Unit-1. CGM:

~~Pixel:~~

Computer graphics is the use of computer to create and manipulate images. It is the intersection of design and computer science.

Pixel:

It is the smallest unit of a digital image on display and stands for picture element. It is very small isolated dot that stands for one color and plays the most basic part in digital image.

Resolution:

Resolution is a measurement of number of pixels -- picture elements or individual point of color - that can be contained or a display screen or in a camera sensor. In practical term it describes the sharpness or clarity of an image or picture.

Expressed in terms of number of pixels that can be displayed both horizontally and vertically.

PPI - Pixels Per inch:

PPI is a measurement of how many pixels are in each inch of a digital image. It's also used to describe the resolution of a computer screen.

Higher PPI → better image quality.

Lower PPI → larger pixels, makes image look blocky.

Aspect Ratio:

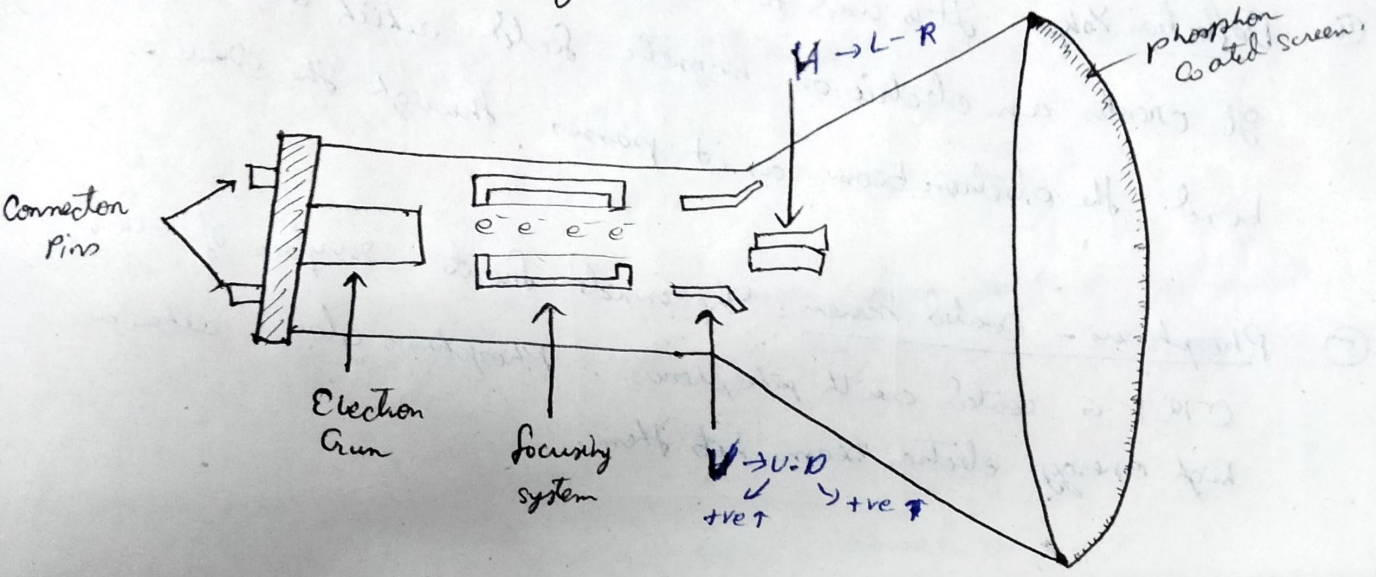
It is proportional relationship between the width of a video image compared to its height. It is usually expressed as width : height

Some come aspect ratio = 16 : 9
9 : 16
1 : 1

Frame buffer:

A Frame buffer is a region of memory that stores the data for an image that appears on a computer screen.

CRT : Cathode Ray Tube:



- Is a technology used in traditional computer monitors and televisions. The image of CRT display is created by firing electron from the back of the tube of phosphorus located towards the front of the screen. Once e⁻ heats the phosphorus, they light up, & they are projected on a screen. The color we view on the screen is produced by a blend of red, blue and green light.

Components:

① **Electron Gun:** Consisting of a series of elements, primarily a heating filament and a cathode. The electron gun creates a source of electrons which are focused into a narrow beam directed at the face of the CRT.

② Control Electrode: It is used to turn the electron beam on or off.

③ Focusing system: It is used to create a clear picture by focusing the electrons into a narrow beam.

④ Deflection Yoke: It is used to control the direction of electron beam. It creates an electric or magnetic field which will bend the electron beam as it passes through the area.

⑤ Phosphorous - Coated screen: The inside front surface of every CRT is coated with phosphorous. Phosphors glow when a high energy electron beam hits them.

Vector Scan Display:

It uses an electron beam which operates like a pencil to create a line image on the CRT screen. The picture is created / constructed out of a sequence of straight line segments. Each line segment is drawn on the screen by directing the beam to move from one point on the screen to the next, where its x & y coordinates define each point. After drawing the picture, the system cycles back to first line and design all the lines of the image 30 to 60 time each second.

Advantage – ① A CRT has the e⁻ beam directed only to the parts of the screen where image is to be drawn.

② Produce smooth line drawings.

③ High resolution.

Disadvantage :

① Cannot display realistic shaded scenes.

Procedure :

① The display receives instruction that describes lines or shapes.

(eg. draw a line from $(x1, y1)$ to $(x2, y2)$;

② Electron Beam Positioning:

The electron beam is directed to start coordinate $(x1, y1)$ without drawing.

③ Line Drawing:

The beam is turn on and moves from start to end point $(x2, y2)$ drawing a bright line on screen.

④ Beam turn off:

After reaching endpoint, the beam is turned off.

⑤ Repetition for Next Line:

The process is repeated for all lines in the picture until entire image is displayed.

⑥ Refresh Cycle:

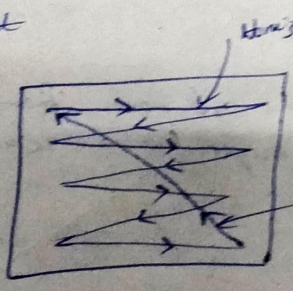
Display refreshes many times per seconds (20 - 60) to maintain stable image.

* Raster Scan Display:

It is based on intensity control of pixel in the form of a rectangle box called Raster on the screen. Information of on and off pixels is stored in refresh buffer or frame buffer. The raster scan system can store info. of each pixel position so it is suitable for realistic display of objects. It provides a refresh rate of 60 to 80 frames per second.

Frame buffer also known as Raster or bit map. Its Buffer the positions are called picture elements or pixels. Beam refreshing is of two types [Horizontal retracing.
Vertical retracing.

Vertical retracing: When the beam starts from the top left corner & reaches the bottom right side, it will again return to top left side.



Horizontal — Top to bottom

Advantage — ① Realistic image
② Different colors to be generated
③ Shadow scenes possible

Disadvantage : ① Low Resolution
② Expensive.

CGM

SCAN Conversion:

It is a process of representing graphics objects a collection of pixels. The graphics object are continuous, the pixel used are discrete. Each pixel can have either on or off state.

It is a technique for changing the vertical / horizontal scan frequency of video signal for diff purposes and applications. The device which performs this conversion is called scan converter. Scan conversion is also known as Rasterization.

Two method for scan conversion:

① Analog Method:

The conversion is done using large number of delay cells and is appropriate for analog video.

② Digital method:

The picture is stored in frame buffer with x1 speed and is read with x2 speed.

Scan Converting a line:

① Direct Method.

② DDA (Digital Differential Analyzer)

③ Bresenham's Algo.

Algo for direct

Step 1- Read two end pts $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$

Step 2- Calculate :

$dx = x_2 - x_1$

$dy = y_2 - y_1$

Step 3- Now,

$m = dy/dx$

step 4. Set (x, y) to starting pt.

if $dx > 0$ then

$x = x_1$

$y = y_1$

$x_{end} = x_2$

if $dy < 0$ then.

$x = x_2$

$y = y_2$

$x_{end} = x_1$

Step 5. Now calculate

$c = y - mx$.

step 6. Plot a pt at current (x, y) coordinates.

Step 7. Increment value of x.

$x = x + 1$

Step 8- Compute value of y.

$y = mx + c$

step 9- If $x = x_{end}$ then

stop.

otherwise go to step 6.

```
#include<stdio.h>
#include<graphics.h>
void main(){
    float x1, x2, y1, y2, xe, dy, dx, m, c;
    int gd=DETECT, gm; initgraph(&gd,&gm,"C:\\TC\\BGI");
    printf("Enter the first coordinate: ");
    scanf("%d %d",&x1,&y1);
    printf("Enter the second coordinate: ");
    scanf("%d %d",&x2,&y2);
    dx= x2-x1;
    dy= y2-y1;
    m= dy/dx;
    if(dx>0){
        x=x1;
        y=y1;
        xe=x2;
    }
    if(dx<0){
        x=x2;
        y=y2;
        xe=x1;
    }
    c= y-m*x;
    while(x<=xe){
        putpixel(x,y,GREEN);
        x=x+1;
        y=m*x+c;
    }
    getch();
    closegraph();
}
```

DDA Algo -

Step 1 - Read two end pts $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$

Step 2 - Find approximate length of line.

if (abs $(x_2 - x_1)$ > abs $(y_2 - y_1)$) then

length = abs $(x_2 - x_1)$

else

length = abs $(y_2 - y_1)$.

Step 3 - Find pixel unit.

dx = $(x_2 - x_1)$ / length.

dy = $(y_2 - y_1)$ / length.

Step 4 - set $x = x_1$, $y = y_1$ & i = 0.

Step 5 - Now plot pt (x, y)

x = x + dx.

y = y + dy;

Step 6 ; Repeat step 5 until i <= length

Step 7 - Stop.

Bresenham Line Drawing -

Step 1 - Read two end pt $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$

Step 2 - Calculate.

$dx = x_2 - x_1$,

$dy = y_2 - y_1$;

Step 3 - Calculate decision parameter p:

$p = 2dy - dx$.

Step 4 - Set initial pt:

$x = x_1$, $y = y_1$

& $i = 0$.

Step 5 - Now plot the pt (x, y)

if $p < 0$ then,

$x = x + 1$

$p = p + 2dy$

otherwise

$x = x + 1$;

$y = y + 1$;

$p = p + 2dy - 2dx$.

Step 6 - Repeat step 5 until $i <= dx$.

Bresenham's Circle Drawing algo :

Step 1: Input center of circle (p, q)

Step 2: Input radius of circle r.

Step 3: Calculate.

$$d = 3 - 2r.$$

Step 4: Initialize,

$$x = 0$$
$$y = r.$$

Step 5: Now center is at (p, q) & current pixel is (x, y) So plot eight pts by using eight way symmetry concept.

putpixel $(x+p, y+q)$
" $(x+p, -y+q)$
" $(-x+p, -y+q)$
" $(-x+p, y+q)$
" $(y+p, x+q)$
" $(y+p, -x+q)$
" $(-y+p, -x+q)$
" $(-y+p, x+q)$

Step 6: Find pixel location;

if $d <= 0$ then,
$x = x+1;$
$d = d + 4x + 6.$

else $x = x+1;$
$y = y-1;$
$d = d + 4(x-y) + 10;$

Step 7: If $x <= y$ then go to step 5

end.

Midpoint circle drawing

Step 1:- Input center of circle (xc, yc)

Step 2:- Input radius of circle r.

Step 3:- Calculate decision parameter.

$p = 1 - r$,

Step 4:- Initialize first pt.

$x = 0$

$y = r$.

Step 5:- Now center is at (xc, yc) & current pixel is (x, y) so plot eight points by using eight way symmetry.

Concept.

putpixel (x+xc, y+yc)

+7.

Step 6:- Find next pixel,

if $p < 0$ then,

$x = x + 1$;

$p = p + 2 * x + 3$

else $x = x + 1$;

$y = y - 1$

$p = p + 2(x - y) + 5$

Step 7:- If $x < = y$ then.

go to step 5.

otherwise.

stop.

Boundary fill algo :-

```
Boundary (x, y, f, b)
{
 if (getpixel (x,y) != b
      &&
     getpixel (x,y) != f)
 { putpixel (x,y,f);
   boundary (x+1, y, f, b)
   boundary (x, y+1, f, b)
   boundary (x-1, y, f, b);
   boundary (x, y-1, f, b);
   boundary (x-1, y-1, f, b);
   boundary (x-1, y+1, f, b);
   boundary (x+1, y-1, f, b);
   boundary (x+1, y+1, f, b);
 }
}
```

x, y : coordinates
f : color of polygon
b : color boundary.

fills an area until it encounters a boundary color. It is useful when the region to be filled is enclosed by a distinct boundary color.

Flood fill.

o - old colon

```
flood (x, y, n, o) {
  if (getpixel (x, y) == o)
  {
    putpixel (x, y, n)
    flood (x+1, y, n, o)
    " (x, y+1, n, o)
    " (x-1, y, n, o)
      (x, y-1, n, o)
      (x-1, y-1, n, o)
      (x-1, y+1, n, o)
      (x+1, y-1, n, o)
      (x+1, y+1, n, o);
}
```

Fills an area based on the color of the starting pixel. It replaces pixels of the target color with new color, spreading outwards in all direction until it encounters pixel of a diff. color.