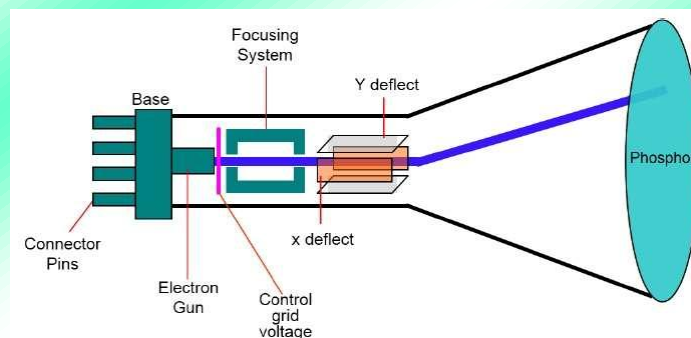


Figure 1 Cathode ray tube

-Cathode Ray Tube

It contain tube of glass, with a big end represents the screen coated inside by phosphor layer, while the other end essentially contains the

Electronic gun and Deflection Coils.

The Electronic gun contains parts as follows:

- Heating Element
- Cathode
- Control Grid
- Acceleration Anode
- Focusing Grid.

CRT based on the physical concept that the phosphor produce a light if its strike with Electrons have velocity and moment that affects the phosphor electron to speed and free to give the light. CRT It is an evacuated glass tube with an inner side phosphor coated screen.

1. The electron gun emits a beam of electrons (cathode rays).
2. The electron beam passes through focusing and deflection systems that direct it towards specified positions on the phosphor-coated screen.
3. When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam.
4. It redraws the picture by directing the electron beam back over the same screen points quickly. There are two techniques used for producing images on the CRT screen:

-Vector scan/Random Scan Display

it directly traces out only the desired lines on CRT based on equation stored at the computer memory. To draw a line between point p1 & p2 we directly drive the beam deflection circuitry which focus beam directly from point p1 to p2.

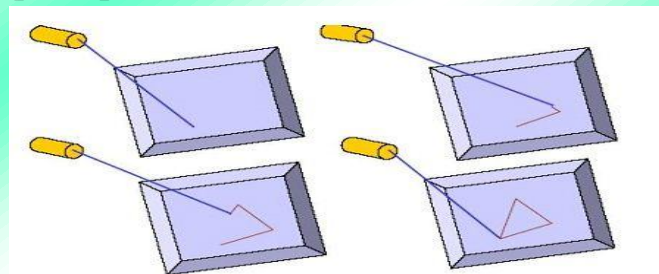


Figure 2 Vector scan

-Raster Scan Display

The display image is stored in the form of 1's and 0's in the refresh buffer. The video controller reads this refresh buffer and produces the

actual image on screen. It will scan one line at a time from top to bottom & then back to the top.

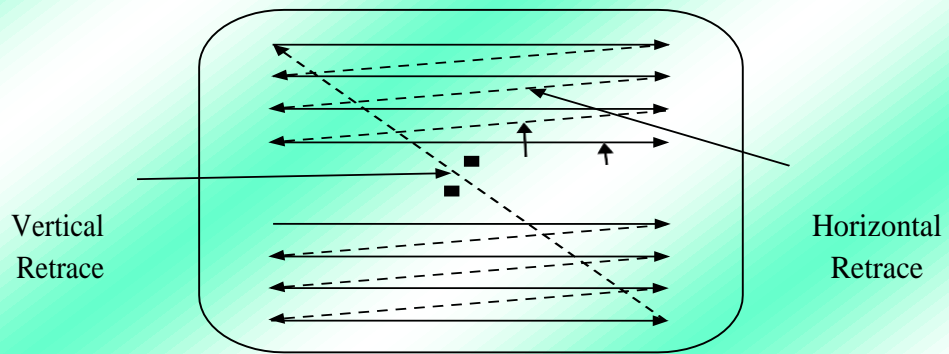
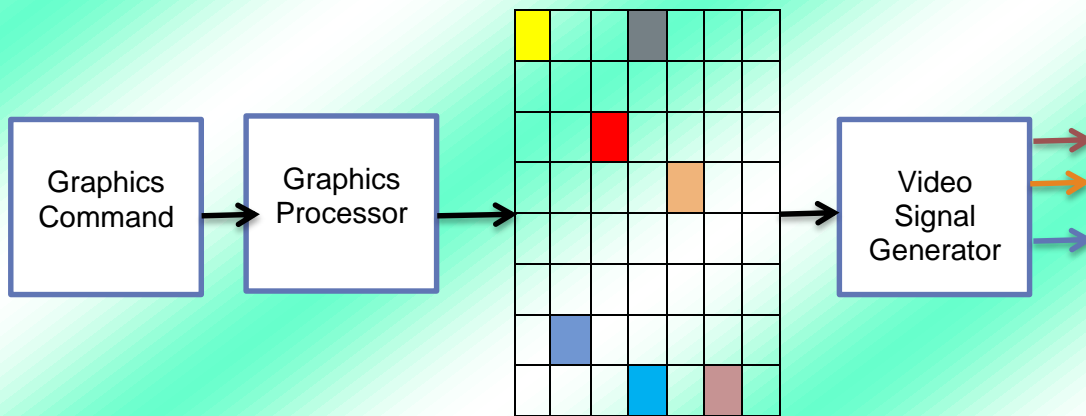


Figure 3 raster scan CRT



Frame buffer

Figure 4 Frame buffer

Frame Buffer is a special area of memory dedicated to graphics only used in raster image. The Frame buffer holds set of intensity values for all the screen points. These values are retrieved from frame buffer and display on screen one row at a time. Additional bits are required when color and intensity variations can be displayed up to 24-bits per pixel are included in true color display systems. For monochrome system with one bit per pixel the frame buffer is commonly called a **Bitmap**. And for systems with multiple bits per pixel, the frame buffer is often referred as a **Pix map**.

-ADVANTAGES OF CRT DISPLAY

- 1.Low cost.
- 2.Low weight.
- 3.High response and high resolution.
- 4.High flame and wide view angle.
- 5.Flat screen with clear image, clear colors and high resolution..
- 6.Authenticated technique and easy, expensive repairs.

-DISADVANTAGES OF CRT DISPLAY

1. The view angle is relatively narrow.
2. Big size, huge weight with the screen size.
3. Energy consumed.
4. Electromagnetic rays surrounds the screen.
- 5.

of the CRT screen works with analogue signal and rarely with digital one.

Most

LECTURE 2 : COMPUTER GRAPHICS INTRODUCTION

2-COLOR CRT DISPLAY

CRT monitors displays color pictures by using a combination of phosphors that emit different colored light. Red , Green and Blue. It contain 3 Electronic gun corresponds to each color. It produces range of colors by combining the light emitted by different phosphors with different velocity. There are two basic techniques for color display:

- Beam-penetration technique/ Stripe Pattern.
- Shadow-mask technique / Delta mask.

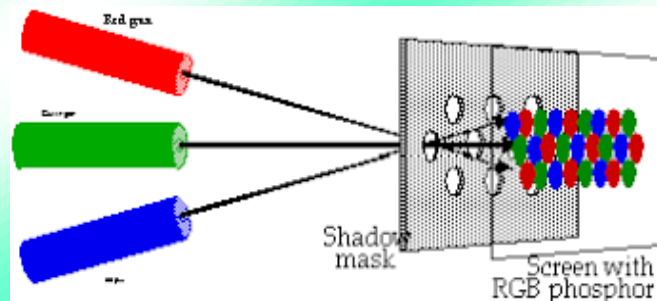


Figure 1 Color CRT Display

1-BEAM-PENETRATION TECHNIQUE

1. This technique is used with random scan monitors.
2. Colored CRT is coated with two phosphor layers usually red and green. The outer layer of red and inner layer of green phosphor.
3. It is a low cost technique to produce color in random scan monitors.
4. It can display only Red, Green ,Orange and yellow colors.
5. Quality of picture is not good compared to other techniques.

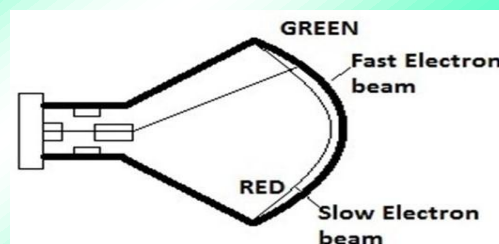


Figure 2 Beam-penetration CRT

2-SHADOW-MASK TECHNIQUE

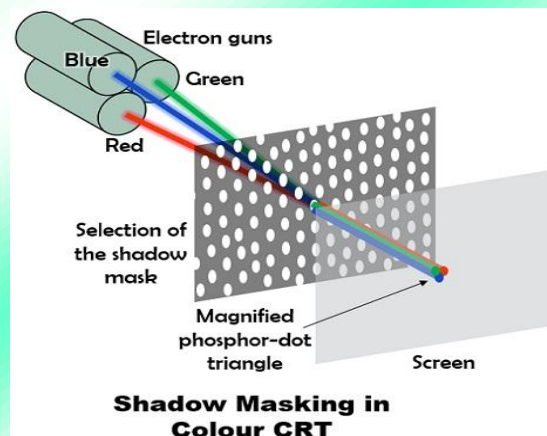


Figure 3 Shadow-mask CRT

1. It produces wide range of colors as compared to beam-penetration technique.
2. This technique is generally used in raster scan displays. Including color TV.
3. In this technique CRT has three phosphor color dots at each pixel position. One dot for red, one for green and one for blue light. This is commonly known as **Dot Triangle**.
4. A shadow mask grid just behind the phosphor coated screen. The shadow mask grid consists of series of holes aligned with the phosphor dot pattern.
5. By changing the intensity of the three electron beams we can obtain different colors in the shadow mask CRT.

-ADVANTAGES

- 1- Produce accurate images
- 2- Various colors and shadows scenes production.

-DISADVANTAGES

- 1- low resolution
- 2- expensive
- 3- electron beam directed to whole screen

- **Difference between Beam Penetration and Shadow Mask method.**

	Beam Penetration method	Shadow Mask method
Usage	used with Random Scan System to display color.	Used With Raster Scan System to display color.
Colors	It can displays Only four colors i.e. Red , Green, Orange and Yellow.	it can display Millions of colors.
Color Dependency	Less colors are available because the colors in Beam Penetration depends on the speed of the electron beam.	Millions of colors are available because the colors in Shadow Mask depends on the type of the ray.
Cost	It is Less Expensive as compared to Shadow Mask.	It is More Expensive than other methods.
Picture Quality	Quality of picture is not so good i.e. Poor with Beam Penetration Method.	Shadow Mask gives realism in picture with shadow effect and millions of color.
Resolution	High Resolution.	Low Resolution.
Criteria	Color display depends on how far electron excites outer Red layer and then Green layer.	There are no such criteria for producing colors. It is used in computers, in color TV etc.

1. MDA (Monochrome DISPLAY ADAPTER): Monitor with text display and the kinds of colors, such as red, green, or blue. with a resolution of 720 x 350
2. CGA (COLOR GRAPHICS ADAPTER): Color monitor first manufactured by IBM in 1981.It recognize text and graphics. Resolution chart 320 x 200 pixel. Character has a pixel density of 7 x 9 points.
3. EGA (ENHANCED GRAPHICS ADAPTER): Color monitor, manufactured in 1981, coinciding with the release of IBM PC AT. Resolution chart 640 x 340 pixel. Character has a pixel density of 7 x 9 points.
4. VGA (MONITOR GRAPHICS ARRAY): Color monitor, manufactured by IBM in 1986 along with IBM Resolution 256 has a 720 x 400 pixel. Character has a density of 9 x 16 dots.

LECTURE 3 : FLAT PANEL DISPLAY

-FLAT PANEL DISPLAY

The term flat panel display recently used that refers to a class of video device that have reduced volume, weight & power requirement compared to a CRT. As flat panel display is thinner than CRTs, we can hang them on walls or wear on our wrists. Since we can even write on some flat panel displays they will soon be available as pocket notepads. Flat panel is classified into two types:

1. **Emissive displays:** the emissive display or **emitters** are devices that convert electrical energy into light. Such as Plasma panel and light emitting diodes.
2. **Non emissive displays:** non emissive display or **non emitters** use optical effects to convert sunlight or light from some other source into graphics patterns. Such as LCD (Liquid Crystal Display).

1-PLASMA PANELS DISPLAYS

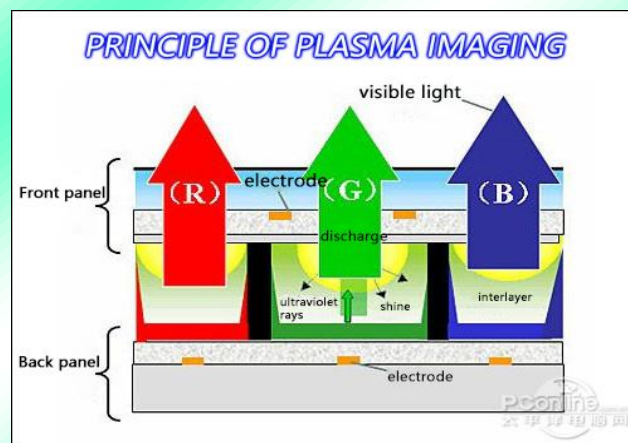


Figure 1 Basic design of a plasma-panel display device

1. It is constructed by a thousands of vacuumed glass tube filled with mixture of (Ne, He, Xe) gases.
2. The tubes are colored by red, green and blue phosphor and positioned between the front and end glass layer. Each tube is connected from the two ends, the front for display and the end for addressing. The gas is ion through the electrical voltage directed to both ends so electrical discharge is happened at the cell. That affects the gas to be ion and converted to Plasma and UV rays are emitted.

3. Picture definition is stored in a refresh buffer and the firing voltages are applied to refresh the pixel positions, 60 times per second.

4. Alternating current methods are used to provide faster application of firing voltages and thus brighter displays.

-ADVANTAGES OF PLASMA DISPLAY

1. Manufactured with huge size that enabled to be used on theater, stadium and parks .

2. Low weight and small thickness which help the user to hang it on the wall.

3. Flat screen with clear image, clear colors and high resolution.

4. Without Electromagnetic rays surrounded the screen, so audio system could be used without rays affection.

5. Clear vision angle approximate to 160.

-DISADVANTAGES OF PLASMA DISPLAY

1. The Plasma screen couldn't be manufactured with small size for the reason that it is impossible to eliminate the distance between pixel pitch also the tube glass should be thick for technical reason .

2. The Plasma screen has default age because the phosphor layer coated the tube inside is decayed by gas particles hit.

2-ORGANIC LIGHT EMITTING DIODE (OLED/LED)

1. OLED is a very new display depends on a very modern technique named Organic Light Emitting Diode. The diode is produced from a very thin organic material known as organic Polymers, its main element is Carbon with thickness about 500 nanometer.

2. In this display a matrix of multi-color light emitting diode is arranged to form the pixel position in the display. And the picture definition is stored in refresh buffer.

3. Similar to scan line refreshing of CRT information is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light pattern on the display. OLED are classified into many categories the most important are follows:

- **Passive matrix OLED:** : it is used to manufacture the small screen such

mobile, pad and video game screen.

- **Active matrix OLED:** it is used to manufactured a big screen such TV and computer screen.
- **Transparent OLED:** it is transparent used to manufactured the car and airplane wind screen.
- **Foldable OLED:** it is resistant to broken so the used with lab top.
- **White OLED:** it is used with the modern light systems due to its light weight , little energy consumed and a very high flash.

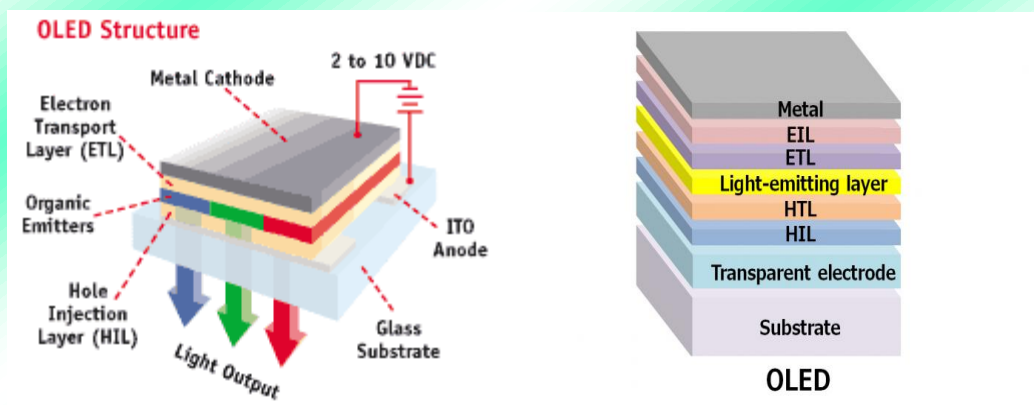


Figure 2 OLED Architecture

-ADVANTAGE OF OLED

- 1.The plastic, organic layers of an OLED are **thinner, lighter and more flexible**.
- 2.OLEDs are **brighter** than LEDs.
- 3.OLEDs do not require backlighting while OLEDs generate light themselves. Because OLEDs do not require backlighting, they **consume much less power** This is especially important for battery-operated devices such as cell phones.
- 4.OLEDs are easier to produce and can be made to larger sizes. Because OLEDs are essentially plastics, they can be made into large, thin sheets.
- 5.OLEDs have **large fields of view**, about 170 degrees. They have a much wider viewing range.

-DISADVANTAGE OF OLED

1.Lifetime - While red and green OLED films have longer lifetimes (46,000 to 230,000 hours), blue organics currently have much shorter lifetimes (up to around 14,000 hours).

2.Manufacturing - Manufacturing processes are expensive right now.

3.Water - Water can easily damage OLEDs.

3-LIQUID CRYSTAL DISPLAY (LCD)

- 1, It is generally used in small system such as calculator and portable laptop.
- 2.This non emissive device produce picture by passing polarized light from the surrounding or from an internal light source through liquid crystal material that can be aligned to either block or transmit the light.
3. The intersection of two conductors defines a pixel position.
4. In the ON state polarized light passing through material is twisted so that it will pass through the opposite polarizer. In the OFF state it will reflect back towards source.
5. This type of flat panel device is referred to as a passive matrix LCD.
6. In active matrix LCD transistors are used at each (x, y) grid point.
7. Transistor cause crystal to change their state quickly and also to control degree to which the state has been changed.
8. Transistor can also serve as a memory for the state until it is changed. So transistor make cell ON for all time giving brighter display then it would be if it had to be refresh periodically.

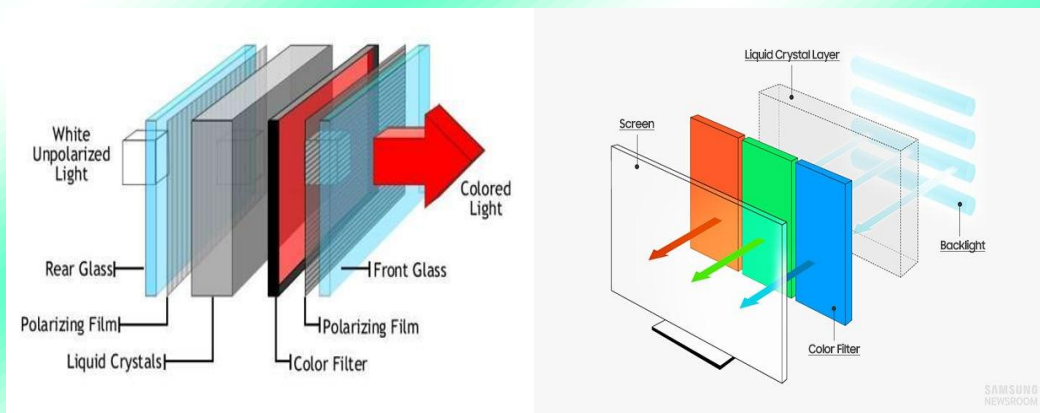


Figure 3 LCD display

-ADVANTAGES OF LCD DISPLAY

1. Low cost.
2. Low weight.
3. Small size
4. Low power consumption.
5. Flat screen with clear image, clear colors and high resolution.
6. Without Electromagnetic rays surrounding the screen.
7. Big screen that can be hanged on the wall.
8. Viewing Angle : Up to 165°, Picture suffers from the side.

-DISADVANTAGES OF LCD DISPLAY

1. The aspect ratio and resolution are fixed. 2. Lower contrast than CRTs due to a poor black-level.
3. Slow response times and scan rate conversion result in severe motion artifacts and image degradation for moving or rapidly changing images.

LECTURE 4 : THREE DIMENSIONAL VIEWING DEVICES

The 3D images are displayed using a technique that reflects a CRT image from a silver flexible mirror. The 3D Graphics is one of the most important and sophisticated Computer graphics branches. It offers the user a 3Dimensional Virtual environment and figures can be in an interactive environment or just watching a movie or image.

-APPLICATION OF 3D VIEWING DEVICES

1. Entertainment and games.
2. Cinema movies and Visual tricks.
3. Education and Explanation Drawings and figures.
4. Virtual reality application.
5. Commercial announcements.
6. Web and Architecture: 3d design software can be used to develop ideas for new buildings or additional extensions to a building. The same can be said for web based design, a website can be built from scratch just like a designer would build a building design in a programmer.
7. Building design- In architectural design the producer must be accurate with his development, and also has to take in consideration of the laws of physics.
8. Web based design- For example Google is a plain professional looking site, the colors work and the site is used daily by millions of people who are always happy with the way it looks. The same can be said with Facebook although people complain about the changes that are made they eventually endure and get used to the new design. However there are bad web designers out there, terrible site design does not sell the viewers, they will probably leave the site after viewing the front page. Below is a prime example of awful website design, viewers would want to leave instantly.

-THE SOFTWARE USED WITH COMPUTER GRAPHICS

The CG requires a program to perform its application. These programs are classified into many types the most important are as follows:

1-Virtual reality

Virtual reality is the system which produce images in such a way that we feel that our surrounding is what we are set in display devices but in actually it does not. E.g. You'll probably never go to Mars, swim with dolphins or run an Olympic 100 meters. But if **virtual reality** ever lives up to its promise, you might be able to do all these things—and many more—without even leaving your home. Unlike *real* reality (the actual world in which we live), virtual reality means simulating bits of our world (or completely imaginary worlds) using high-performance computers and sensory equipment, like headsets and gloves. Apart from games and entertainment, it's long been used for training airline pilots and surgeons and for helping scientists to figure out complex problems such as the structure of protein molecules. How does it work? Let's take a closer look!



Figure 1 Virtual reality

Virtual reality (VR) means experiencing things through our computers that don't really exist. From that simple definition, the idea doesn't sound especially new. When you look at an amazing for example, you're experiencing the sites and sounds of Italy as it was about 250 years ago—so that's a kind of virtual reality. In the same way, if you listen to ambient instrumental or classical music with your eyes closed, and start dreaming about things, isn't that an example of virtual reality. If we're going to understand why books, movies, paintings, and pieces of music aren't the same thing as virtual reality, we need to define VR fairly clearly.

2-Computer graphics animation and cartoon

Graphic animation is a variation of stop motion (and possibly more conceptually associated with traditional flat animation and paper

drawing animation, but still technically qualifying as stop motion) consisting of the animation of photographs (in whole or in parts) and other non-drawn flat visual graphic material, such as newspaper and magazine clippings. In its simplest form, Graphic "animation" can take the form of the animation camera merely panning up and down and/or across individual photographs, one at a time, (filmed frame-by-frame, and hence, "animated") without changing the photographs from frame to frame.

Cartoon animators work largely in the motion picture and advertising industries. Their general function is to create drawings, either by hand or with electronic aids, and then use computers to produce the chain of images that shapes the animated film or special effect. They may work with a team of professionals to develop films, webpages, promotional spots and computer artwork. Cartoon animators create animated narrative sequences by combining artistic skills with talents in many areas, such as comedy, drama, advertising and computer modeling. Professional cartoon animators usually work in teams where they participate in the storyboard, typesetting and editing process of animation.



Figure 2 Animation and Cartoon

3-The image processing programs

Are those program and applications used with image captured by mobile or camera. These programs permit to change in the form of the image and manipulate it such as change its brightness, contrast, rotate, merge and change image colors. As well as the application which permit the user to free hand use the program with color and edit tools to draw his graph . Also the programs based on CAD program to design his map and construction.

-GRAPHICS INPUT DEVICES

- 1. Keyboards** are used as entering text strings. It is efficient devices for inputting such a non-graphics data as picture label.
- 2.Mouse** is small size hand-held box used to position screen cursor. Wheel or roller or optical sensor is directing pointer on the according to movement of mouse.
- 3.Trackball and Space ball** is ball that can be rotated with the finger or palm of the hand to produce cursor movement.
- 4.Joysticks** consists of small vertical lever mounted on a base that is used to steer the screen cursor around.
- 5. Data glove** is used to grasp virtual objects. The glove is constructed with series of sensors that detect hand and figure motions.
- 6.Digitizer** is common device for drawing painting or interactively selecting coordinates position on an object.
- 7.Image Scanner** Image Scanner scan drawing, graph, color, & black and white photos or text and can stored for computer processing by passing an optical scanning mechanism over the information to be stored.
- 8.Touch Panels** As name suggest Touch Panels allow displaying objects or screen-position to be selected with the touch or finger.
- 9.Light pens** Light pens are pencil-shaped device used to select positions by detecting light coming from points on the CRT screen.
- 10.Voice systems** It is used to accept voice command in some graphics workstations. It is used to initiate graphics operations.

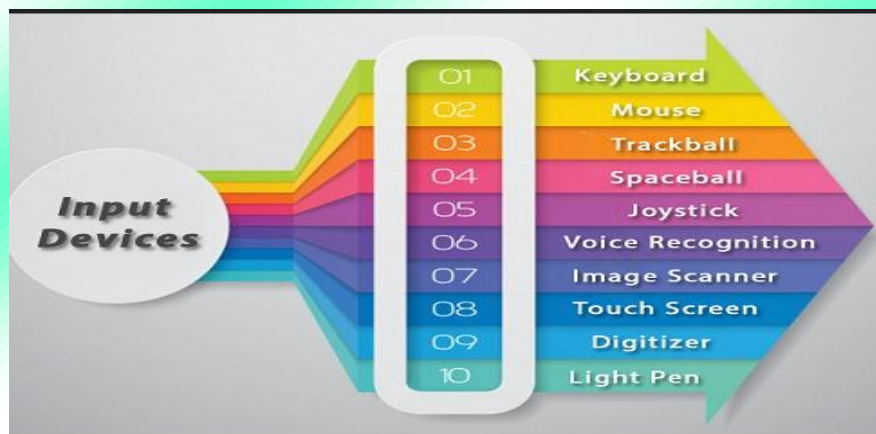


Figure 3 Graphics input device

LECTURE5 : COLORS FUNDEMENTAL

-COLOR DEPTH

Alternatively referred to as **pixel depth**, **color depth** refers to the number of bits per pixel on a computer monitor to represent a specific color. The more bits per pixel, the higher color variety and quality of the monitor. The first graphics cards and monitors supported 1-bit color, which was monochrome (most commonly black and white), for early computers like the Apple Macintosh and Atari ST. Today, most computers support at least 32-bit color, which allows for up to 16.7 million colors. Windows 7 introduced support for 48-bit color, assuming the computer's video card supports this color depth.

-EVOLUTION OF COLOR DEPTH

As technology and available system resources have increased, so has the color depth. Below is a listing of all the different color depths over the history of computers. Color depth is equal to(2 power n), where n is equal to the number of bit per pixel.

- 1-bit (2^1 or 2 colors) - Monochrome displays.
- 2-bit (2^2 or 4 colors) - CGA displays.
- 4-bit (2^4 or 16 colors) - EGA displays.
- 8-bit (2^8 or 256 colors) - VGA displays.
- 16-bit (2^{16} or 65,536 colors) - XGA displays.
- 24-bit (2^{24} or 16,777,216 colors) - SVGA displays.
- 32-bit (16,777,216 colors + Alpha channel (2^{32} or 4,294,967,296 color combinations))
- 48-bit (2^{48} or 281,474,976,710,656 colors)

-COLOR LOOKUP TABLES

Lookup tables are simply arrays of numbers where an input number is treated as an address (or position) in the array. The table then outputs the number stored at that address. Any function graph where each x value (address) has a corresponding y value (output) can be used as a lookup table. A good way to save storage space for such an image is to include a Lookup Table(or LUT) in the file header. Lookup tables are also referred to as Color Maps or Color Palettes. Color image files that use lookup tables are called Indexed Color images. Image files that do not use a lookup

table and store individual pixel data with full precision are called True Color images. Because the lookup table is distinct from the pixel intensity data the way image data is displayed can be easily and conveniently changed by manipulation of the look up table without having to adjust the individual pixel intensity or color data .

1- PSEUDO-COLOR FRAME BUFFER

The pseudo-color frame buffer allows representation of color images. The storage scheme is identical to the grey scale frame buffer. However the pixel values do not represent shades of grey. Instead each possible value (0–255) represents a particular color; more specifically, an index into a list of 256 different colors maintained by the video hardware. The colors themselves are stored in a “Color Lookup Table” (CLUT) which is essentially a map< color index, color >i.e. a table indexed with an integer key (0–255) storing a value that represents color. In alternative term of the CLUT is sometimes called a palette. A color look up table can also be used to display gray scale image data as a ‘pseudo color’ image Lookup tables are also used to adjust the output of display hardware. the simplest form of frame buffer is the grey scale frame buffer. Grey scale buffers encodes pixels using various shades of grey. In common implementations, pixels are encoded as an unsigned integer using 8 bits (1 byte) and so can represent $2^8 = 256$ different shades of grey. Usually black is represented by value 0, and white by value 255. A mid-intensity grey pixel has value 128. Consequently an image of width W pixels and height H pixels requires $W \times H$ bytes of memory for its frame buffer. A typical computer graphics card (display adapter) includes a built-in lookup table that adjusts the raw display data to suit the specific monitor attached to the card. Monitor calibration systems adjust these lookup tables in order to produce a defined monitor light output (color and brightness) as measured by a photometer placed on the monitor face.

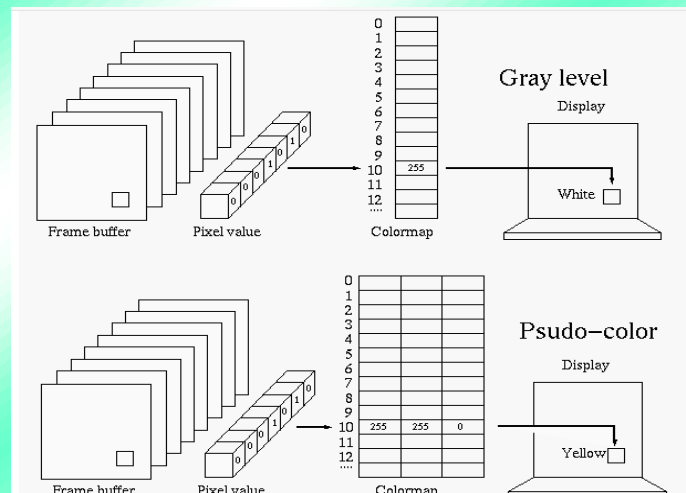


Figure 1 Pseudo color LU

2-TRUE-COLOR FRAME BUFFER

The true-color frame-buffer also represents color images, but does not use a CLUT. The RGB color value for each pixel is stored directly within the frame buffer. So, if we use 8bits to represent each Red, Green and Blue component, we will require 24 bits (3 bytes) of storage per pixel. As with the other types of frame buffer, pixels are stored in left-right, then top-bottom order. So in our 24 bit color example, pixel (0,0) would be stored at buffer locations 0, 1 and 2. The advantages of the true-color buffer complement the disadvantages of the pseudo-color buffer. We can represent all 16 million colors at once in an image (given a large enough image!), but our image takes 3 times as much storage as the pseudo-color buffer. The image would also take longer to update (3 times as many memory writes) which should be taken under consideration on resource constrained platforms (e.g. if writing a video codec on a mobile phone). For example, Red and Green light mix to produce Yellow light. Therefore the value stored in the CLUT for each color is a triple (R, G, B) denoting the quantity (intensity) of Red, Green and Blue light in the mix. Each element of the triple is 8bit i.e. has range (0–255) in common implementations. The earliest color displays employed pseudo-color frame buffers. This is because memory was expensive and color images could be represented at identical cost to grey scale images (plus a small storage overhead for the CLUT). The obvious disadvantage of a pseudo-color frame buffer is that only a limited number of colors may be displayed at any one time (i.e. 256 colors). However the color range (we say gamut) of the display is $28 \times 28 \times 28 = 224 = 16,777,216$ colors.

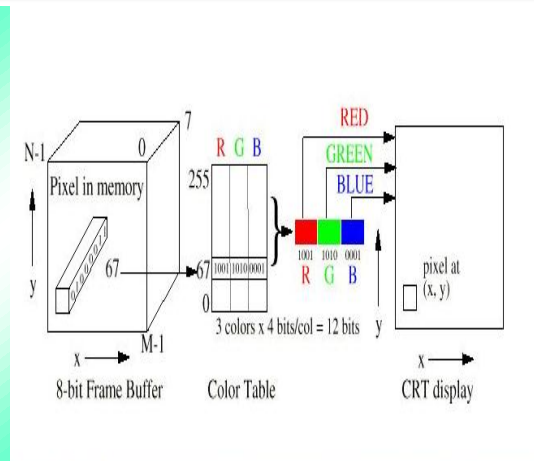
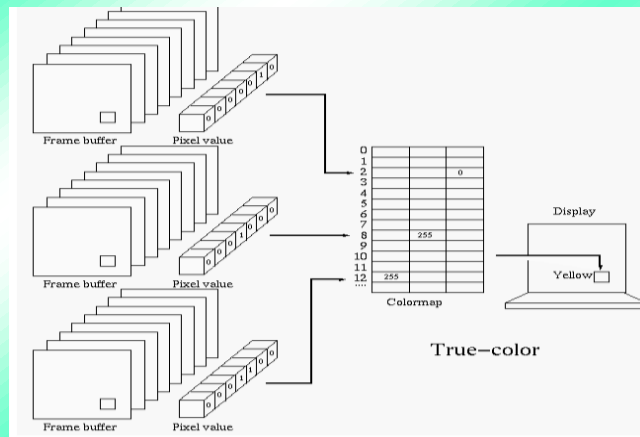


Figure 2 True color LUT

LECTURE6 : IMAGE AND CG FILE FORMAT

-IMAGE FILE FORMAT

1. Image files for the display on the computer screen and printing through different printers:

Images that are saved for the purpose of displaying them in the form of photo albums on the computer screen such as family photos, or images added to other files such as writing editor or PowerPoint presentations, or image files that are saved from digital cameras after they are captured, Above the average and the dimensions of the image are also medium, but the size of the image file is preferred to be average (Not too large, not too small). Large files may take up unnecessary space, and very small files may be of poor resolution, so the files of these images should be compressed bodies such as JPEG, It is popular and used to save images on the computer screen and is saved in RGB.

2. Image files used for online viewing and exchange via e-mail:

Many images are placed on the web or as an attachment with emails and then displayed on the screen. For this purpose it is best to use small files sent over the Internet quickly. JPEG is the most common body in these cases to save images, but other bodies have been created such as GIF, which achieves other uses such as animation and transparency on the image.

1. Image files used with image processing and editing software:

Producers continuously provide new image processing programs or develop existing applications, and note that they have a tendency to establish image files for their applications known as native formats. The goal of the new bodies is to understand the new procedures and capabilities of the programs and to excel at the competitors. However, local bodies cause many difficult problems especially for those who wish to process images using more than one application Or seeks to transfer images to others. Local authorities are often read only by their program and can not be downloaded through other programs. The types of image files are as follows:

1-GIF standard: Limited to 8-bit (256) color images only, which, while producing acceptable color images, is best suited for images with few distinctive colors (e.g., graphics or drawing). GIF The Graphic Interchange Format(GIF) is ideal for storage of simple images containing few distinct colors and very limited tonal detail. Only 256 different colors may be stored and these are encoded in a lookup table.

GIF actually comes in two types:

1. **GIF87a:** The original specification.
2. **GIF89a:** The later version. Supports simple animation via a Graphics Control Extension block in the data, provides simple control over delay time, a transparency index.



Figure 1 GIF image

2-JPEG: The most important current standard for image compression.

- 1.The human vision system has some specific limitations and JPEG takes advantage of these to achieve high rates of compression.
- 2.JPEG allows the user to set a desired level of quality, or compression ratio (input divided by output).

3-PNG: standing for **Portable Network Graphics** meant to supersede the GIF standard, and extends it in important ways. Special features of PNG include:

1. Support for up to 48 bits of color information | a large increase.
2. Files may contain special information for correct display of color images, as well as other information form to be used as control of transparency.
3. The display progressively displays pixels in a 2-dimensional fashion by showing a few pixels at a time over seven passes through

each 8x8 block of an image.



Figure 2 PNG image

4- TIFF: stands for **Tagged Image File Format** can store many different types of image. The TIFF format was developed by the Aldus Corporation in the 1980's and was later supported by Microsoft.

1. 1-bit, Gray scale, 8-bit color, 24-bit RGB, etc.
2. TIFF was originally a lossless format.



Figure 3 Tiff image

5- DICOM: Most medical imaging systems archive and transmit image data in DICOM(Digital Imaging and Communications in Medicine) format. The DICOM standard is designed to enable efficient exchange of radiologic information (images, patient information, scheduling information, treatment planning, etc.) independent of modality and device manufacturer. The DICOM standard provides for lossless and lossy JPEG compression, and other lossless compression formats. Multiple frames, such as the contiguous slice images of a 3D data set, can be stored in a single DICOM file. An important feature of the DICOM format is its ability to store pixel intensity data with precision of 8, 12, 16, or 32 bits according to the measurement precision of the imaging system.

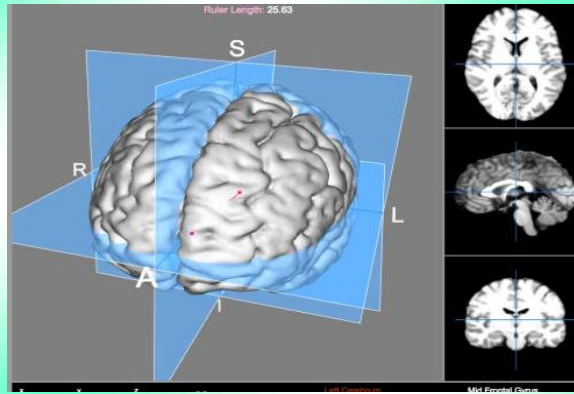


Figure 4 DICOM image

6- EXIF: (Exchange Image File) is an image format for digital cameras:

1. Compressed EXIF files use the baseline JPEG format.
2. A variety of tags (many more than in TIFF) are available to facilitate higher quality printing, since information about the camera and picture-taking conditions (flash, exposure, light source, white balance, type of scene, etc.) can be stored and used by printers for possible color correction algorithms.
3. The EXIF standard also includes specification of file format for audio that accompanies digital images. As well, it also supports tags for information needed for conversion to Flash Pix.

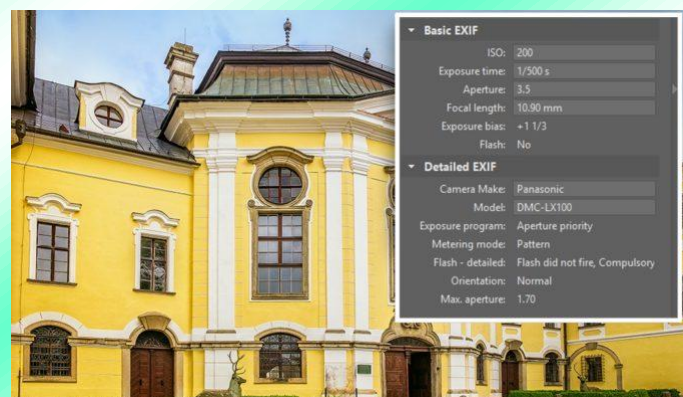


Figure 5 EXIF image

7- PS stands for Postscript is an important language for typesetting, and many high-end printers have a Postscript interpreter built into them. Postscript is a vector-based picture language, rather than pixel-

based: page element definitions are essentially in terms of vectors.

1. Postscript includes text as well as vector/structured graphics.
2. GL bit-mapped images can be included in output files.
3. Encapsulated Postscript files add some additional information for inclusion of Postscript file document.

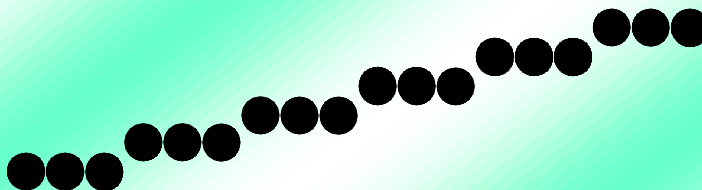
LECTURE 7 : LINE GENERATION ALGORITHM

1-. PLOTTING POINT

Point plotting is done by converting a single coordinate position furnished by an application program into appropriate operations for the output device in use. Pixel in programming language is accessed by a positive integer (x,y) coordinate pair, the value x start at origin 0, and increase from left to right, the y value start at 0 increase from top to bottom. Pixel position will referenced according to scan-line number and column number which is illustrated by following figure. To load the specified color into the frame buffer at a particular position, we will assume we have available low-level procedure of the form `putpixel(x, y)`.

2-LINE DRAWING ALGORITHMS

Line segment is displayed by turning on a string of adjacent pixels. In order to draw a line, it is necessary to determine which pixels lie nearest the line and provide the best approximation to the desired line. The accuracy and quality of the displayed line depends on the resolution of the display device. High resolution displays draw lines that look straight and continues and start and end accurately. Lower resolution displays may draw lines with gaps. A Discrete coordinate positions along the line path are calculated from the equation of the line. Reading from the frame buffer, the video controller then plots the screen pixels. The output device is then directed to fill in those positions between the end points with some color. For some device such as a pen plotter or random scan display, a straight line can be drawn smoothly from one end point to other. For example line position of (10.36, 25.87) would be converted to pixel position (10, 26). This rounding of coordinate values to integers causes lines to be displayed with a stair step appearance as represented in the following figure.



- Draw Horizontal Line

To draw horizontal line, the y value is fixed and the value x varies. The following codes draw horizontal line from (xstart , y) to (xend , y).

```
For x = xstart:xend putpixel ( x , y , RED ) ;
```

```
If xstart > xend
```


- Draw Vertical Line

To draw vertical line, the x value is fixed and y value varies. The following codes draw a vertical line from (x , ystart)to(x , yend).

```
For y = ystart:yend do  
putpixel ( x , y ,GREEN) ;  
If ystart > yend
```

- Draw Diagonal Line

To draw a diagonal line with a slope equal to +1 , we need only repeatedly increment by one unit both the x and y values from the starting to the ending pixels. The following codes draw a diagonal line:

```
x := xstart ;  
y := ystart ;  
i := 0 ;  
while ( x + i ) <= xend do  
begin  
putpixel ( x + i , y + i , white ) ;  
i := i + 1 ;  
end ;
```

To draw a line with a slope -1 , replace y+i by y-i in the code.

1-THE CARTESIAN SLOPE-INTERCEPT EQUATION

The Cartesian slope-intercept equation for a straight line is $Y=mx+b$ With m representing the slope of the line and b as the y intercept. Given that the two endpoints of the line segment are specified at positions(x1,y1) and (x2,y2) we can determine values for the slope m and y intercept b with the following calculations, $m=(y2-y1)/(x2-x1)$ so $b=y1-m*x1$.

Example: Draw a line start with P1(5,7) and end with P2(8,9).

$$m = \frac{\Delta y}{\Delta x} = \frac{9-7}{8-5} = \frac{2}{3} = 0.6$$

$$b = y_1 - mx_1 = 7 - 0.6 * 5 = 4$$

$$5 \leq x \leq 8$$

$$y_1 < y_2 \Rightarrow 7 < 8$$

$$y_{i+1} = mx + b$$

x	$y_{i+1} = mx + b$	point
5	7	(5,7)
6	$y = 0.6 * 6 + 4 = 7.6 \approx 8$	(6,8)
7	$y = 0.6 * 7 + 4 = 8$	(7,8)
8	$y = 0.6 * 8 + 4 = 8.8 \approx 9$	(8,9)

LECTURES : LINE GENERATION ALGORITHM

-DIGITAL DIFFERENTIAL ANALYZER (DDA)

Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.

A Line Drawing Algorithm (Case a: $m \leq 1$)

$x=x_0$; $y=y_0$; $y=y+m$

$x = x_0 + 1$

$y = y_0 + 1 * m$

Illuminate pixel (x , $\text{round}(y)$)

$x = x + 1$

$y = y + 1 * m$

Illuminate pixel (x , $\text{round}(y)$)

...

Until $x == x_1$

(x_1, y_1)

$x = x_0$, $y = y_0$

Illuminate pixel (x , $\text{round}(y)$)

▪ **if $|m| \leq 1$**

If $|m| \leq 1$ and $y_1 < y_2$ (increment) then $y_{i+1} = y_i + m$

If $|m| \leq 1$ and $y_1 > y_2$ (decrement) then $y_{i+1} = y_i - m$

Example 1 : Draw a line start with P1(5,7) and end with P2(8,9) with DDA

$$m = \frac{\Delta y}{\Delta x} = \frac{9-7}{8-5} = \frac{2}{3} = 0.6 < 1$$

$$5 \leq x \leq 8$$

$$y_1 < y_2 \Rightarrow 7 < 8$$

$$y_{i+1} = y_i + m$$

x	$y_{i+1} = y_i + m$	point
5	7	(5,7)
6	$y=7+0.6=7.6$	(6,8)
7	$y=7.6+0.6=8.2$	(7,8)
8	$y=8.2+0.6=8.8$	(8,9)

Example 2: Draw a line start with P1(5,7) and end with P2(8,5) with DDA

$$m = \frac{\Delta y}{\Delta x} = \frac{5-7}{8-5} = \frac{-2}{3} = -0.6$$

$$|m| < 1$$

$$5 \leq x \leq 8 \text{ increment by } 1$$

x	$y_{i+1} = y_i - m$	point
5	7	(5,7)
6	$y=7-0.6=6.4$	(6,6)
7	$y=6.4-0.6=5.8$	(7,6)
8	$y=5.8-0.6=5.2$	(8,5)

$$y_1 > y_2 \text{ (decrement)} \Rightarrow 7 > 5$$

$$y_{i+1} = y_i - m$$

Example 3: Draw line with p1(8,10) and p2(2,6) with DDA?

$$m = \frac{\Delta y}{\Delta x} = \frac{6-10}{2-8} = \frac{-4}{-6} = 0.6 < 1$$

$$8 \geq x \geq 2$$

$$y_1 > y_2 \text{ (decrement)} \Rightarrow 10 > 6$$

$$y_{i+1} = y_i - m$$

x	$y_{i+1} = y_i - m$	point
8	10	(8,10)
7	$y = 10 - 0.6 = 9.4$	(7,9)
6	$y = 9.4 - 0.6 = 8.8$	(6,9)
5	$y = 8.8 - 0.6 = 8.2$	(5,8)
4	$y = 8.2 - 0.6 = 7.6$	(4,8)
3	$y = 7.6 - 0.6 = 7$	(3,7)
2	$y = 7 - 0.6 = 6.4$	(2,6)

▪ if $|m| > 1$

If $|m| > 1$ and $x_1 < x_2$ (increment) then $x_{i+1} = x_i + \frac{1}{m}$

If $|m| > 1$ and $x_1 > x_2$ (decrement) then $x_{i+1} = x_i - \frac{1}{m}$

Example 4: Draw line with P1(3,2) and P2(8,10) with DDA?

$$m = \frac{\Delta y}{\Delta x} = \frac{10-2}{8-3} = \frac{8}{5} = 1.6 > 1$$

$$\frac{1}{m} = \frac{5}{8} = 0.625$$

$$2 \leq y \leq 10$$

$$x_1 < x_2 \text{ (increment)} \Rightarrow 3 < 8$$

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

y	$x_{i+1} = x_i + \frac{1}{m}$	point
2	3	(3,2)
3	$x = 3 + 0.625 = 3.625$	(4,3)
4	$x = 3.625 + 0.625 = 4.25$	(4,4)
5	$x = 4.25 + 0.625 = 4.875$	(5,5)
6	$x = 4.875 + 0.625 = 5.5$	(6,6)
7	$x = 5.5 + 0.625 = 6.125$	(6,7)
8	$x = 6.125 + 0.625 = 6.75$	(7,8)
9	$x = 6.75 + 0.625 = 7.375$	(7,9)
10	$x = 7.375 + 0.625 = 8$	(8,10)

Example 5: Draw line with P1(5,8) and P2(9,5) with DDA?

$$m = \frac{\Delta y}{\Delta x} = \frac{5-8}{9-5} = \frac{-3}{4} = |-0.75| < 1$$

$$5 \leq x \leq 9$$

$$y_1 > y_2 \text{ (decrement)} \Rightarrow 8 > 5$$

$$y_{i+1} = y_i - m$$

x	$y_{i+1} = y_i - m$	point
5	8	(5,8)
6	$y = 8 - 0.75 = 7.25$	(6,7)
7	$y = 7.25 - 0.75 = 6.5$	(7,7)
8	$y = 6.5 - 0.75 = 5.75$	(8,6)
9	$y = 5.75 - 0.75 = 5$	(9,5)

Example 6: Draw line with P1(0,0) and P2(5,5) with DDA?

$$M = 5/5 = 1$$

$$5 \geq x \geq 0$$

$$y_1 < y_2 \Rightarrow \Delta > 0$$

$$y_{i+1} = y_i + m$$

x	$y_{i+1} = y_i + m$	point
0	0	(0,0)
1	$y = 0 + 1 = 1$	(1,1)
2	$y = 1 + 1 = 2$	(2,2)
3	$y = 2 + 1 = 3$	(3,3)
4	$y = 3 + 1 = 4$	(4,4)
5	$y = 4 + 1 = 5$	(5,5)

-Advantages of DDA Algorithm

1. It is the simplest algorithm and it does not require special skill for implementation.
2. It is a faster method for calculating pixel positions than the direct use of equation $y = mx + b$.
3. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.

- Disadvantages

1. DDA algorithm is time consuming because Floating point arithmetic and it is Rounding-off.
2. The algorithm is orientation dependent. Hence end point accuracy is poor.
3. Although DDA is fast, the accumulation of round-off error in successive additions of floating point increment, can cause the calculation pixel position to be wrong from the true line path.

LECTURE9 : LINE GENERATION ALGORITHM

-BRESENHAM LINE DRAWING ALGORITHM

It is a line drawing algorithm which determines the points of a rectangular pattern of a parallel scanning lines followed by the electron beam on a television screen or computer monitor that should be selected in order to form a close approximation to a straight line between two points.

This algorithm seeks to select the optimum screen locations to represent a straight line. To accomplish this algorithm take, as an example, a line in the first quadrant (i.e. a line with slope between 0 and 1). The algorithm step by step is shown below:

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

if $|m| < 1$

- y is calculated , x increment by 1 with $x_1 \leq x \leq x_2$

$$d_i = 2dy - dx$$

$$d_{i+1} = d_i + 2(dy - dx)$$

if $d_i < 0$ then $y_i = y_{i-1}$

$$d_{i+1} = d_i + 2(dy)$$

if $|m| > 1$

- x is calculated , y increment by 1 with $y_1 \leq y \leq y_2$

$$d_i = 2dx - dy$$

if $d_i \geq 0$ then $x_i = x_{i-1} + 1$

$$d_{i+1} = d_i + 2(dx - dy)$$

if $d_i < 0$ then $x_i = x_{i-1}$

$$d_{i+1} = d_i + 2(dx)$$

if $|m| = 1$

if slope equal 1 the solution could be in either choice

Example 1: Use Bresenham Line drawing algorithm to draw a line with P1(5,8) & P2(9,11)?

$$m = \frac{\Delta y}{\Delta x} = \frac{11-8}{9-5} = \frac{3}{4} = 0.75 < 1$$

y is calculated and $x_1 \leq x \leq x_2$

$$d_i = 2dy - dx = 2 * 3 - 4 = 2 > 0$$

$$y_i = y_{i-1} + 1 = 8 + 1 = 9$$

$$d_{i+1} = d_i + 2(dy - dx) = 2 + 2(3 - 4) = 0$$

$$y_i = y_{i-1} + 1 = 9 + 1 = 10$$

$$d_{i+1} = d_i + 2(dy - dx) = 0 + 2(3 - 4) = -2 < 0$$

$$y_i = y_{i-1} = 10$$

$$d_{i+1} = d_i + 2(dy) = -2 + 2 * 3 = 4 > 0$$

$$y_i = y_{i-1} + 1 = 10 + 1 = 11$$

x	y
5	8
6	9
7	10
8	10
9	11

Example 2: Use Bresenham Line drawing algorithm to draw a line with P1(5,2) & P2(9,7)?

$$m = \frac{\Delta y}{\Delta x} = \frac{7-2}{9-5} = \frac{5}{4} = 1.25 > 1$$

$y_1 \leq y \leq y_2$, compute x

$$d_i = 2dx - dy = 2 * 4 - 5 = 3 > 0$$

$$x_i = x_{i-1} + 1 = 5 + 1 = 6$$

$$d_{i+1} = d_i + 2(dx - dy) = 3 + 2(4 - 5) = 1 > 0$$

$$x_i = x_{i-1} + 1 = 6 + 1 = 7$$

$$d_{i+1} = d_i + 2(dx - dy) = 1 + 2(4 - 5) = -1 < 0$$

$$x_i = x_{i-1} = 7$$

$$d_{i+1} = d_i + 2(dx) = -1 + 2 * 4 = 7 > 0$$

$$x_i = x_{i-1} + 1 = 7 + 1 = 8$$

$$d_{i+1} = d_i + 2(dx - dy) = 7 + 2(4 - 5) = 5 > 0$$

$$x_i = x_{i-1} + 1 = 8 + 1 = 9$$

x	y
5	2
6	3
7	4
7	5
8	6
9	7

Example 3: Use Bresenham Line drawing algorithm to draw a line with P1(-5,2) & P2(-9,7)?

$$m = \frac{\Delta y}{\Delta x} = \frac{7-2}{-9+5} = \frac{5}{-4} = |-1.25| > 1$$

increment y by 1 and ranged between $y_1 \leq y \leq y_2$

compute x and decrement by 1

NOTE: $dx = |dx|$

$dy = |dy|$

$$d_i = 2dx - dy = 2 * 4 - 5 = 3 > 0$$

$$x_i = x_{i-1} - 1 = -5 - 1 = -6$$

$$d_{i+1} = d_i + 2(dx - dy) = 3 + 2(4 - 5) = 1 > 0$$

x	y
-5	2
-6	3
-7	4
-7	5
-8	6
-9	7

$$x_i = x_{i-1} - 1 = -6 - 1 = -7$$

$$d_{i+1} = d_i + 2(dx-dy) = 1 + 2(4-5) = -1 < 0$$

$$x_i = x_{i-1} = -7$$

$$d_{i+1} = d_i + 2(dx) = -1 + 2 * 4 = 7 > 0$$

$$x_i = x_{i-1} - 1 = -7 - 1 = -8$$

$$d_{i+1} = d_i + 2(dx-dy) = 7 + 2(4-5) = 5 > 0$$

$$x_i = x_{i-1} - 1 = -8 - 1 = -9$$

-Advantages of Bresenham line drawing algorithm

1. It is a simple form of algorithm which is implemented in the firmware of the graphics hardware.
2. It uses only integer calculations.
3. The performance of this algorithm is faster.

-Disadvantages of Bresenham line drawing algorithm

1. Although this algorithm is simple it is used only to draw basic line, it is not meant for smooth lines.

2.The essential difference between Bresenham and DDA algorithm is that DDA uses floating point values while Bresenham employs integer with round off functions.

3.Furthermore, the computation of DDA algorithm involves multiplication and division but in Bresenham algorithm, addition and subtraction are the main operations performed over the integers.

-Comparison Chart between DDA and Bresenham line drawing algorithm

Basis for comparison	DDA Algorithm	Bresenham Algorithm
Efficiency	Low	High
Calculations involved	Complex	Simple
Speed	Comparatively less	More
Operations used	Multiplication and division	Additions and subtraction
Arithmetic computation values	Floating point	Integer type
Precision	Low	High
Cost	Expensive	Moderate or cheaper relatively.

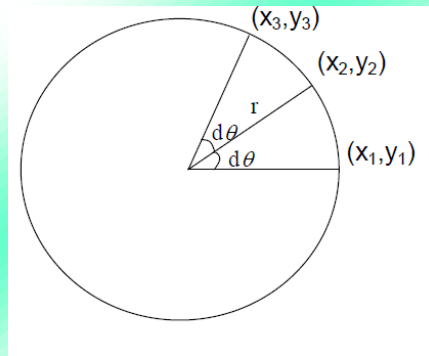
LECTURE 10 : THE CIRCLE

The circle is a special kind of curves. The circle is a closed curve with same starting and ending point. Circles are probably the most used curves in elementary graphics. A circle is specified by the coordinates of its center (**xc,yc**) and its radius **r** as shown below. The most familiar equation of the circle is:

$$(x - xc)^2 + (y - yc)^2 = r^2$$

If the center of the circle is at the origin (0,0) the above equation reduces to:

$$X^2 + y^2 = r^2$$



-CIRCLE DRAWING ALGORITHMS

- 1) General method using the circle equation.
- 2) Polar coordinates.
- 3) The Bresenham's method.

1-GENERAL METHOD

Drawing center (xc, yc) and drawing radius r.

To draw the circle points, y values should be extracted from the previous general equation:

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

This method has disadvantages

To calculate the values of y in the drawing of each point requires complex calculations in the equation such as square root and multiplication.

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

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

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

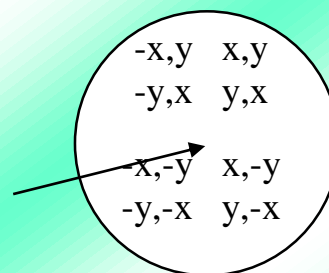
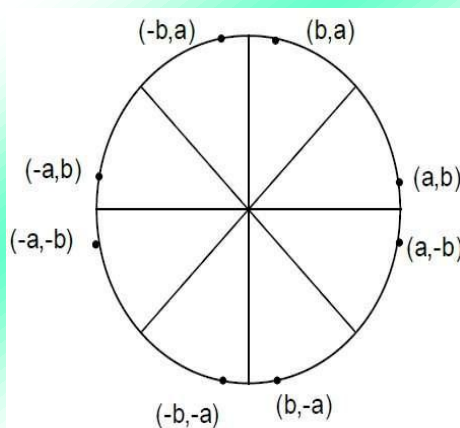
The perimeter points appear spaced so the second method is used to draw the circle.

2-POLAR COORDINATES

This method was ignored because of the time it took to calculate the polar functions to draw each point of the circle, the third method was adopted.

3-BRESENHAM'S CIRCLE DRAWING ALGORITHM

Draw the circle using the coordinates of the center (0,0) or one of the quarters of the circle in the drawing. We assume that the circle is drawn from the origin point to be a general case, so we draw the part points of the circle, and compensate the rest of the parts based on the principle of symmetry, add x_c , y_c to the rest of the values as follows:



$$(x, y) \Rightarrow (x + x_c, y + y_c)$$

First quadratic

$$(y, x) \Rightarrow (y + x_c, x + y_c)$$

$$(-x, y) \Rightarrow (-x + x_c, y + y_c)$$

Second quadratic

$$(-y, x) \Rightarrow (-y + x_c, x + y_c)$$

$$(-x, -y) \Rightarrow (-x + x_c, -y + y_c)$$

Third quadratic

$$(-y, -x) \Rightarrow (-y + x_c, -x + y_c)$$

$$(x, -y) \Rightarrow (x + x_c, -y + y_c)$$

Forth quadratic

$$(y, -x) \Rightarrow (y + x_c, -x + y_c)$$

The algorithm is described step by step as follows:

$$(x_1, y_1) = (0, r)$$

$$d_i = 3 - 2r$$

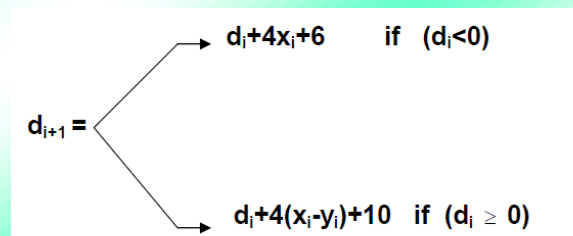
$$\text{if } d_i < 0 \text{ then } \left. \begin{array}{l} x_{i+1} = x_i + 1 \\ y_{i+1} = y_i \end{array} \right\} (x_i + 1, y_i) \Rightarrow (x++, y)$$

$$d_{i+1} = d_i + 4 * x_{i+1} + 6$$

$$\text{else } \left. \begin{array}{l} x_{i+1} = x_i + 1 \\ y_{i+1} = y_i - 1 \end{array} \right\} (x_i + 1, y_i - 1) \Rightarrow (x++, y--)$$

$$d_{i+1} = d_i + 4 (x_{i+1} - y_{i+1}) + 10$$

Repeat the last step in the calculation of a new coefficient and compare it with zero until we reach the breakpoint is that $x_i \geq y_i$.



Example1: Draw by Bresenham's circle drawing algorithm circle with equation $(x - 10)^2 + (y + 7)^2 = 16$

Solution

$x_c = 10$, $y_c = -7$, $r = 4$ $(x_1, y_1) = (0, r) = (0, 4)$, the circle is on the 4'th quarter

$$d_i = 3 - 2r = 3 - 2*4 = -5 < 0$$

$$(x++, y) \Rightarrow (1, 4)$$

$$d_{i+1} = d_i + 4*x + 6 = -5 + 4*1 + 6 = 5 > 0$$

$$(x++, y--) \Rightarrow (2, 3)$$

$$d_{i+1} = d_i + 4(x-y) + 10 = 5 + 4(2-3) + 10 = 11 > 0$$

$$(x++, y--) \Rightarrow (3, 2)$$

stop because $x \geq y$ (3,2)

x	y
0	4
1	4
2	3
3	2

Stop

Here we find the typical part to start the solution and we rely on it and on the principle of symmetry to find the total values as follows:

$x + x_c$	$y + y_c$	$y + x_c$	$x + y_c$	1'st quad.
$0+10= 10$	$4-7= -3$	$4+10= 14$	$0-7= -7$	
$1+10= 11$	$4-7= -3$	$4+10= 14$	$1-7= -6$	
$2+10= 12$	$3-7= -4$	$3+10= 13$	$2-7= -5$	
$3+10= 13$	$2-7= -5$	$2+10= 12$	$3-7= -4$	
$-x + x_c$	$y + y_c$	$-y + x_c$	$x + y_c$	2'nd quad.
$-0+10= 10$	$4-7= -3$	$-4+10= 6$	$0-7= -7$	
$-1+10= 9$	$4-7= -3$	$-4+10= 6$	$1-7= -6$	
$-2+10= 8$	$3-7= -4$	$-3+10= 7$	$2-7= -5$	
$-3+10= 7$	$2-7= -5$	$-2+10= 8$	$3-7= -4$	
$-x + x_c$	$-y + y_c$	$-y + x_c$	$-x + y_c$	3'rd quad.
$-0+10= 10$	$-4-7= -11$	$-4+10= 6$	$-0-7= -7$	
$-1+10= 9$	$-4-7= -11$	$-4+10= 6$	$-1-7= -8$	
$-2+10= 8$	$-3-7= -10$	$-3+10= 7$	$-2-7= -9$	
$-3+10= 7$	$-2-7= -9$	$-2+10= 8$	$-3-7= -10$	
$x + x_c$	$-y + y_c$	$y + x_c$	$-x + y_c$	4'th quad.
$0+10= 10$	$-4-7= -11$	$4+10= 14$	$-0-7= -7$	
$1+10= 11$	$-4-7= -11$	$4+10= 14$	$-1-7= -8$	
$2+10= 12$	$-3-7= -10$	$3+10= 13$	$-2-7= -9$	
$3+10= 13$	$-2-7= -9$	$2+10= 12$	$-3-7= -10$	

HW: Draw the circles with equations as follows

$$(x - 10)^2 + (y - 7)^2 = 4$$

$$x^2 + y^2 = 9$$

The sub-program to draw a circle using the Bresenham's Algorithm

```
void CIR ( int x , int y , int r )
{int x1 , y1 , d ;
x1 = 0 ;
y1 = r ;
d = abs(3 - 2 * r) ;
while ( x1 <= y1 ) {
x1 ++ ;
putpixel ( x1 + x , y1 + y , RED) ;
```

```

putpixel ( -x1 + x , y1 + y , RED) ;
putpixel ( x1 + x , -y1 + y , RED) ;
putpixel ( -x1 + x , -y1 + y , RED) ;
putpixel ( y1 + x , x1 + y , RED) ;
putpixel ( -y1 + x , x1 + y , RED) ;
putpixel ( y1 + x , -x1 + y , RED) ;
putpixel ( -y1 + x , -x1 + y , RED) ;
if ( d < 0 ) d = d + 4 * x1 + 6 ;
    else { d = d + 4 * (x1 - y1) + 10 ;
          y1 -- ;}
    }

```

Also the following sub-program to draw the circle with Bresenham circle drawing algorithm

```

Input: r.
Output: circle.
{ x=0; y=r;d=3-2r;
  while ( x ≤ y)
  { putpixel(x,y,color);
    if (d<0)
      d=d+4x+6;
    else
      { d=d+4(x-y)+10; y--;}
    x++
  }
}

```


LECTURE 11: PAINT AREA (FILLING AREA)

lectures Description

- Main techniques covered in this lecture include:
 - paint area (filling area) (scan line algorithm, flood fill algorithm, boundary fill algorithm)
 - problem with boundary fill
 - character generation
 - Polygon paint, triangle paint and circle paint

Prerequisites

- Mathematical fundamental.
- Geometry fundamental.
- C++ programming.

lecture Objectives

- This lecture will learn the students about current techniques in computer graphics. By the end of the lecture, the students should:
 - Paint and filling area in general with color and pattern with various method.
 - Character paint algorithm.
 - How to paint triangle, circle and polygon.

Reference Material

-Online textbook /Introduction to Computer Graphics Using Java 2D and 3D
Frank Klawon

DOI <https://doi.org/10.1007/978-1-4471-2733-8> . Copyright Information Springer-Verlag London Limited 2012

- Computer Graphics I, Polygon Clipping and Filling, Week 3, Lecture 5, David Breen, William Regli and Maxim Peysakhov, Geometric and Intelligent Computing Laboratory, Department of Computer Science, Drexel University

<http://gicl.cs.drexel.edu>

LECTURE 11: PAINT AREA (FILLING AREA)

Polygon is an Geometrical shape with vertices as shown below in the following figure. For filling polygons with particular colors or shading with different pattern, one need to identify the pixels that fall into the polygon and those fall on the border.



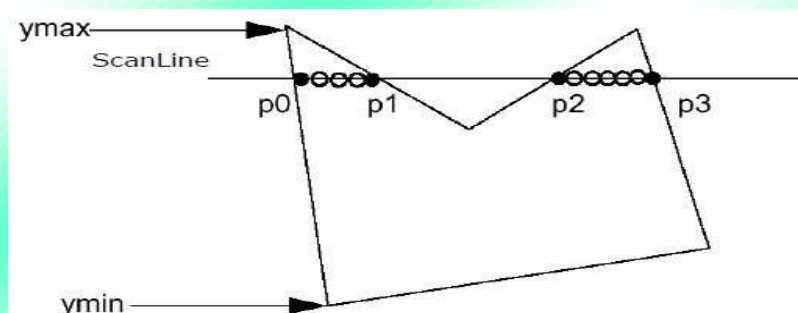
Figure 1 polygon boarder

There are various methods which used to fill the area. One way to fill an area is to scan line that cross the area. Another method is to fill the area is to start from a given interior position and paint out wards from this point until the boundary is encountered.

1-SCAN LINE ALGORITHM

This algorithm works by intersecting scan line with polygon edges and fills the polygon between pairs of intersections. The following steps depict how this algorithm works.

Step 1 – Find out the Ymin and Ymax from the given polygon.



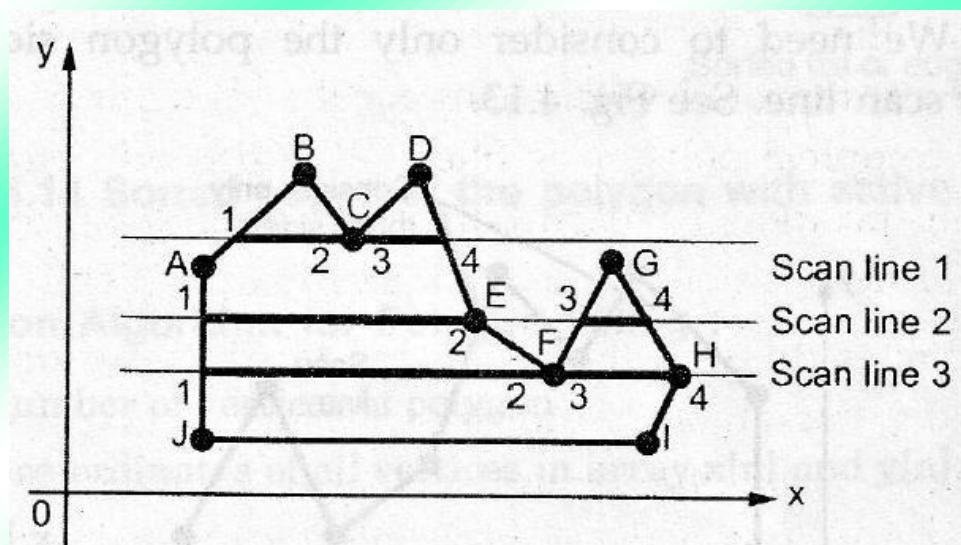
Step 2 – Scan Line intersects with each edge of the polygon from Ymin to Ymax. Name each intersection point of the polygon. As the figure shown above, they are named as p0, p1, p2, p3.

Step 3 – Sort the intersection point in the increasing order of X coordinate i.e. (p0, p1), (p1, p2), and (p2, p3).

Step 4 – Fill all those pair of coordinates that are inside polygons and ignore the alternate pairs.

For each scan-line crossing a polygon, the algorithm locates the intersection points are of scan line with the polygon edges.

- 1.This intersection points are stored from left to right.
- 2.For vertex we must look at the other endpoints of the two line segments of the polygon which meet at this vertex.
3. If these points lie on the same (up or down) side of the scan line, the that point is counts as two intersection points.
- 4.If they lie on opposite sides of the scan line, then the point is counted as single intersection. As shown below



2-FLOOD FILL ALGORITHM

Sometimes we come across an object where we want to fill the area and its boundary with different colors. We can paint such objects with a specified interior color instead of searching for particular boundary color as in boundary filling algorithm. Instead of relying on the boundary of the object, it relies on the fill color. In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original

interior color exist, the algorithm is completed. Once again, this algorithm relies on the Four-connect or Eight-connect method of filling in the pixels. But instead of looking for the boundary color, it is looking or all adjacent pixels that are a part of the interior.

3-BOUNDARY FILL ALGORITHM

1. In this method, edges of the polygons are drawn. Then starting with some pixel, any point inside the polygon the neighboring pixels are examined to check whether the boundary pixel is reached.

2.If boundary pixels are not reached, pixels are highlighted and the process is continued until boundary pixels are reached.

3.Boundary defined regions may be either 4-cornected or 8-connected as shown in the Figure below

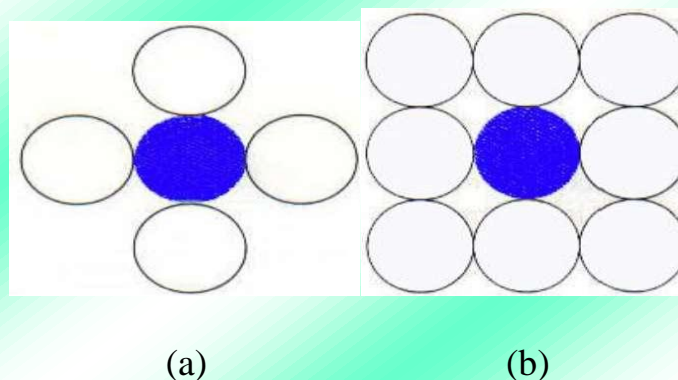


Figure 2 neighbor pixel connected to one pixel.

(a) Four connected region (b) Eight connected

- **4-connected:**

In this firstly there is a selection of the interior pixel which is inside the boundary then in reference to that pixel, the adjacent pixel will be filled up that is top-bottom and left-right.

- **8-connected:**

This is the best way of filling the color correctly in the interior of the area defined. This is used to fill in more complex figures. In this four

diagonal pixel are also included with a reference interior pixel (including top-bottom and left right pixels).

-Problem with boundary fill

- 1.It may not fill regions correctly, if same interior pixels are also displayed in full color.
- 2.In 4-connected there is a problem. Sometimes it does not fill the corner pixel as it checks only the adjacent position of the given pixel.

-CHARACTER GENERATION

1. We can display letters and numbers in variety of size and style.
2. The overall design style for the set of character is called typeface.
3. Today large numbers of typefaces are available for computer application for example Helvetica, New York etc.
4. Originally, the term font referred to a set of cast metal character forms in a particular size and format, such as 10-point Courier Italic or 12- point Palatino Bold. Now, the terms font and typeface are often used interchangeably, since printing is no longer done with cast metal forms.
5. Two different representations are used for storing computer fonts.

-Bitmap Font/ Bitmapped Font

1. A simple method for representing the character shapes in a particular typeface is to use rectangular grid as shown below

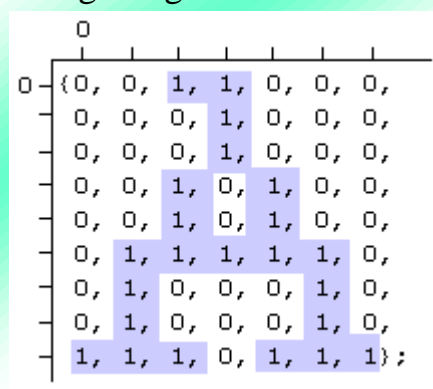


Figure 3 Grid pattern for letter A

2. When the pattern in is copied to the area of frame buffer, the 1 bits designate which pixel positions are to be displayed on the monitor.
3. Bitmap fonts are the simplest to define and display as character grid only need to be mapped to a frame- buffer position.

4. Bitmap fonts require more space because each variation (size and format) must be stored in a font cache.

-Outline Font

In this method character is generated using curve section and straight line as combine assembly.

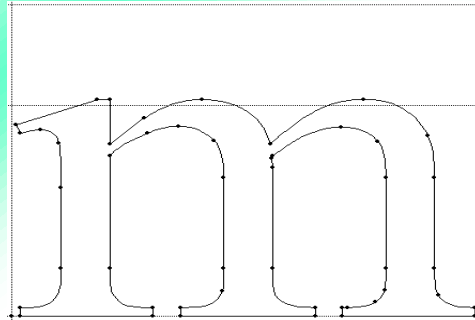


Figure 4 outline for letter m

- To display the character shown in figure one need to fill interior region of the character.
- This method requires less storage since each variation does not required a distinct font cache.
- We can produce boldface, italic, or different sizes by manipulating the curve definitions for the character outlines.
- But this will take more time to process the outline fonts, because they must be scan converted into the frame buffer.

-Fill Styles

1. Area are generally displayed with three basic style hollow with color border, filled with solid color, or filled with some design.
2. Value of **FS** s include hollow, solid, pattern etc.
3. Another values for fill style is hatch, which is patterns of line like parallel line or crossed line. As shown below:

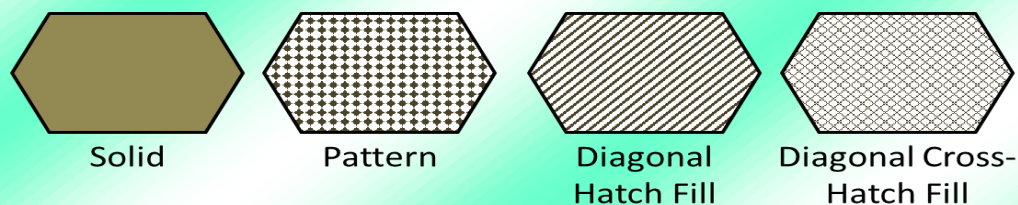


Figure 5 Different style of area filling.

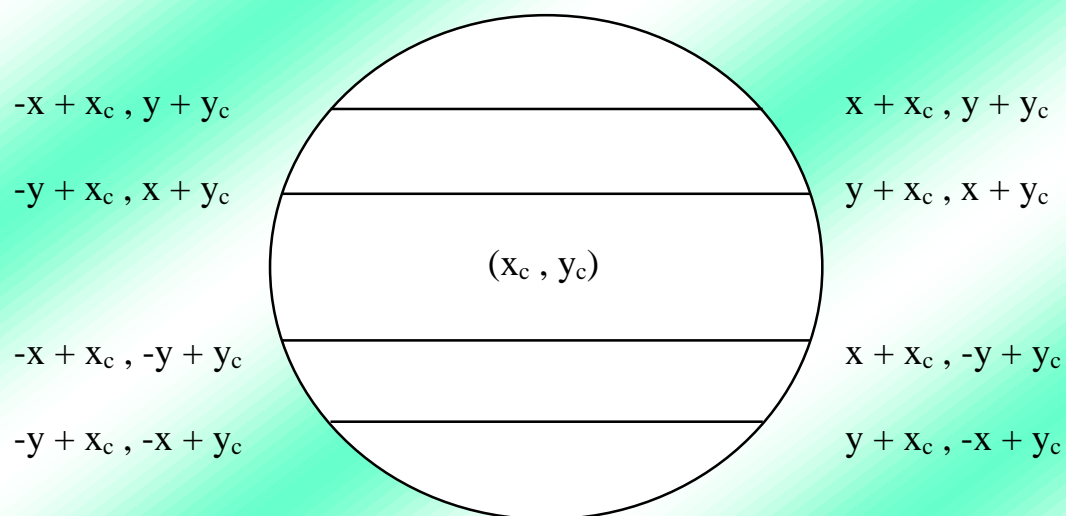
-RECTANGLE PAINT

It is one of the simplest types of coloration of spaces. It has a length and width. The scanner starts horizontally, one line after another. The value of y changes so that it increases by 1, and the scanner starts from left to right, turning the coloring from geometric to pixel.

```
void FillRectangle (int x , int y , int width , int hieght , int color)
{
  int i , j ;
  for ( i = y ; i <= y + height - 1 ; i ++ )
    for ( j = x ; j <= x + width - 1 ; j ++ )
      putpixel ( j , i , color ) ;
}
```

- CIRCLE PAINT

The same style is used to draw the circle but by taking the lines between the horizontal horizontal lines as follows:



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

The subprogram to fill a circle using the Bresenham algorithm

```
void CIR ( int x , int y , int r )
{
    int x1 , y1 , d ;
    x1 = 0 ;
    y1 = r ;
    d = abs(3 - 2 * r) ;
    while ( x1 <= y1 ) {
        x1 ++ ;
        setcolor( RED ) ;
        line ( x1+x , y1+y , -x1+x , y1+y ) ;
        line ( x1+x , -y1+y , -x1+x , -y1+y ) ;
        line ( y1+x , x1+y , -y1+x , x1+y ) ;
        line ( y1+x , -x1+y , -y1+x , -x1+y ) ;
        if ( d < 0 ) d = d + 4 * x1 + 6 ;
        else { d = d + 4 * (x1 - y1) + 10 ;
            y1 -- ;}}}
```

- TRIANGLE PAINT

We will take the triangle when its base is parallel to the scan line (horizontal) as follows:

$$\therefore m = \frac{y - y_1}{x - x_1}$$

$$x - x_1 = \frac{1}{m}(y - y_1)$$

$$\therefore x = \frac{1}{m}(y - y_1) + x_1$$

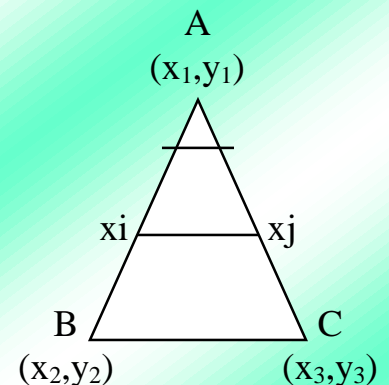
The procedure is to draw with 3 Points

$$m_1 = \frac{y_2 - y_1}{x_2 - x_1} \text{ calculate the slope between pt. A,B}$$

$$m_2 = \frac{y_3 - y_1}{x_3 - x_1}$$

$$y_1 \leq y \leq y_2$$

calculate xi based on m1 calculate xj based on m2



$$x_i = \frac{1}{m_1}(y - y_1) + x_1$$

$$x_j = \frac{1}{m_2}(y - y_1) + x_1$$

(x_j, y) and (x_i, y) then draw the line between these points

Line (x_i, y, x_j, y)

continue the algorithm until $y = y_2$

Sub program to fill a triangle

```
void FillTri ( int x1 , int y1 int x2 , int y2 , int x3 , int y3 )
```

```
{float m1 , m2 , xi , xj ;
```

```
    m1 = ( float ) ( y2 - y1 ) / ( x2 - x1 ) ;
```

```
    m2 = ( float ) ( y3 - y1 ) / ( x3 - x1 ) ;
```

```
    setcolor ( RED ) ;
```

```
    for ( int y = y1 ; y < y2 ; y ++ ) {
```

```
        xi = 1/m1 * ( y - y1 ) + x1 ;
```

```
        xj = 1/m2 * ( y - y1 ) + x1 ;
```

```
        line ( xi , y , xj , y ) };
```

LECTURE 12: 2D TRANSFORMATIONS

1 - TRANSLATION

Is the process of converting, moving or dragging the shape or line and move it from one location to another by adding the transfer value or drag T_x , T_y to the original coordinates, where T_x , T_y are the vector of conversion.

$$x' = x + T_x$$

$$y' = y + T_y \quad \text{where } T_x, T_y \text{ are translation vector}$$

To convert any shape, for example, triangle, rectangle or any polygon, add the transformation values of the vector to each point of the shape, while the circle is added to the center of the circle. As shown in the figure below.

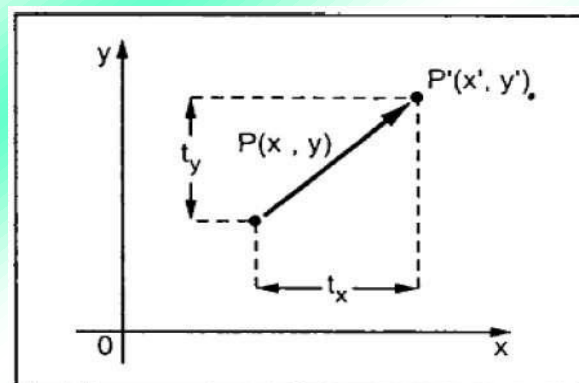


Figure 1 translation transformation

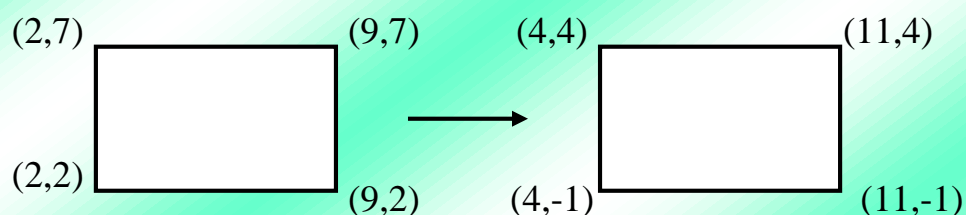
-Example : Translate the point defined $P(5,6)$ by $(4,4)$

$$[x' \ y'] = [x \ y] + [T_x \ T_y]$$

$$= [5 \ 6] + [4 \ 4]$$

$$= [9 \ 10] \quad \text{the final coordinate is } P'(9,10)$$

-Example : Translate the rectangle below by $T_x=2$, $T_y= -3$



--Translation with matrix

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{T}$$

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

-Example : translate P1(2,7) by $T_x=2$, $T_y= -3$

$$[x' \ y' \ 1] = [2 \ 7 \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -3 & 1 \end{bmatrix} = \begin{bmatrix} 2*1+7*0+1*2 \\ 2*0+7*1+1*-3 \\ 2*0+7*0+1*1 \end{bmatrix} \\ = [4 \ 4 \ 1]$$

The final coordinate is P'(4,4)

2- SCALING

The process of resizing the shape by multiplying the coordinates of the head with the scaling vector (S_x , S_y)

$$\mathbf{x}' = \mathbf{x} * \mathbf{S}_x$$

$$\mathbf{y}' = \mathbf{y} * \mathbf{S}_y \quad \text{where } S_x, S_y \text{ are scaling vector}$$

if $S_x, S_y < 1 \Rightarrow$ Minimization process

if $S_x, S_y > 1 \Rightarrow$ Maximization process

if $S_x, S_y = 1 \Rightarrow$ The shape still without change

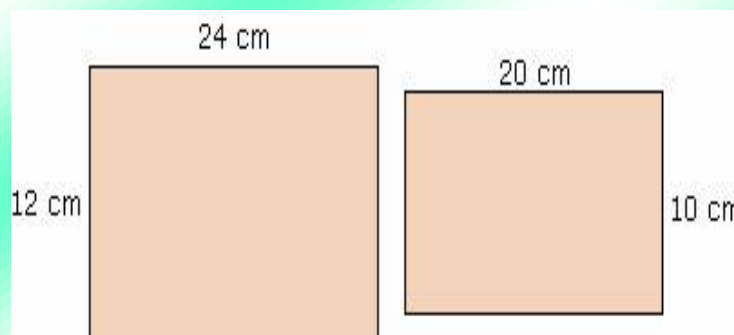


Figure 2 scaling transformation

In the scaling process, the position of the shape changes so that the point (x_f, y_f) , which represents the fixed point of the scaling, is taken.

$$\mathbf{x}' = \mathbf{x}_f + (\mathbf{x} - \mathbf{x}_f) \mathbf{S}_x$$

$$y' = yf + (y - yf) S_y$$

Example: change the size of the following figure using 2 units on the x-axis and 0.5 units on the y-axis and around the point [2 3]?

$$x_1' = 2 + (4 - 2) * 2 = 6$$

$$y_1' = 3 + (8 - 3) * 0.5 = 5.5$$

$$(4, 8) \Rightarrow (6, 5.5)$$

$$x_2' = 2 + (2 - 2) * 2 = 2$$

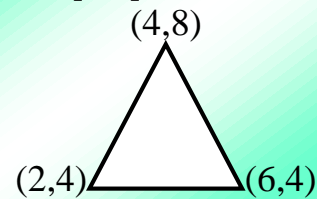
$$y_2' = 3 + (4 - 3) * 0.5 = 3.5$$

$$(2, 4) \Rightarrow (2, 3.5)$$

$$x_3' = 2 + (6 - 2) * 2 = 10$$

$$y_3' = 3 + (4 - 3) * 0.5 = 3.5$$

$$(6, 4) \Rightarrow (10, 3.5)$$



-Scaling by a matrix

$$P' = P.S$$

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example : scale the line starts with p1(4,10) and p2(10,10). With $S_x=2$ and $S_y=4$, either by direct equation and by matrix.

$$x' = x * S_x$$

$$y' = y * S_y$$

$$p1'(x,y) = [4*2 \ 10*4] = [8 \ 40]$$

$$p2'(x,y) = [10*2 \ 10*4] = [20 \ 40]$$

So the scaling line final coordinate are p1(8,40) & p2(20,40)

$$P'=P.S$$

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p1'[x' \ y' \ 1] = [4 \ 10 \ 1] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [8 \ 40 \ 1]$$

$$p1'[x' \ y' \ 1] = [10 \ 10 \ 1] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [20 \ 40 \ 1]$$

3- ROTATION

To rotate a point (x,y) with a clockwise angle θ about the origin of the coordinate system, we write:

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

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

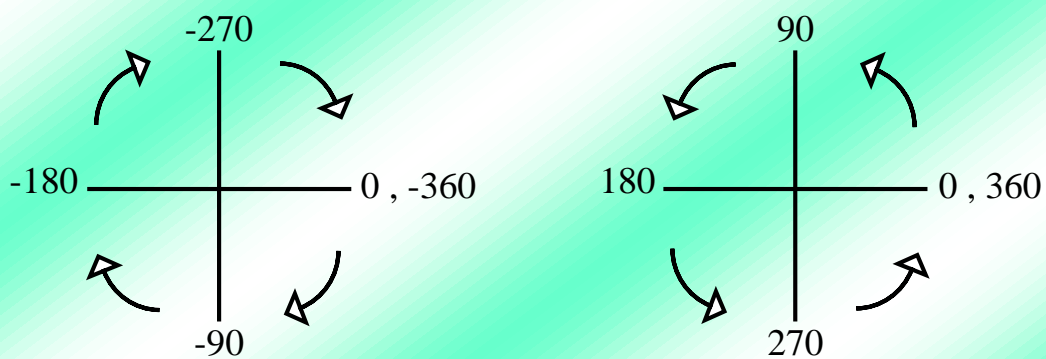


Figure 3 rotation transformation

Rotation transformation can be represented in a uniform way by a 2×2 matrix as shown below:

$$[x' \ y'] = [x \ y] \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

If θ is negative (clockwise rotation matrix) could be

$$[x' \ y'] = [x \ y] \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

Example : Rotate the point (2,8) by angle (90°), *where* $\cos 90 = 0$, $\sin 90 = 1$

$$[x' \ y' \ 1] = [12 \ 8 \ 1] \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-8 \ 12 \ 1]$$

The final coordinate after rotation is (-8,12)

Example : Consider triangle defined by its three vertices (20,0), (60,0), (40,100) being rotated 250 clockwise about the origin. What are the new vertices?

- HOMOGENOUS MATRIX

Is the result of the process of multiplying the three matrices of the transfers with each other and make it a single matrix. Rotation and

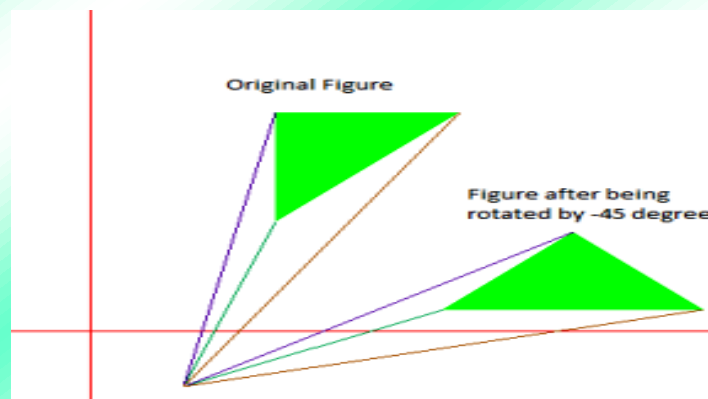


Figure 4 rotation transformation

conversion at a fixed point:

1. Convert the shape at the point of origin (0,0) by $T_x = -xf$, $T_y = -yf$.
2. Rotate around the origin point at an angle of θ .

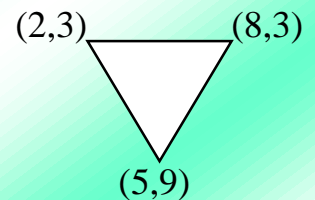
3. Reverse conversion at point of origin (0,0) by $T_x = +xf$, $T_y = +yf$.

The matrix will be as follows:

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ (1-\cos\theta)xf + yf \sin\theta & (1-\cos\theta)yf - xf \sin\theta & 1 \end{bmatrix}$$

Example : Rotate the triangle with angle 90 degree with fixed point (4,4) with homogenous matrix.

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ (1-\cos\theta)xf + yf \sin\theta & (1-\cos\theta)yf - xf \sin\theta & 1 \end{bmatrix} \\ = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ (1-0)4 + 4*1 & (1-0)4 - 4*1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 8 & 0 & 1 \end{bmatrix}$$



$$\begin{aligned} [x' \ y' \ 1] &= [2 \ 3 \ 1] * \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 8 & 0 & 1 \end{bmatrix} = [5 \ 2 \ 1] \\ p'_{1,2,3} [x' \ y' \ 1] &= [8 \ 3 \ 1] * \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 8 & 0 & 1 \end{bmatrix} = [5 \ 8 \ 1] \\ [x' \ y' \ 1] &= [5 \ 9 \ 1] * \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 8 & 0 & 1 \end{bmatrix} = [-1 \ 5 \ 1] \end{aligned}$$

Example : rotate the line with points P1(3,3) and p2(7,2) with Symmetry matrix.

$$\begin{aligned} [x' \ y' \ 1] &= [3 \ 3 \ 1] * \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 9 & -5 & 1 \end{bmatrix} = [6 \ -2 \ 1] \\ [x' \ y' \ 1] &= [10 \ 7 \ 1] * \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 9 & -5 & 1 \end{bmatrix} = [2 \ 5 \ 1] \end{aligned}$$

LECTURE 13: 2D TRANSFORMATION

4- REFLECTION

It is the process of creating a mirror for a particular shape image. In other words, we can say that it is a **rotation** operation with 180° . In **reflection transformation**, the size of the object does not change.

- reflect about X-axis

using the following matrix the value remains x-axis and reflect the y-axis values.

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

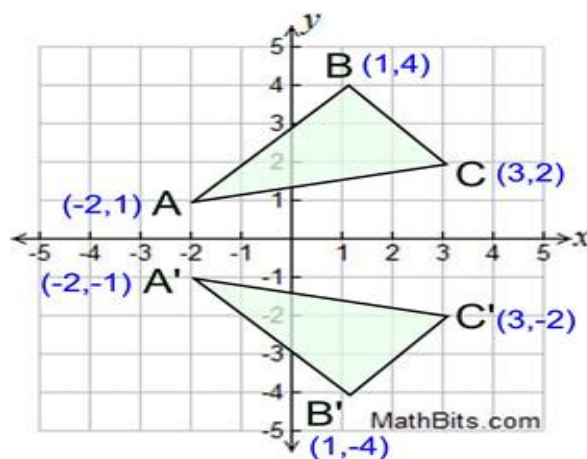


Figure 1 reflection about x-axis

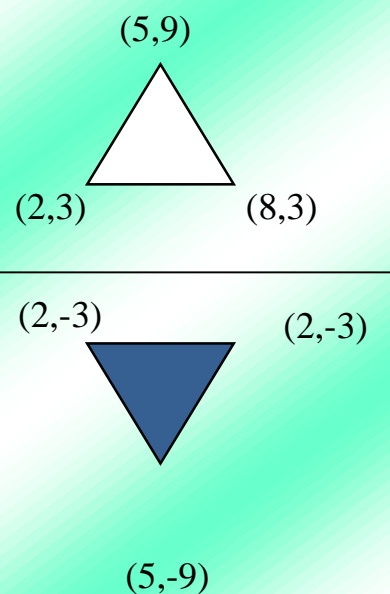
Example : Reflects the triangle about x-axis

$$[x' \ y' \ 1] = [2 \ 3 \ 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [2 \ -3 \ 1]$$

$$[x' \ y' \ 1] = [8 \ 3 \ 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [8 \ -3 \ 1]$$

$$[x' \ y' \ 1] = [5 \ 9 \ 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [5 \ -9 \ 1]$$

The final coordinate are (2,-3), (8,-3), (5,-9)



- Reflection about Y-axis

using the following matrix: the value remains Y-axis and reflect the X-axis signal.

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

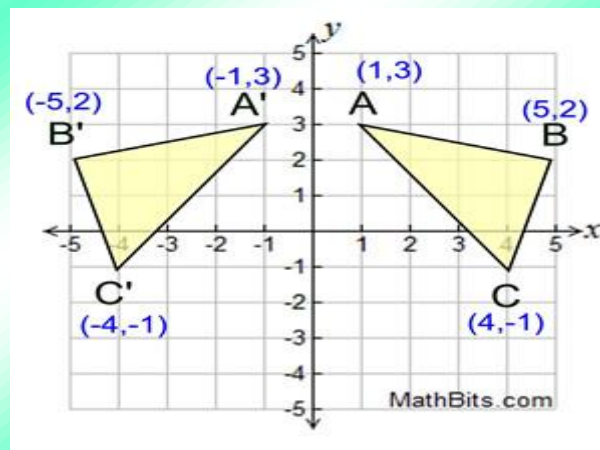
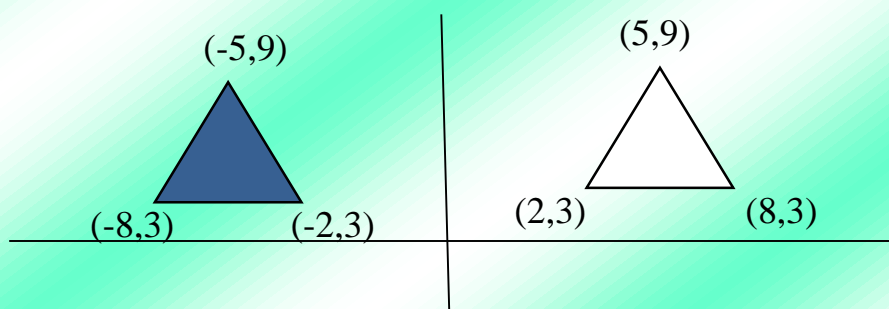


Figure 2 reflection about y-axis

Example: Reflects the triangle about y-axis

$$\begin{aligned} [x' \ y' \ 1] &= [2 \ 3 \ 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-2 \ 3 \ 1] \\ [x' \ y' \ 1] &= [8 \ 3 \ 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-8 \ 3 \ 1] \\ [x' \ y' \ 1] &= [5 \ 9 \ 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-5 \ 9 \ 1] \end{aligned}$$



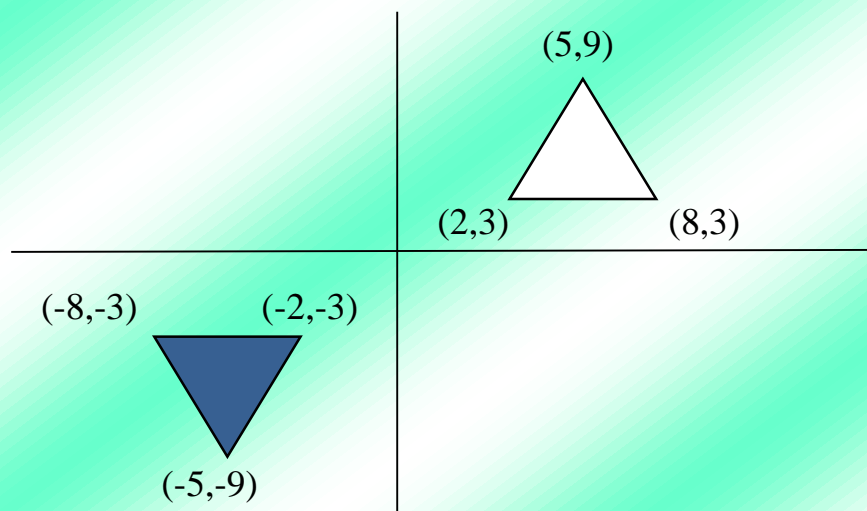
The final coordinate are (-2,3), (-8,3), (-5,9)

- Reflection about the origin , reflects both x and y-axis.

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

Example : Reflects about the origin , reflects both x and y-axis.

$$\begin{aligned} [x' \ y' \ 1] &= [2 \ 3 \ 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-2 \ -3 \ 1] \\ [x' \ y' \ 1] &= [8 \ 3 \ 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-8 \ -3 \ 1] : \\ [x' \ y' \ 1] &= [5 \ 9 \ 1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [-5 \ -9 \ 1] \end{aligned}$$



The final coordinate are (-2,3), (-8,-3), (-5,-9)

- **Reflects over $y=x$**

When a point is reflected across the line $y = x$, the x -coordinate and y -coordinate change places. **The reflection of the point (x,y) across the line $y = x$ is the point (y, x) .**

If a point is reflected over the line $y = -x$, the x -coordinate and y -coordinate change places and are negated (the signs are changed). **The reflection of the point (x,y) across the line $y = -x$ is the point $(-y, -x)$.**

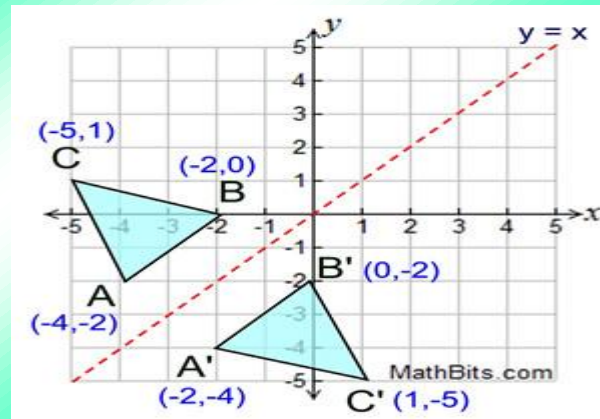


Figure 3 reflection over $y=x$

- **Reflects over any line**

Remember that each point of a reflected image is the same distance from the line of reflection as the corresponding point of the original figure. The line of reflection will lie directly in the middle between the original figure and its image. one may be able to simply "count" these distances on the grid

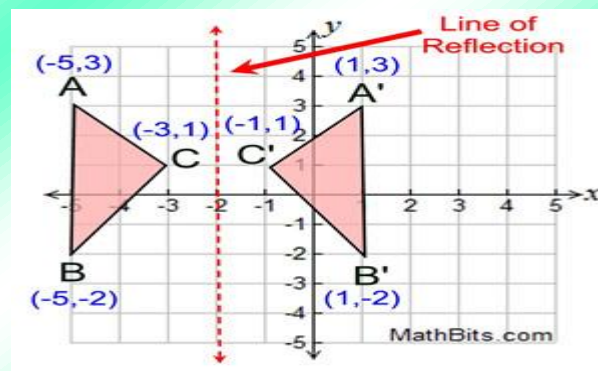


Figure 4 reflection over any line

5-SHEARING

-Shearing the shape along the X coordinate using matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ Shx & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

-Shearing the shape along the Y coordinate using matrix

$$\begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

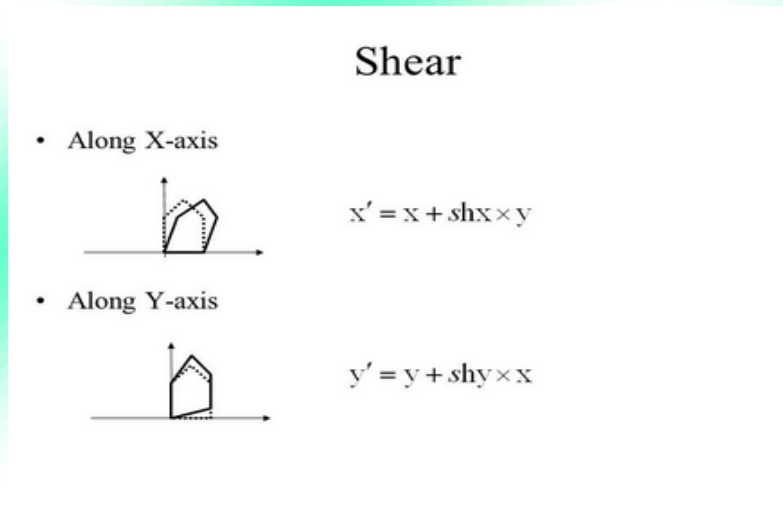


Figure 4 shearing transformation

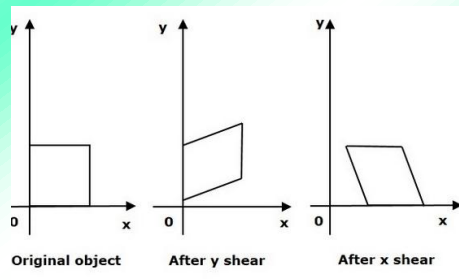
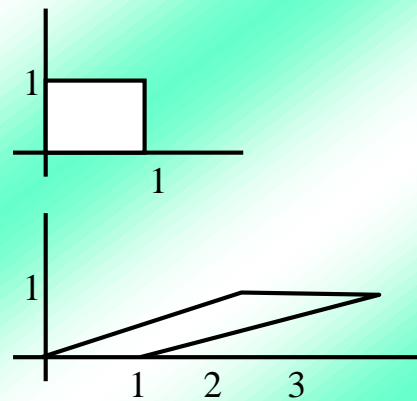


Figure 5 shearing transformation

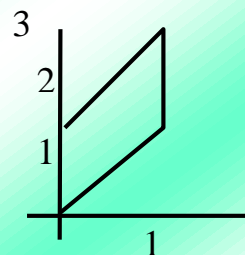
-Example : shear the following figure about x-axis and once on the y-axis. With 2 unit both.

- Shear on X-axis

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 1 \\ 1 & 0 & 1 \\ 3 & 1 & 1 \end{bmatrix}$$



- Shear on Y-axis



$$\begin{array}{l}
[0 \ 0 \ 1]^* \begin{bmatrix} 1 & 2 & 0 \end{bmatrix} = [0 \ 0 \ 1] \\
[0 \ 1 \ 1]^* \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} = [0 \ 1 \ 1] \\
[1 \ 0 \ 1]^* \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = [1 \ 2 \ 1] \\
[1 \ 1 \ 1]^* \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = [1 \ 3 \ 1]
\end{array}$$

Homework

- 1) Consider a triangle defined by its three vertices $(40, 0)$, $(80, 0)$, $(60, 100)$ be translated 120 units to the right and 20 units up.
- 2) Consider a triangle defined by its three vertices $(20, 0)$, $(60, 0)$, $(40, 100)$ be translated 20 units to the left.
- 3) Consider a triangle defined by its three vertices $(40, 0)$, $(100, 0)$, $(60, 100)$ be translated 40 units to the left and 40 units down.
- 4) The triangle $(10, 0)$, $(30, 0)$, $(50, 80)$ rotate 45° clockwise about the origin.
- 5) Rotate the triangle $(10, 0)$, $(30, 0)$, $(50, 80)$ 45° counterclockwise about the origin.
- 6) Rotate the triangle $(7, 8)$, $(4, 4)$, $(10, 5)$ 90° counterclockwise about the point $(7, 8)$.
- 7) Rotate the above triangle 90° clockwise about the point $(4, 4)$.
- 8) Scale the triangle $(4, 4)$, $(7, 8)$, $(10, 5)$ by $S_x = 2$ and $S_y = 2$, about the origin point.
- 9) Magnify the triangle $(0, 0)$, $(8, 10)$, $(12, 4)$, 4 times its size, about the origin point.
- 10) Reflect the shape $(20, 70)$, $(40, 50)$, $(60, 70)$, $(40, 90)$, about:
 - 1- X – axis
 - 2- Y- axis
 - 3- origin $(0,0)$
 - 4- $y = x$
 - 5- $y = -x$, by used matrix representation, and draw the result.
- 11) Magnify the above triangle $1/2$ its size.
- 12) Magnify the triangle $(0, -3)$, $(-6, -7)$, $(6, -7)$, 3 times its size, about the point $(0, -3)$.
- 13) Magnify the triangle $(0, 0)$, $(10, 12)$, $(14, 6)$, 3 times its size, using matrix representation.

14) Rotate the above triangle 90° clockwise about the point $(10, 12)$ using matrix representation.

15) Translated the shape above 20 units to the left, and 10 units up. Using matrix representation.

16 Reflect the shape $(2, 2), (4, 4), (6, 2),$

about: X – axis b- Y- axis

origin $(0,0)$ d- $y = x$ e- $y = -x$, then draw the result. Shear the shape above with: a- $shx = 4$

b- $shy = 4$.

LECTURE 14: SUCCESSIVE TRANSFORMATIONS

1-TRANSLATION

The successive translation is performed as follows

$$\mathbf{P}' = \mathbf{P} + \mathbf{T}_1$$

$$\mathbf{P}'' = \mathbf{P}' + \mathbf{T}_2 \quad \text{OR} \quad \mathbf{P}'' = \mathbf{P} + (\mathbf{T}_1 + \mathbf{T}_2)$$

$$\begin{aligned} &= \mathbf{P} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x_1} & T_{y_1} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x_2} & T_{y_2} & 1 \end{bmatrix} \\ &= \mathbf{P} \cdot \begin{bmatrix} 1 & & 0 & 0 \\ 0 & & 1 & 0 \\ T_{x_1} + T_{x_2} & & T_{y_1} + T_{y_2} & 1 \end{bmatrix} \end{aligned}$$

2-SCALING

The successive scaling is performed as follows

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{S}_1$$

$$\mathbf{P}'' = \mathbf{P}' \cdot \mathbf{S}_2 \quad \text{OR} \quad \mathbf{P}'' = \mathbf{P} \cdot (\mathbf{S}_1 \cdot \mathbf{S}_2)$$

$$\begin{aligned} &= \mathbf{P} \cdot \begin{bmatrix} S_{x_1} & 0 & 0 \\ 0 & S_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_{x_2} & 0 & 0 \\ 0 & S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \mathbf{P} \cdot \begin{bmatrix} S_{x_1} S_{x_2} & & 0 & 0 \\ 0 & & S_{y_1} S_{y_2} & 0 \\ 0 & & 0 & 1 \end{bmatrix} \end{aligned}$$

3-ROTATION

The successive rotation is performed as follows

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{R}(\theta_1)$$

$$\mathbf{P}'' = \mathbf{P}' \cdot \mathbf{R}(\theta_2) \quad \text{OR} \quad \mathbf{P}'' = \mathbf{P} \cdot (\mathbf{R}(\theta_1 + \theta_2))$$

$$= P. \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= P. \begin{bmatrix} \cos(\theta_1 + \theta_2) & \sin(\theta_1 + \theta_2) & 0 \\ -\sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example : translate the line start with $P_1(2,3)$ and ends with $P_2(5,7)$
By $T_1(1,3)$ and $T_2(4,5)$?

$$\begin{aligned} [x' \ y' \ 1] &= [2 \ 3 \ 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1+4 & 3+5 & 1 \end{bmatrix} = [7 \ 11 \ 1] \\ [x' \ y' \ 1] &= [5 \ 7 \ 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1+4 & 3+5 & 1 \end{bmatrix} = [10 \ 15 \ 1] \end{aligned}$$

The final coordinates are (7,11) and (10,15)

Transformation summery

☒ Translation by $[T_x \ T_y]$

$$x' = x + T_x$$

$$y' = y + T_y$$

☒ Scaling by $[S_x \ S_y]$

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

- $S > 1 \Rightarrow$ scale up
- $S < 1 \Rightarrow$ scale down
- Uniform scale $\Rightarrow S_x = S_y$
- Non-uniform scale $\Rightarrow S_x \neq S_y$
- Null scaling $\Rightarrow [1 \ 1]$
- Inverse scaling $\Rightarrow [1/S_x \ 1/S_y]$
- Scale about fixed point (x_f, y_f)

$$x' = x_f + (x - x_f) \cdot S_x$$

$$y' = y_f + (y - y_f) \cdot S_y$$

LECTURE 15: 3D TRANSFORMATION

1- 3D TRANSLATION

(T_x, T_y, T_z) , where the first value represents the change in the location of the body in the three dimensions of the site x , the second in the location y , and the third in location z as shown in the figure 1. 3D translation use 4X4 matrices (X, Y, Z, h) . In 3D translation point (X, Y, Z) is to be translated by amount t_x, t_y and t_z to location (X', Y', Z') .

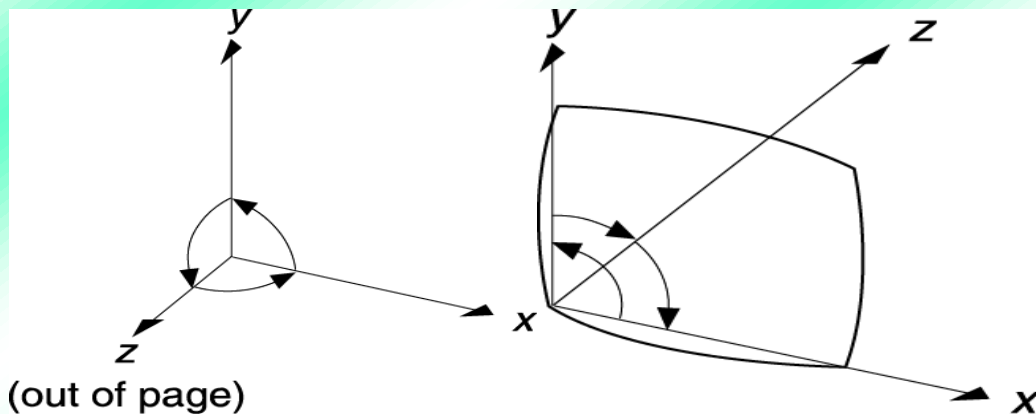


Figure 1 An illustration of the coordinates of the 3D as they appear on the screen

$$X' = x + t_x$$

$$Y' = y + t_y$$

$$Z' = z + t_z$$

The translation using matrix $v = (T_x, T_y, T_z)$

$$P' = T.P$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

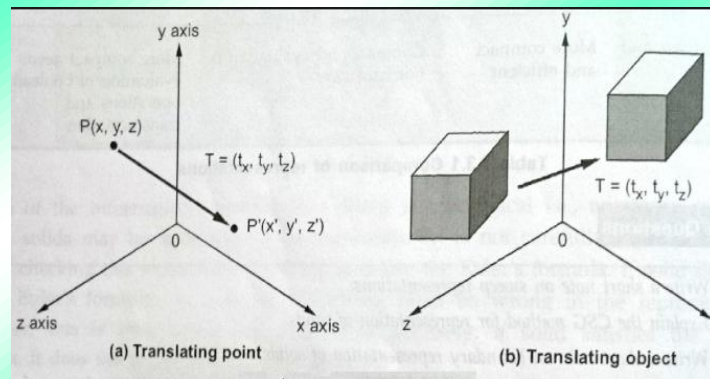


Figure 2 3D translation

Example : translate the point with coordinates (2,2,2) by the translation factor $T(2,4,6)$.

$$P' = T.P$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 8 \\ 1 \end{pmatrix}$$

Final translation coordinate is $P'(4,6,8)$

2- 3D ROTATION

Rotation is here by axis.

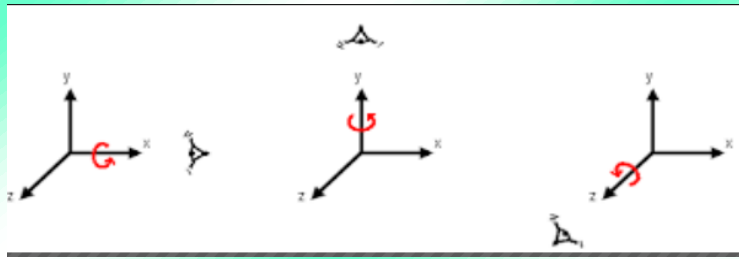


Figure 3 3D rotation

-For example, the rotation around the x-axis at a certain angle θ is represented by the following equations, and leave x unchanged

$$Y' = y \cos\theta - z \sin\theta$$

$$Z' = y \sin\theta + z \cos\theta$$

$$X' = x$$

matrix equation of rotation about the x axis

$$P' = R(\theta) \cdot P$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

the rotation around the y-axis at a certain angle θ is represented by the following equations, and leave y coordinate unchanged

$$X' = z \cos\theta - x \sin\theta$$

$$Z' = z \sin\theta + x \cos\theta$$

$$Y' = y$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

-the rotation around the z-axis at a certain angle θ is represented by the following equations, and leave z coordinate unchanged

$$X' = x \cos\Theta - y \sin\Theta$$

$$Y' = y \cos\Theta + x \sin\Theta$$

$$Z' = z$$

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

3- 3D SCALING

Scale change is to resize the object by scaling factors. It is performed by equation

$$P' = P.S$$

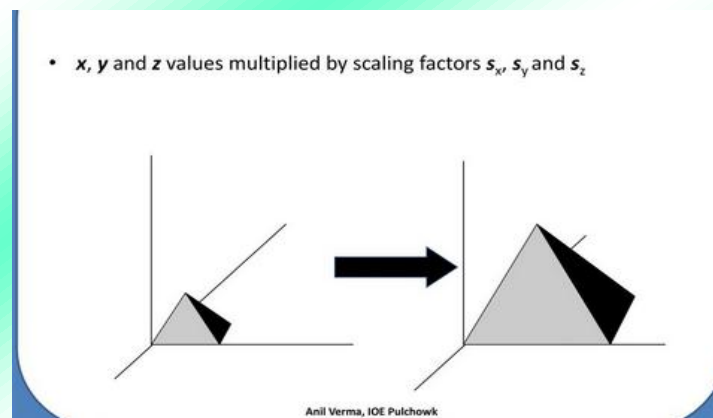


Figure 4 3D scaling

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- Example: - Scale the line AB with coordinates (10,20,10) and (20,30,30) respectively with scale factor S(3,2,4).

$$P' = S \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} A_x' & B_x' \\ A_y' & B_y' \\ A_z' & B_z' \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ 20 & 30 \\ 10 & 30 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A_x' & B_x' \\ A_y' & B_y' \\ A_z' & B_z' \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 30 & 60 \\ 40 & 60 \\ 40 & 120 \\ 1 & 1 \end{bmatrix}$$

Final coordinates after scaling are A' (30, 40, 40) and B' (60, 60, 120).

- **FIXED POINT SCALING**

$$P' = T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f) \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} s_x & 0 & 0 & (1 - s_x)x_f \\ 0 & s_y & 0 & (1 - s_y)y_f \\ 0 & 0 & s_z & (1 - s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P$$

4- 3D SHEARING

- Shearing matrix for Z-axis

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Shear matrix for X-axis is.

$$SH_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Shear matrix for Y-axis is.

$$SH_y = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5- 3D REFLECTION

- Reflection means mirror image produced when mirror is placed at require position.
- When mirror is placed in XY-plane we obtain coordinates of image by just changing the sign of z coordinate.
- Transformation matrix for reflection about XY-plane is given below.

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Transformation matrix for reflection about YZ-plane is.

$$RF_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Transformation matrix for reflection about XZ-plane is.

$$RF_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LECTURE 16: THE VIEWING AND WINDOW

Window: A window defines a rectangular area in world coordinates for display is called a window. In computer graphics, a window is a graphical control element. It consists of a visual area containing some of the graphical user interface of the program it belongs to and is framed by a window decoration.

Viewport: A viewport is a polygon viewing region in computer graphics. The viewport is an area expressed in rendering-device-specific coordinates, e.g. pixels for screen coordinates, in which the objects of interest are going to be rendered. A viewport defines in normalized coordinates a rectangular area on the display device where the image of the data appears.

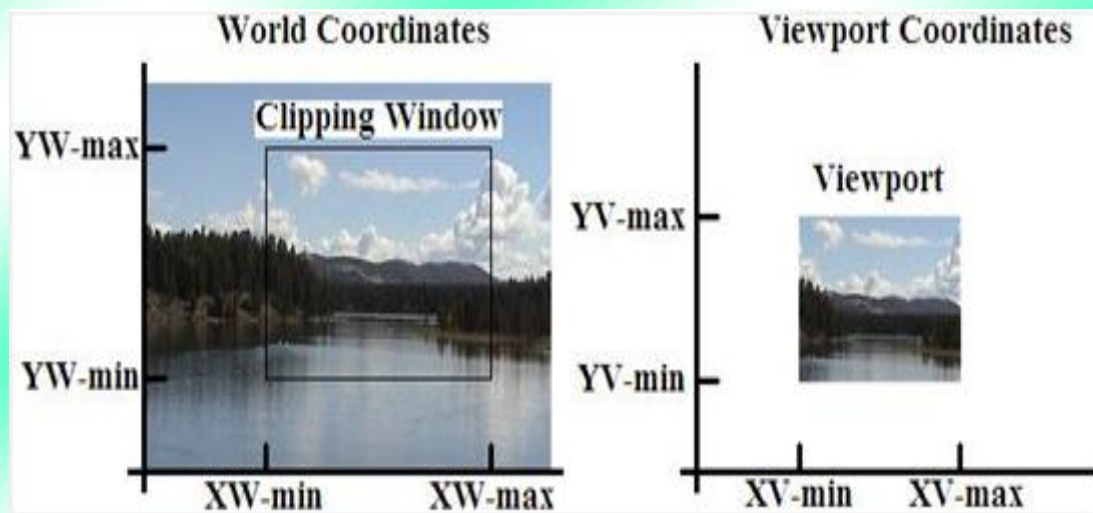
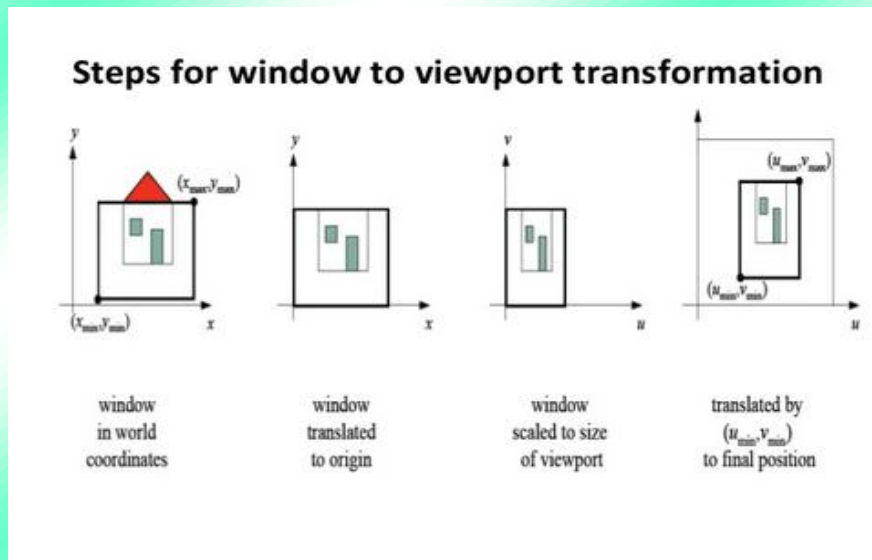


Figure 1 window and clipping

Viewing transformation: Is the transformation process from a world coordinates to device coordinates of viewport. In many cases window and viewport are rectangles, also other shapes may be used as window and viewport. In general, the clipping window selects the part of the scene that is to be displayed. The viewport then positions the clip on the output device.



-CLIPPING

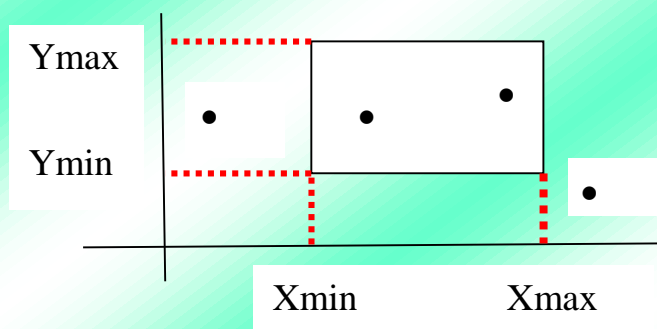
Is the possibility to draw part of drawing in 2D pane with minimum on the values (0,0) and (Xmax,Ymax) on the screen. This part is inside the area viewport, whereas the rest of the drawing which is outside viewport is clipped (is not appear on the screen). Note that clipping takes a part of time when we use it with any program, because it needs a group of tests in order to specify what part to display and what part to delete.

- POINT CLIPPING ALGORITHM

1. Get the minimum and maximum coordinates of both viewing pane.
2. The pane coordinate are (Xmin,Ymin) and (Xmax,Ymax).
3. Get the coordinates for a point (x,y).

If $X_{min} \leq x \leq X_{max}$ & $Y_{min} \leq y \leq Y_{max}$ then clip the point

Else discard the point.



-C++ program for point clipping Algorithm

```
#include<stdio.h>

// Function for point clipping
void pointClip(int XY[][2], int n, int Xmin, int Ymin,
               int Xmax, int Ymax)
{
    /***** Code for graphics view
    // initialize graphics mode
    detectgraph(&gm,&gr);
    initgraph(&gm,&gr,"g:\\tc\\BGI");
    for (int i=0; i<n; i++)
    {
        if ( (XY[i][0] >= Xmin) && (XY[i][0] <= Xmax))
        {
            if ( (XY[i][1] >= Ymin) && (XY[i][1] <= Ymax))
                putpixel(XY[i][0],XY[i][1],3);
        }
    }
    *****/
    printf ("Point inside the viewing pane:\n");
    for (int i=0; i<n; i++)
    {
        if ((XY[i][0] >= Xmin) && (XY[i][0] <= Xmax))
        {
            if ((XY[i][1] >= Ymin) && (XY[i][1] <= Ymax))
                printf ("%d, %d ", XY[i][0], XY[i][1]);
        }
    }
}

// print point coordinate outside viewing pane
printf ("\nPoint outside the viewing pane:\n");
for (int i=0; i<n; i++)
{
    if ((XY[i][0] < Xmin) || (XY[i][0] > Xmax))
        printf ("%d, %d ", XY[i][0], XY[i][1]);
    if ((XY[i][1] < Ymin) || (XY[i][1] > Ymax))
        printf ("%d, %d ", XY[i][0], XY[i][1]);
}

// Driver code
```

```
int main()
{
    int XY[6][2] = {{10,10}, {-10,10}, {400,100},
                   {100,400}, {400,400}, {100,40}};

    // getmaxx() & getmaxy() will return Xmax, Ymax
    // value if graphics.h is included
    int Xmin = 0;
    int Xmax = 250;
    int Ymin = 0;
    int Ymax = 250;
    pointClip(XY, 6, Xmin, Ymin, Xmax, Ymax);
    return 0;
}
```

LECTURE 17: LINE CLIPPING

-LINE CLIPPING

Lines that do not intersect the clipping window are either completely inside the window or completely outside the window. On the other hand a line that intersects the clipping window is divided by the intersection point (s) into segments that are either inside or outside the window. The following algorithm provide efficient way to decide the relationship between an arbitrary line and the clipping window to find intersection point (s).

-COHEN-SUTHERLAND LINE CLIPPING ALGORITHM

In this algorithm we divide the line clipping into two phases:

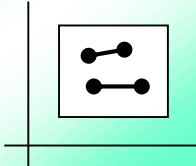
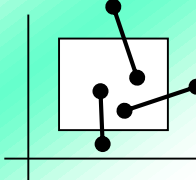
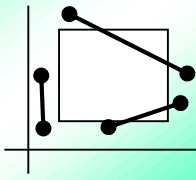
1- Identify those lines which intersect the clipping window and so need to be clipped.

2- Perform the clipping. All lines fall into one of the following clipping categories:

1- Visible: both end points of the line lie within the window.

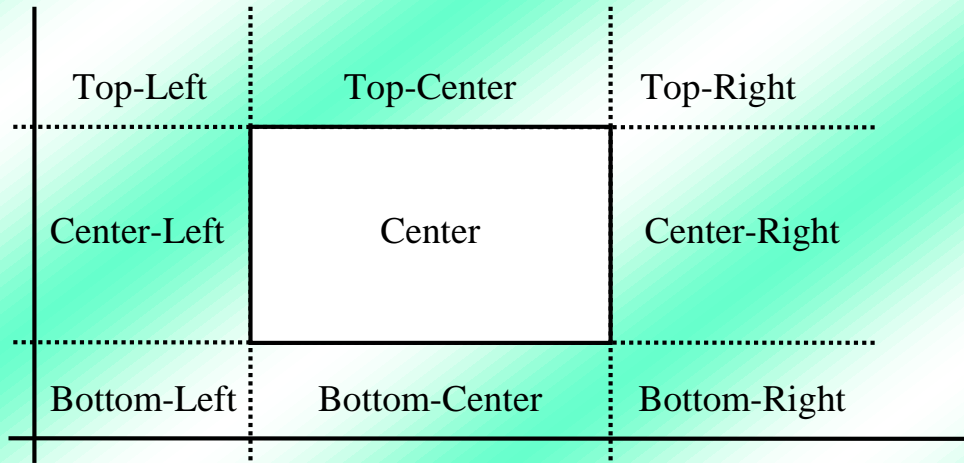
2- Not visible: the line lies outside the window. This will occur if the line from (x_1, y_1) to (x_2, y_2)

This involves developing more powerful clipping algorithms that can be determine the visible and invisible portions of such picture elements.

Situation	Solution	Example
Both end-points inside the window	Don't clip	
One end-point inside the window, one outside	Must clip	
Both end-points outside the window	Don't know	

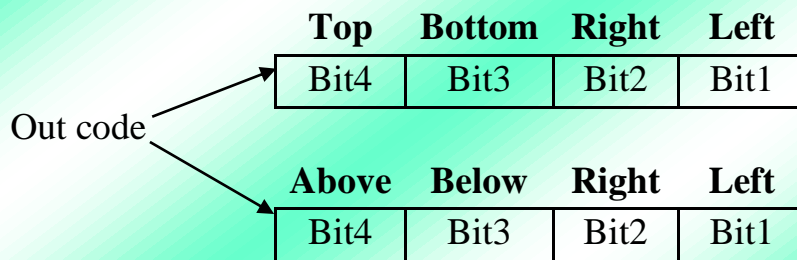
-Region and region code

At first the algorithm divides the region outside and the window to 9 segments as follows



Then define these nine areas and represent them with a 4-bit code called an out code. The following sequence is used to find the out code as follows:

Left	Bit1=1 if P positioned on the left of the window
Right	Bit2=1 if P positioned on the right of the window
Bottom/Below	Bit3=1 if P positioned on the bottom of the window
Top/Above	Bit4=1 if P positioned on the top of the window



1001	1000	1010
0001	0000	0010
0101	0100	0110

The window edges are (x_{\min}, y_{\min}) to (x_{\max}, y_{\max})

-x is constant for horizontal coordinates, for left $x=x_{\min}$ and $x=x_{\max}$ for right

- y is constant for vertical coordinates, for bottom $y=y_{\min}$ and $y=y_{\max}$ for top

-THE ALGORITHM STEPS

To perform the clipping operation on the line whose head is P1 (x_1, y_1) and P2 (x_2, y_2)

1- Calculate slope $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

2- Calculate the out code for both points, ABRL1 and ABRL2 as follows:

If $x < x_{\min} \Rightarrow \text{bit1} = 1 \Rightarrow (\text{left}) \quad \text{***1}$

If $x > x_{\max} \Rightarrow \text{bit2} = 1 \Rightarrow (\text{right}) \quad \text{**1*}$

If $y < y_{\min} \Rightarrow \text{bit3} = 1 \Rightarrow (\text{below}) \quad \text{*1**}$

If $y > y_{\max} \Rightarrow \text{bit4} = 1 \Rightarrow (\text{above}) \quad \text{1***}$

3. For full acceptance of the rectangle within the window (no clipping), work between the points P1 and P2 and the output must be equal to 0000.

4. For complete rejection of the rectangle outside the window (a total clip), AND works between the points P1 and P2 and the output must be equal to 0000.

5 - If the above conditions are not met in 3 and 4, a partial clip of the line must be made by calculating the intersection of each point with the boundary were 1 appears in the out code as follows:

If left: Make $x = x_{\min}$ and calculate $y = m(x - x_1) + y_1$

If Right: Make $x = x_{\max}$ and calculate $y = m(x - x_1) + y_1$

In the case of Below: Make $y = y_{\min}$ and calculate $x = 1/m(y - y_1) + x_1$

In the case of Above: Make $y = y_{\max}$ and account $x = 1/m(y - y_1) + x_1$

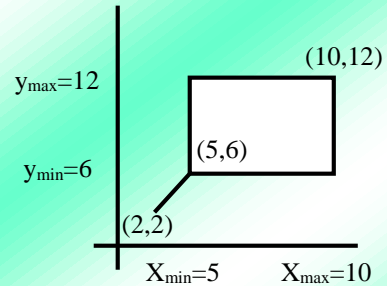
If more than 1 appears in the out code of the point, the above calculations must be made one after one.

Example : Using Cohen-Sutherland algorithm to clip the line defined by points P1 (5,6) and P2 (2,2) on the window known as V1 (5,6) and V2 (10,12)?

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2 - 6}{2 - 5} = \frac{4}{3}$$

out code for P₁

$$\left. \begin{array}{l} x < x_{\min} \Rightarrow 5 < 5 \Rightarrow \text{bit1} = 0 \\ x > x_{\max} \Rightarrow 5 > 10 \Rightarrow \text{bit2} = 0 \\ y < y_{\min} \Rightarrow 6 < 6 \Rightarrow \text{bit3} = 0 \\ y > y_{\max} \Rightarrow 6 > 12 \Rightarrow \text{bit4} = 0 \end{array} \right\} P_1 = 0000$$



out code for P₂

$$\left. \begin{array}{l} x < x_{\min} \Rightarrow 2 < 5 \Rightarrow \text{bit1} = 1 \\ x > x_{\max} \Rightarrow 2 > 10 \Rightarrow \text{bit2} = 0 \\ y < y_{\min} \Rightarrow 2 < 6 \Rightarrow \text{bit3} = 1 \\ y > y_{\max} \Rightarrow 2 > 12 \Rightarrow \text{bit4} = 0 \end{array} \right\} P_2 = 0101$$

- Acceptance Test

$$P_1 = 0000$$

$$\frac{P_2 = 0101}{0101} \vee \Rightarrow \text{The result} \neq 0000 \text{ is not total accept}$$

- Reject Test

$$P_1 = 0000$$

$$\frac{P_2 = 0101}{0000} \wedge \Rightarrow \text{The result} = 0000 \text{ is not total reject}$$

From the result there is a **partial clip** calculated from code ABRL₂

Left for P₂

$$\begin{aligned} x &= x_{\min} = 5 \\ y &= m(x - x_1) + y_1 \\ &= \frac{4}{3}(5 - 5) + 6 = 6 \\ P'_2 &(5,6) \end{aligned}$$

Bottom for P_2

$$y = y_{\min} = 6$$

$$x = \frac{1}{m}(y - y_1) + x_1$$

$$= \frac{3}{4}(6 - 2) + 2 = 5$$

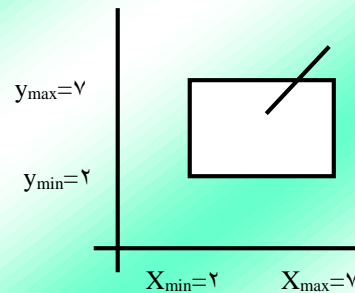
$$P'_2 (5,6)$$

Example : Using the Cohen-Sutherland algorithm to clip the line defined by points $P_1 (4,5)$ and $P_2 (7,8)$ on the rectangular region known as $V_1 (2,2)$ and $V_2 (7,7)$?

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{8 - 5}{7 - 4} = \frac{3}{3} = 1$$

out code for P_1

$$\left. \begin{array}{l} x < x_{\min} \Rightarrow 4 < 2 \Rightarrow \text{bit1} = 0 \\ x > x_{\max} \Rightarrow 4 > 7 \Rightarrow \text{bit2} = 0 \\ y < y_{\min} \Rightarrow 5 < 2 \Rightarrow \text{bit3} = 0 \\ y > y_{\max} \Rightarrow 5 > 7 \Rightarrow \text{bit4} = 0 \end{array} \right\} P_1 = 0000$$



out code for P_2

$$\left. \begin{array}{l} x < x_{\min} \Rightarrow 7 < 2 \Rightarrow \text{bit1} = 0 \\ x > x_{\max} \Rightarrow 7 > 7 \Rightarrow \text{bit2} = 0 \\ y < y_{\min} \Rightarrow 8 < 2 \Rightarrow \text{bit3} = 0 \\ y > y_{\max} \Rightarrow 8 > 7 \Rightarrow \text{bit4} = 1 \end{array} \right\} P_2 = 1000$$

-Acceptance Test

$$P_1 = 0000$$

$$\frac{P_2 = 1000 \vee}{1000} \Rightarrow \text{The result} \neq 0000 \text{ is not totally accepted}$$

-Reject Test

$$P_1 = 0000$$

$$\frac{P_2 = 1000 \wedge}{0000} \Rightarrow \text{The result} = 0000 \text{ is not totally reject}$$

From the result there is a **partial clip** calculated from code ABRL

Top for P₂

$$\begin{aligned}y &= y_{\max} = 7 \\x &= \frac{1}{m}(y - y_1) + x_1 \\&= 1(7-8)+7 = -1+7 = 6 \\P'_2 &(6,7)\end{aligned}$$

Example : Using the Cohen-Sutherland algorithm to clip the lines defined in line A by points P1 (6,8) and P2 (8,9)

line B by points P1 (6,8) and P2 (7,14)

line C by points P1 (8,9) and P2 (6,14)

with the window defined by vertexes V1 (2,2) and V2 (10,12)?

$$m_A = \frac{y_2 - y_1}{x_2 - x_1} = \frac{9 - 8}{8 - 6} = \frac{1}{2} = 0.5$$

$$m_B = \frac{y_2 - y_1}{x_2 - x_1} = \frac{14 - 8}{7 - 6} = \frac{6}{1} = 6$$

$$m_C = \frac{y_2 - y_1}{x_2 - x_1} = \frac{9 - 14}{8 - 7} = \frac{-5}{1} = -5$$

out code for line **A** P₁, P₂

$$x < x_{\min} \Rightarrow 6 < 2 \Rightarrow \text{bit1} = 0 \quad 8 < 2 \Rightarrow \text{bit1} = 0 \quad \text{out code P1} = 0000$$

$$x > x_{\max} \Rightarrow 6 > 10 \Rightarrow \text{bit2} = 0 \quad 8 > 10 \Rightarrow \text{bit2} = 0 \quad \text{out code P2} = 0000$$

$$y < y_{\min} \Rightarrow 8 < 2 \Rightarrow \text{bit3} = 0 \quad 9 < 2 \Rightarrow \text{bit2} = 0$$

$$y > y_{\max} \Rightarrow 8 > 12 \Rightarrow \text{bit4} = 0 \quad 9 > 12 \Rightarrow \text{bit2} = 0$$

out code for line **B** P₂

$$x < x_{\min} \Rightarrow 7 < 2 \Rightarrow \text{bit1} = 0 \quad \text{out code P1} = 1000$$

$$x > x_{\max} \Rightarrow 7 > 10 \Rightarrow \text{bit2} = 0$$

$$y < y_{\min} \Rightarrow 14 < 2 \Rightarrow \text{bit3} = 0$$

$$y > y_{\max} \Rightarrow 14 > 12 \Rightarrow \text{bit4} = 1$$

- Acceptance Test

$$P_1 = 0000$$

$$\frac{P_2 = 0000}{0000} \vee \Rightarrow \text{The result of line A} = 0000 \text{ so it is } \mathbf{\text{totally accepted}}$$

$$P_1 = 0000$$

$$\frac{P_2 = 1000}{1000} \vee \Rightarrow \text{The result of line B} \neq 0000 \text{ so it is not accepted}$$

-Reject Test

$$P_1 = 0000$$

$$\frac{P_2 = 1000}{0000} \wedge \Rightarrow \text{The result} = 0000 \text{ is not totally rejected}$$

*From the result there is a **partial clip** calculated from code ABRL*

Top for P_2 1000

$$y = y_{\max} = 7$$

$$x = \frac{1}{m}(y - y_1) + x_1, \quad y=12$$

$$= 1/6(12-6)+8 = 8$$

P'_2 (12,8)

LECTURE 18: POLYGONS CLIPPING

-CLIPPING POLYGONS

An algorithm that clips a polygon must deal with many different cases. The case is particularly in that the concave polygon is clipped into two separate polygons. Edges of polygon need to be tested against clipping rectangle edge new edges may be added. The 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. There are several well-known polygon clipping algorithms, each having its advantages and drawback. The oldest one (from 1974) is called the Sutherland-Hodgman algorithm. In its basic form, it is relatively simple. It is also very efficient.

-SUTHERLAND AND HODGMAN'S POLYGON-CLIPPING ALGORITHM

1. Uses a divide-and-conquer strategy: It solves a series of simple and identical problems that, when combined, solve the overall problem.
2. The algorithm works clockwise.
3. First, it clips the polygon against the right clipping boundary.
4. The resulting, partly clipped polygon is then clipped against the top boundary, and then the process is repeated for the two remaining boundaries. (Of course, it also works in another order.)

Note the difference between the strategy for a polygon clipping and the Cohen-Sutherland line clipping algorithm is that 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 increase.

-Outline for Sutherland-Hodgman Algorithm

Input:

- v_1, v_2, \dots, v_n the vertices defining the polygon
- Single infinite clip edge w/ inside/outside info

Output:

- v'_1, v'_2, \dots, v'_m , vertices of the clipped polygon
- Do this 4 (or ne) times
- Traverse vertices (edges)
- Add vertices one-at-a-time to output polygon
- Use inside/outside info

For each polygon $P \ P' = P$

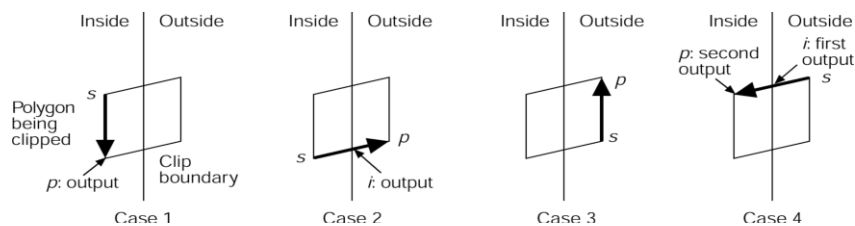
– for each clipping edge (there are 4) {

Clip polygon P' to clipping edge

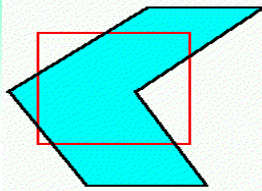
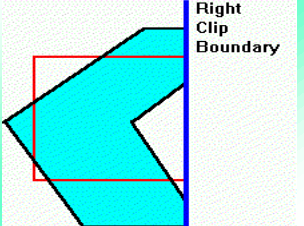
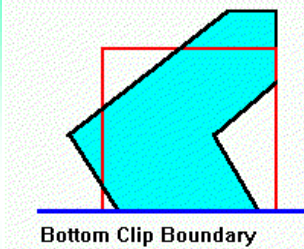
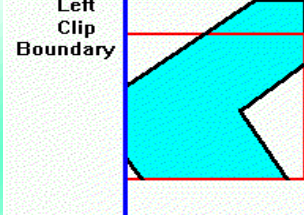

– for each edge in polygon P' ***Four Cases of polygon clipping against one edge***

- Check clipping cases (there are 4)
- Case 1 : Output v_{i+1}
- Case 2 : Output intersection point
- Case 3 : No output
- Case 4 : Output intersection point

& v_{i+1} }



- Step by step polygon clipping algorithm

Case	Clip
The original polygon and rectangle	
After clipped by the right clip boundary	
After clipped by the right and bottom clip boundaries	
After clipped by the right, bottom, and left clip boundaries	
After clipped top boundary The final clip	

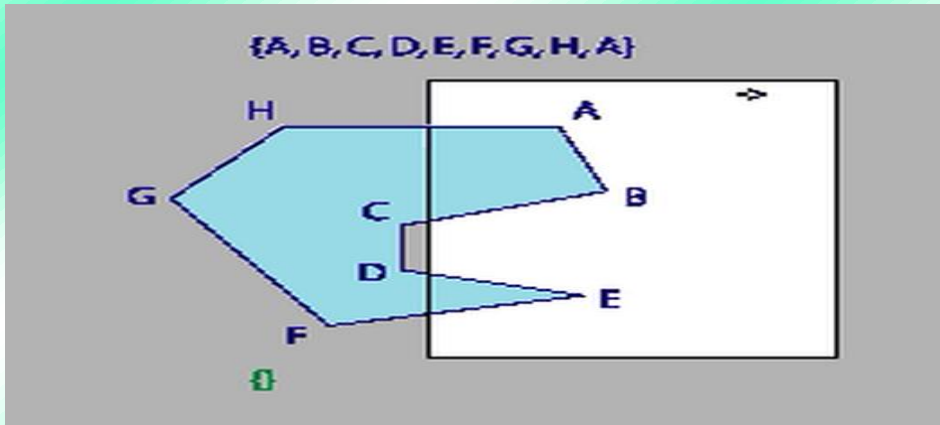


Figure 1 original polygon

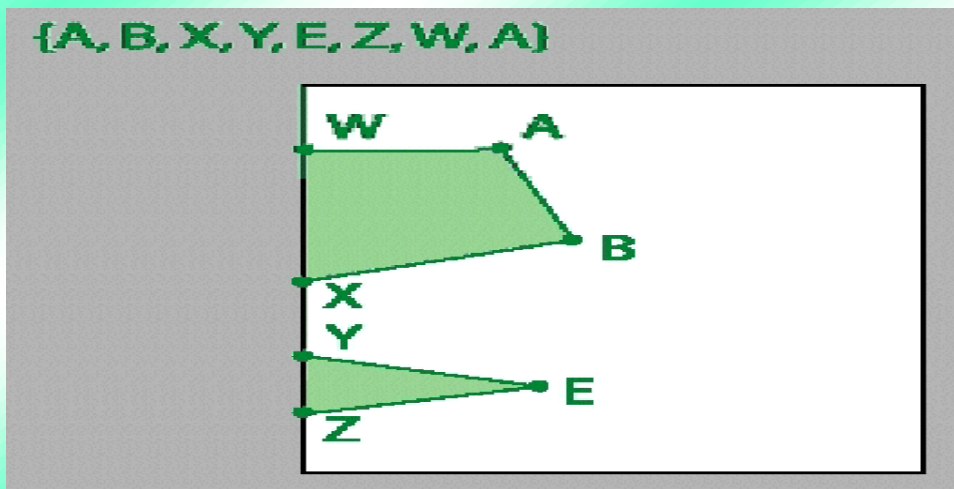


Figure 2 final result

-Clipping a concave polygon can produce two CONNECTED areas

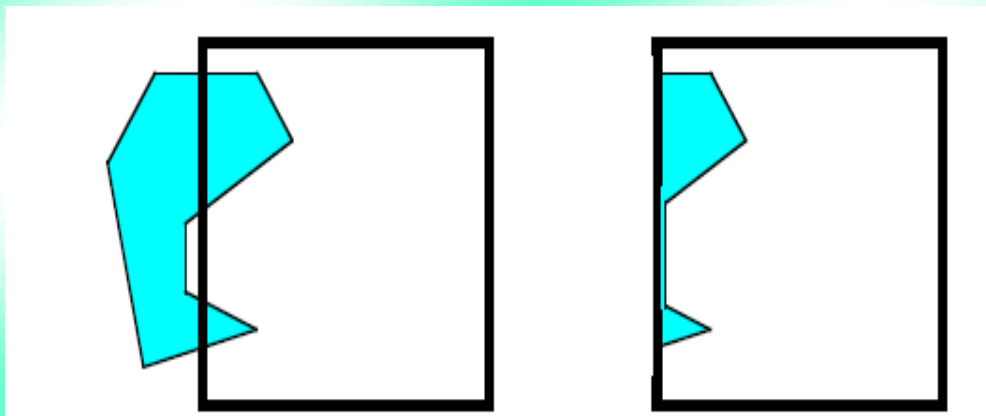


Figure 3 concave polygon

LECTURE 19: COMPUTER ANIMATION

-PREVIEW

Animation means giving life to any object in computer graphics. It has the power of injecting energy and emotions into the most seemingly inanimate objects. Objects can have human features as identity, character, gender, mood, intention, emotion. Computer-assisted animation and computer-generated animation are two categories of computer animation. It can be presented via film or video. The basic idea behind animation is to play back the recorded images at the rates fast enough to fool the human eye into interpreting them as continuous motion. Animation can make a series of dead images come alive. Animation can be used in many areas like entertainment, computer aided-design, scientific visualization, training, education, e-commerce, and computer art.

-HISTORY OF COMPUTER ANIMATION

- 1891: Thomas Edison invented the motion picture projector
- 1896: Georges Melies made objects appear, disappear, change shape using camera tricks
- 1900: smoke is animated in a scene
- 1906: first animated cartoon (J. Stuart Blackton)
- 1915: Max Fleischer patented *rotoscoping* (drawing images on cels by tracing over previously recorded live action)
- Walt Disney use of storyboard to review the story
- Pencil sketches to review motion
- Sound and color in animation
- (1928) First use of sound in animation
- Evans and Sutherland: '60s and '70s first interactive computer graphics programs
- 3D animations emerge in 1980s and 1990s

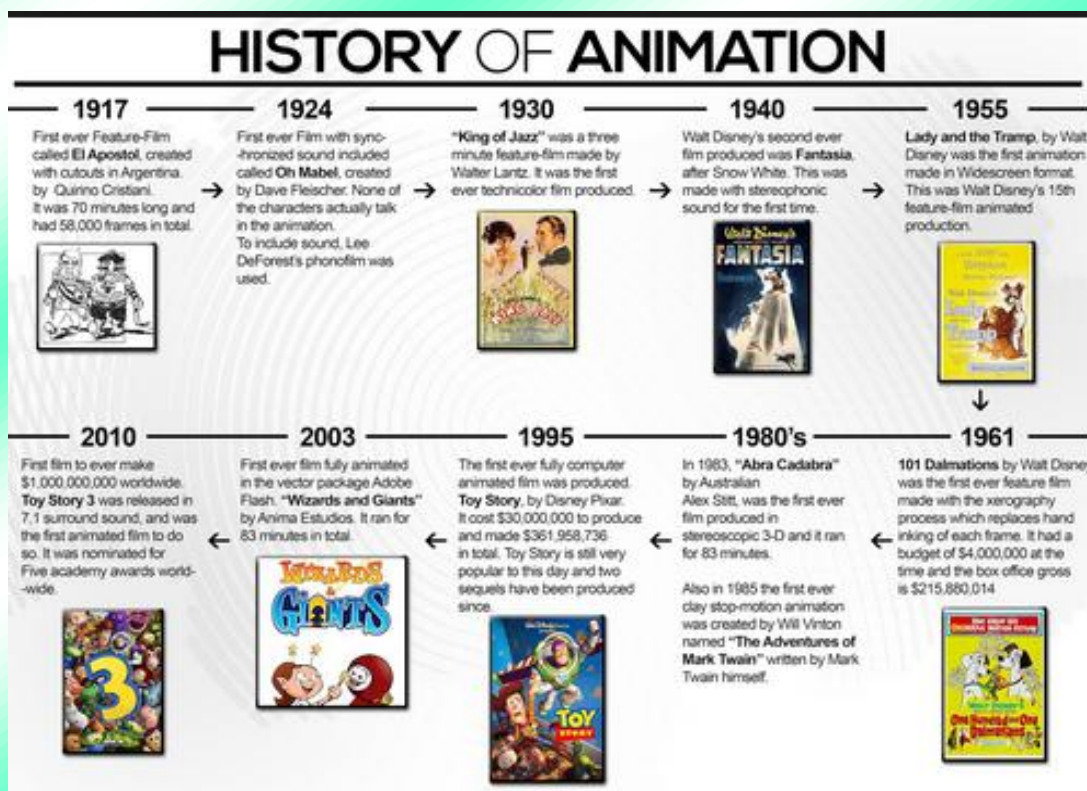
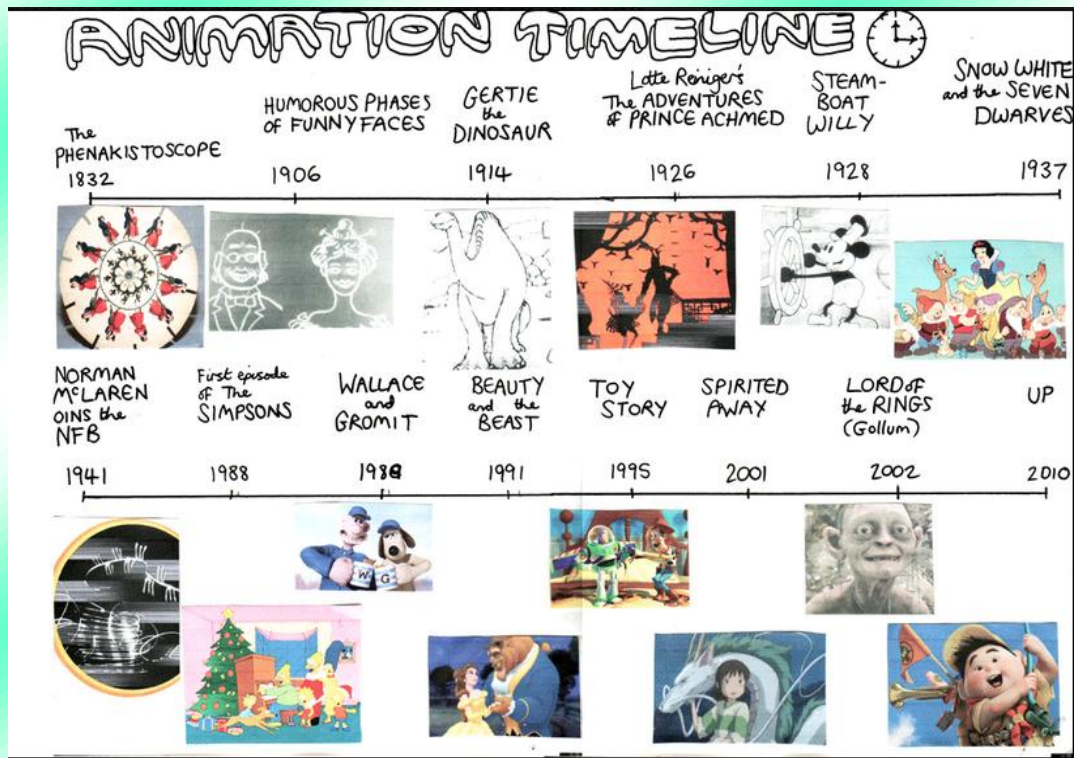


Figure 1 Animation history

-ANIMATION TECHNIQUES

Animators have invented and used a variety of different animation techniques. Basically there are six animation technique discussed in follows.

1-Traditional Animation frame by frame

Traditionally most of the animation was done by hand. All the frames in an animation had to be drawn by hand. Since each second of animation requires 24 frames *film*, the amount of efforts required to create even the shortest of movies can be tremendous.

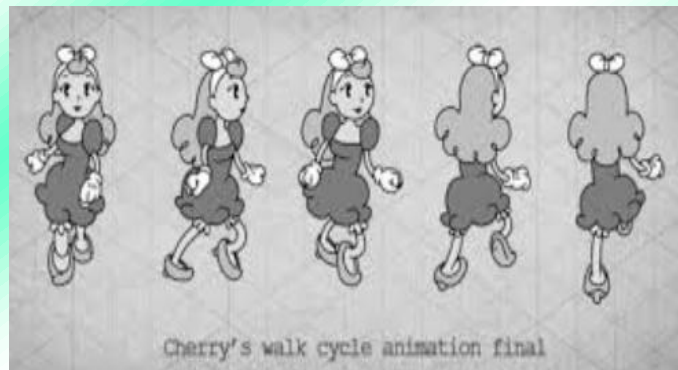


Figure 2 traditional animation

2-Key framing

A keyframe is a frame where we define changes in animation. Every frame is a keyframe when we create frame by frame animation. When someone creates a 3D animation on a computer, they usually don't specify the exact position of any given object on every single frame. They create keyframes. Keyframes are important frames during which an object changes its size, direction, shape or other properties. The computer then figures out all the in-between frames and saves an extreme amount of time for the animator. The following illustrations depict the frames drawn by user and the frames generated by computer. In this technique, a storyboard is laid out and then the artists draw the major frames of the animation. Major frames are the ones in which prominent changes take place. They are the key points of animation. Keyframing requires that the animator specifies critical or key positions for the objects. The computer then automatically fills in the missing frames by smoothly interpolating between those positions.

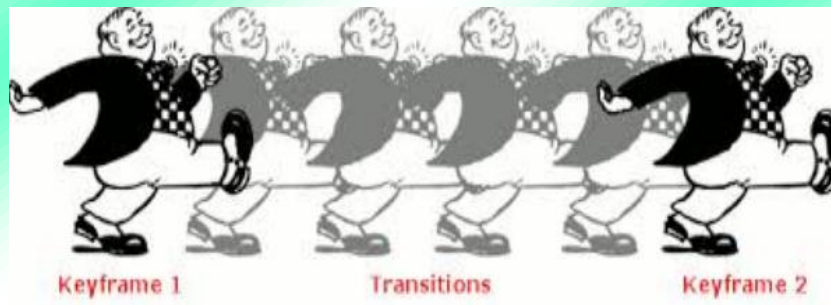


Figure 3 keyframing animation

3-Procedural

In a procedural animation, the objects are animated by a procedure – a set of rules – not by keyframing. The animator specifies rules and initial conditions and runs simulation. Rules are often based on physical rules of the real world expressed by mathematical equations.

4-Behavioral

In behavioral animation, an autonomous character determines its own actions, at least to a certain extent. This gives the character some ability to improvise, and frees the animator from the need to specify each detail of every character's motion.

5-Performance Based Motion Capture

Another technique is Motion Capture, in which magnetic or vision-based sensors record the actions of a human or animal object in three dimensions. A computer then uses these data to animate the object. This technology has enabled a number of famous athletes to supply the actions for characters in sports video games. Motion capture is pretty popular with the animators mainly because some of the commonplace human actions can be captured with relative ease. However, there can be serious discrepancies between the shapes or dimensions of the subject and the graphical character and this may lead to problems of exact execution.

6-Physically Based Dynamics

Unlike key framing and motion picture, simulation uses the laws of physics to generate motion of pictures and other objects. Simulations can be easily used to produce slightly different sequences while maintaining physical realism. Secondly, real-time simulations allow a higher degree of interactivity where the real person can maneuver the actions of the simulated character. In contrast the

applications based on key-framing and motion select and modify motions from a pre-computed library of motions. One drawback that simulation suffers from is the expertise and time required to handcraft the appropriate controls systems.



Figure 4 physical based dynamics

-Morphing

The transformation of object shapes from one form to another form is called morphing. It is one of the most complicated transformations. A morph looks as if two images melt into each other with a very fluid motion. In technical terms, two images are distorted and a fade occurs between them.



Figure 5 morphing