# Techno India NJR Institute of Technology

# Course File

# Computer Graphics & Multimedia (5CS4- 04)

**Ayush Gupta**

Assistant Professor

**Department of CSE**

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwa
(Principal)

## 5CS4-04: Computer Graphics & Multimedia

**Credit: 3** **Max. Marks: 150(IA:30, ETE:120)**
**3L+0T+0P** **End Term Exam: 3 Hours**

| SN | Contents | Hours |
|---|---|---|
| 1 | **Introduction:** Objective, scope and outcome of the course. | 01 |
| 2 | **Basic of Computer Graphics:**Basic of Computer Graphics, Applications of computer graphics, Display devices, Random and Raster scan systems, Graphics input devices, Graphics software and standards | 06 |
| 3 | **Graphics Primitives:**Points, lines, circles and ellipses as primitives, scan conversion algorithms for primitives, Fill area primitives including scan-line polygon filling, inside-outside test, boundary and flood-fill, character generation, line attributes, area-fill attributes, character attributers. Aliasing, and introduction to Anti Aliasing (No anti aliasing algorithm). | 07 |
| 4 | **Two Dimensional Graphics:**Transformations (translation, rotation, scaling), matrix representation, homogeneous coordinates, composite transformations, reflection and shearing, viewing pipeline and coordinates system, window-to-viewport transformation, clipping including point clipping, line clipping (cohen-sutherland, liang- bersky, NLN), polygon clipping | 08 |
| 5 | **Three Dimensional Graphics:**3D display methods, polygon surfaces, tables, equations, meshes, curved lies and surfaces, quadric surfaces, spline representation, cubic spline interpolation methods, Bazier curves and surfaces, B-spline curves and surfaces.3D scaling, rotation and translation, composite transformation, viewing pipeline and coordinates, parallel and perspective transformation, view volume and general (parallel and perspective) projection transformations. | 08 |
| 6 | **Illumination and Colour Models:**Light sources – basic illumination models – halftone patterns and dithering techniques; Properties of light – Standard primaries and chromaticity diagram; Intuitive colour concepts – RGB colour model – YIQ colour model – CMY colour model – HSV colour model – HLS colour model; Colour selection. | 06 |
| 7 | **Animations &Realism:**Design of Animation sequences – animation function – raster animation – key frame systems – motion specification – morphing – tweening.<br><br>**ComputerGraphics Realism:** Tiling the plane – Recursively defined curves – Koch curves – C curves – Dragons – space filling curves – fractals – Grammar based models – fractals – turtle graphics – ray tracing. | 06 |
| | **Total** | **42** |

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

**Prerequisites:**
1. Basic mathematics including round off, floor and ceiling functions.
2. Basics of linear algebra.
3. Intermediate programming skills.
4. Understanding of basic geometric shapes.

**UNIT-I**

**Introduction:** Objective, Scope and Outcome of the Course

| **5CS404** | **Computer Graphics & Multimedia Year of study: 2019-20** |
|---|---|
| CO35404.1 | Students will be able to define the basics of computer graphics, different graphics systems, application of computer graphics and rasterisation of line, circle and ellipse. |
| CO35404.2 | Students will be able to apply geometric transformations on graphics objects, their application in composite form, different color filling algorithm and clipping algorithm. |
| CO35404.3 | Students will be able to identify visible surface detection techniques & curves. |
| CO35404.4 | Students will be able to render projected objects to naturalize the scene in 2D view and use of illumination models & color models. |
| CO35404.5 | Students will be able to identify multimedia components and animation techniques. |

**Mapping of Cos with Pos and PSOs:**

| **Computer Graphics & Multimedia Year of study: 2019-20** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Course Outcome** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO35404.1** | 2 | 3 | 3 | 1 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 1 | 1 |
| **CO35404.2** | 2 | 3 | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |
| **CO35404.3** | 2 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |
| **CO35404.4** | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 0 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |
| **CO35404.5** | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 2 | 1 | 1 |
| **C35404 (AVG)** | 2.20 | 2.60 | 2.60 | 2.40 | 2.40 | 0.40 | 0.40 | 0.00 | 0.20 | 0.00 | 0.80 | 1.60 | 2.00 | 1.00 | 1.00 |

Techno India NJR Institute of Technology
Academic Administration of Techno NJR Institute
Syllabus Deployment

| | | |
|---|---|---|
| Name of Faculty | : Akhilesh Deep Arya | Subject Code: 5CS4-04 |
| Subject | : Computer Graphics & Multimedia | |
| Department | : Computer Science and Engineering | Sem: V |
| Total No. of Lectures Planned: 47 | | |

COURSE OUTCOMES

At the end of this course students will be able to:

CO1: Students will be able understand the basics of computer graphics, different graphics systems, application of computer graphics and rasterization of line, circle and ellipse.

CO2: Students will be able to apply geometric transformations on graphics objects, their application in composite form, different color filling algorithm and clipping algorithm.

CO3: Students will be able to explore visible surface detection techniques& curves.

CO4: Students will be able to render projected objects to naturalize the scene in 2D view and use of illumination models for this& color models.

CO5: Students will be able to explore multimedia components and animation techniques.

| Lecture No./Hours | Unit | Topic |
|---|---|---|
| 1 | **1** | **Introduction:** Objective, scope and outcome of the course |
| 2 | **2** | Basic of Computer Graphics: Basic of Computer Graphics, Application |
| 3 | 2 | Display devices, CRT |
| 4 | 2 | Color Display CRT Beam Penetration and Shadow mask method |
| 5 | 2 | Random and raster scan method |
| 6 | 2 | Graphics input device |
| 7 | 2 | Graphics software and standards |

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

| | | |
|---|---|---|
| 8 | **3** | Graphics Primitives: Points, lines |
| 9 | 3 | Digital Differential analyzer line drawing |
| 10 | 3 | Bresenham line drawing |
| 11 | 3 | Mid-Point circle algorithm |
| 12 | 3 | Ellipses drawing algorithm |
| 13 | 3 | Fill area primitives including scan line polygon filling |
| 14 | 3 | Boundary and flood-fill |
| 15 | 3 | Aliasing, and introduction to Anti Aliasing |
| 16 | **4** | Two Dimensional Graphics: Transformations (translation, rotation, scaling), matrix representation |
| 17 | 4 | Homogeneous coordinates, composite transformations |
| 18 | 4 | Reflection and shearing |
| 19 | 4 | Viewing pipeline and coordinates system |
| 20 | 4 | Window-to-viewport transformation |
| 21 | 4 | Clipping including point clipping |
| 22 | 4 | Line clipping (cohen-sutherland, liang- bersky, NLN) |
| 23 | 4 | Polygon clipping |
| 24 | **5** | Three Dimensional Graphics:3D display methods |
| 25 | 5 | Polygon surfaces, tables, equations |
| 26 | 5 | Meshes, curved lies and surfaces |
| 27 | 5 | Quadric surfaces, spline representation |
| 28 | 5 | Cubic spline interpolation methods |
| 29 | 5 | Bazier curves and surfaces |
| 30 | 5 | B-spline curves and surfaces |
| 31 | 5 | 3D scaling, rotation and translation, composite transformation |
| 32 | 5 | Viewing pipeline and coordinates, parallel and perspective transformation |
| 33 | 5 | View volume and general (parallel and perspective) projection transformations |
| 34 | **6** | Illumination and Color Models: Light sources – basic illumination models |
| 35 | 6 | Halftone patterns and dithering techniques |
| 36 | 6 | Properties of light – Standard primaries and chromaticity diagram |
| 37 | 6 | Colour concepts – RGB colour model |
| 38 | 6 | YIQ colour model – CMY colour model |
| 39 | 6 | HSV colour model – HLS colour model; Colour selection |
| 40 | **7** | Animations &Realism: Design of Animation sequence |
| 41 | 7 | Animation function – raster animation |

| | | |
|---|---|---|
| 42 | 7 | Key frame systems – motion specification – morphing – tweening |
| 43 | 7 | Computer Graphics Realism: Tiling the plane |
| 44 | 7 | Recursively defined curves – Koch curves |
| 45 | 7 | C curves – Dragons – space filling curves |
| 46 | 7 | Fractals – Grammar based models |
| 47 | 7 | Fractals – turtle graphics – ray tracing |

**TEXT/REFERENCE BOOKS**


1. Computer Graphics /Hearn and Baker/PHI
2. Multimedia Systems Design / Prabhat Andleigh and Thakkar /PHI
3. Computer Graphics- Principles / J. Foley, A. Van Dam / Pearson


## UNIT- II

**Ivan Sutherland is a pioneer of Computer Graphics.**

**What is Computer Graphics?**

**Computer graphics** is the branch of computer science that deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer displays, and many specialized applications. A great deal of specialized hardware and software has been developed, with the displays of most devices being driven by computer graphics hardware. It is a vast and recently developed area of computer science.

Computer graphics is responsible for displaying art and image data effectively and meaningfully to the consumer. It is also used for processing image data received from the physical world, such as photo and video content. Computer graphics development has had a significant impact on many types of media and has revolutionized animation, movies, advertising, video games, and graphic design in general.

**Applications of Computer Graphics**

Some of the applications of computer graphics are as follows:

1. Computer Art and Computer Aided Design
2. Presentation Graphics
3. Entertainment

4. Education
5. Training

**Display Devices:**

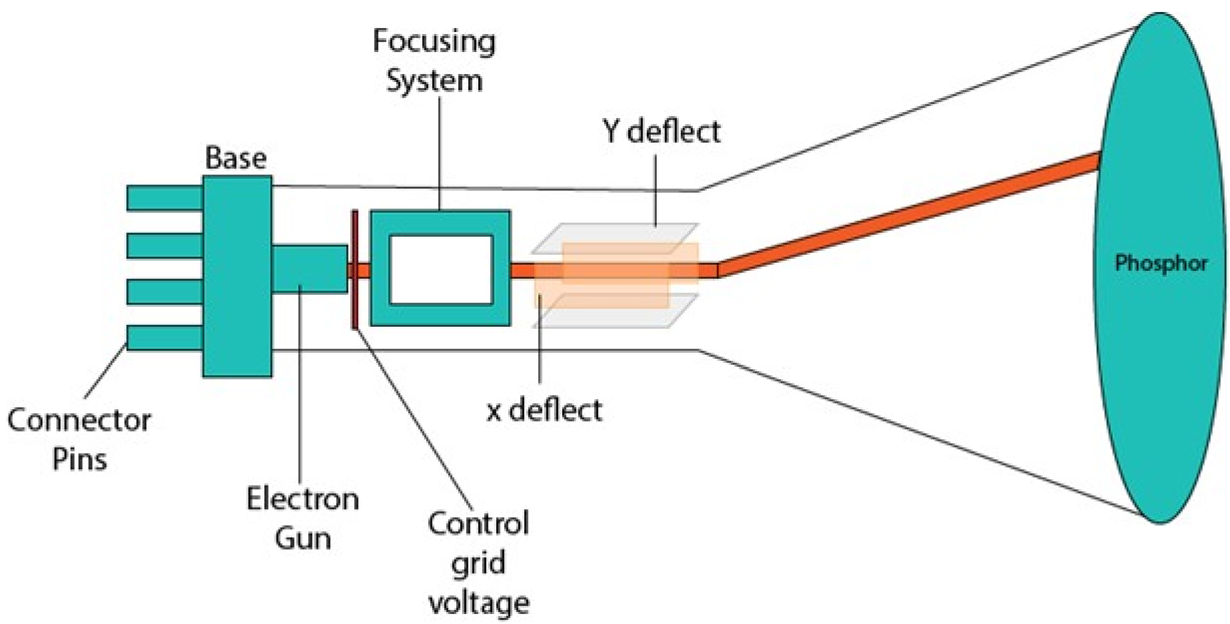
The most commonly used display device is a video monitor. The operation of most video monitors based on CRT (Cathode Ray Tube). Following are the most common display devices:

1. Refresh Cathode Ray Tube
2. Random Scan and Raster Scan
3. Color CRT Monitors
4. Direct View Storage Tubes
5. Flat Panel Display
6. Lookup Table

**Cathode Ray Tube (CRT):**

CRT stands for Cathode Ray Tube. CRT is a technology used in traditional computer monitors and televisions. The image on CRT display is created by firing electrons from the back of the tube of phosphorus located towards the front of the screen.

Once the electron heats the phosphorus, they light up, and they are projected on a screen. The color you view on the screen is produced by a blend of red, blue and green light.




**Cathode Ray Tube**

**Components of CRT:**

Main Components of CRT are:

**1. Electron Gun:** Electron gun consisting of a series of elements, primarily a heating filament (heater) 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 and 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 the electron beam. It creates an electric or magnetic field which will bend the electron beam as it passes through the area. In a conventional CRT, the yoke is linked to a sweep or scan generator. The deflection yoke which is connected to the sweep generator creates a fluctuating electric or magnetic potential.

**5. Phosphorus-coated screen:** The inside front surface of every CRT is coated with phosphors. Phosphors glow when a high-energy electron beam hits them. Phosphorescence is the term used to characterize the light given off by a phosphor after it has been exposed to an electron beam.

**Problem:** Light emitted by phosphors fades very rapidly so picture is not maintained for long.

**Solution:** repeatedly striking the beam to the phosphors on same points this makes picture visible for long time. This type of display is called as "refresh CRT".

**Color CRT:**

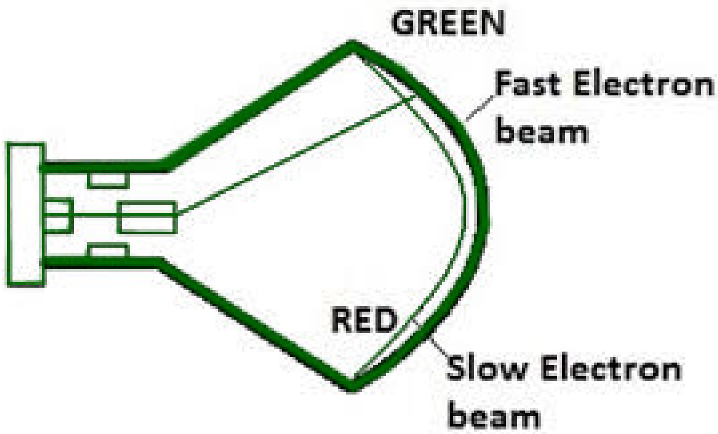
**1. Beam- penetration Method:**

The beam-penetration method for displaying color pictures has been used with ***random-scan*** monitors. Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen, and the displayed color depends on how far the electron beam penetrates into the phosphor layers.

A beam of slow electrons excites only the outer red layer.

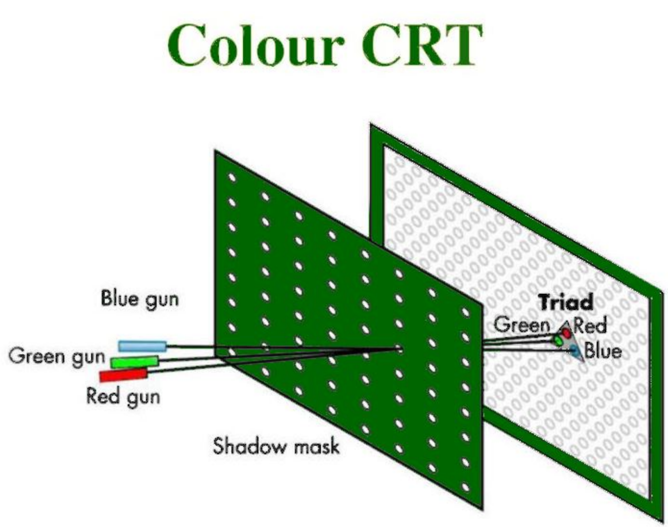
A beam of very fast electrons penetrates through the red layer and excites the inner green layer.

At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors', orange and yellow.




## 2. Shadow Mask Method:

Shadow-mask methods are commonly used in ***raster scan systems*** (including color TV) because they produce a much wider range of colors' than the beam penetration method. A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type of CRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen.




**Shadow mask techniques**

## Scanning Methods (Random & Raster):

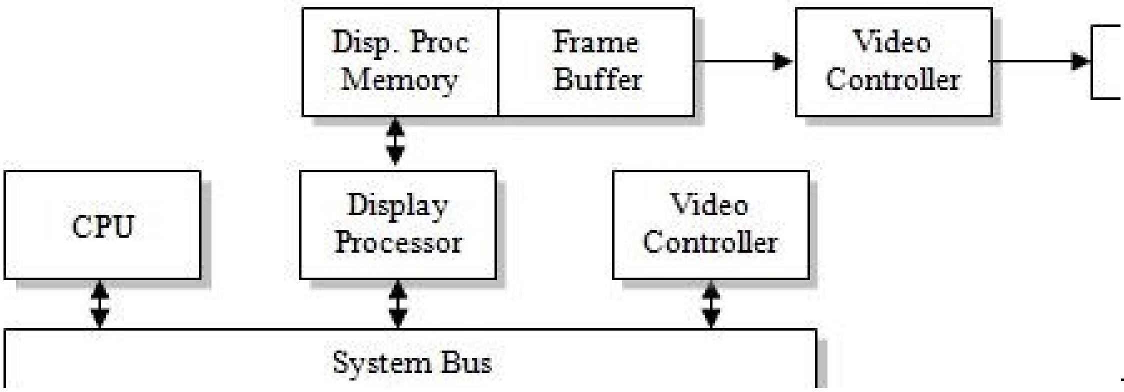
### Random Scan Method:

In black and white system we need only 1 bit per pixel where 1 represents white and 0 represents black.

- If the system has to display more colors' more bits are required. A high quality can have up to 24 bits per pixel

Frame Buffer --> Black & White --> Bitmap

Frame Buffer --> color system --> Pixmap




Refresh rate of raster scan is 60-80 frames per second.

- Horizontal retrace required 17% time of the required for one scan line.

- Vertical retrace retrace required 21% time of the required for one scan line.

- In non interlaced method refresh rate is 30 frames per second. (Flicker is noticeable)

- In interlaced method refresh rate is 60 frames per second.

# GRAPHICS SOFTWARE..

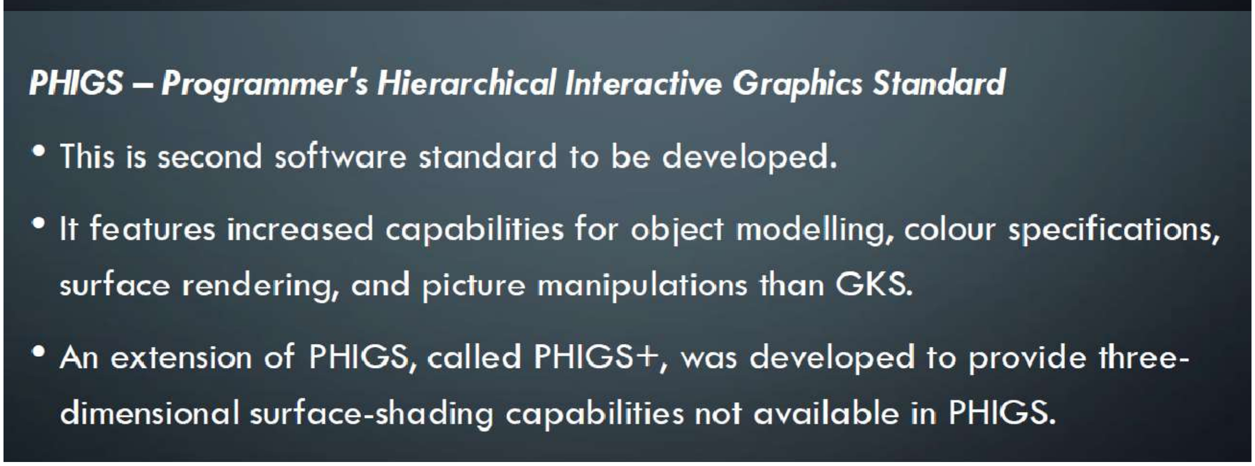
- A **general graphics programming package** provides an extensive set of graphics functions that can be used in a high-level programming language, such as C or FORTRAN.
  - An example of a general graphics programming package is the GL (Graphics Library) system on Silicon Graphics equipment.
  - Basic functions in a general package include those for generating picture components (straight lines, polygons, circles, and other figures), setting colour and intensity values, selecting views, and applying transformations.

# GRAPHICS SOFTWARE..

- **Application graphics packages**, on the other hand, are designed for nonprogrammers, so that users can generate displays without worrying about how graphics operations work.
  - The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms.
  - Examples of such applications packages are the artist's painting programs and various business, medical, and CAD systems.

# GRAPHICS STANDARDS

- The primary goal of standardized graphics software is portability.
- When packages are designed with standard graphics functions, software can be moved easily from one hardware system to another and used in different implementations and applications.
- Without standards, programs designed for one hardware system often cannot be transferred to another system without extensive rewriting of the programs.


**Assignment-1: Quiz**

1. THE INSIDE OF THE CATHODE RAY TUBE IS COATED WITH WHAT MATERIAL?
   A) Fluorescent powder
   B) No coating
   C) Phosphorus
   D) None of the above

ANSWER: C

2. Beam penetration method is usually used in ..............................
   A) LCD
   B) Raster Scan display
   C) Random scan display
   D) DVST

ANSWER: C

3. Shadow mask method is usually used in ..............................
   A) LCD

For Techno India NJR Institute of Technology
Dr. Pankaj Kumar Porwal
(Principal)

B) Raster Scan Display
C) Random Scan Display
D) DVST

ANSWER: B

4. Identify the colors produced in beam penetration method.
A) Red, Green, Blue, White
B) Red, Orange, Yellow, Green
C) Red, Green, Blue
D) Green, Red, White, Orange

ANSWER: B

5. In raster scan display, the frame buffer holds ......................
A) Line drawing commands
B) Scanning instructions
C) Image Resolution
D) Intensity information

ANSWER: D

6. In random scan display, the frame buffer holds ......................
A) Line drawing commands
B) Scanning instructions
C) Image Resolution
D) Intensity information

ANSWER: A

7. THE QUANTITY OF AN IMAGE DEPEND ON
A) No of Pixel used by image
B) No of lines used by image
C) No of resolution used by image
D) None

ANSWER: A

8. WHICH AMONG THE FOLLOWING IS NOT MERIT (ADVANTAGE) OF THE CATHODE RAY TUBE?
A) It runs at highest pixel ratio
B) It is less expensive than any other display technology
C) It is very large, heavy and bulgy
D) None of the above

ANSWER: C

9. WHICH AMONG THE FOLLOWING IS A PART OF THE CATHODE RAY TUBE?
A) Control Electrode
B) Electron Gun
C) Focusing System
D) All

ANSWER: D

10. Electron gun section ________
A) Provides sharp beam

B) Provides poorly focussed beam
C) Doesn’t provide any beam
D) Provides electrons only

ANSWER: A

11. Control grid is given ________
A) Positive voltage
B) Negative voltage
C) Neutral voltage
D) Zero voltage

ANSWER: B

12. What determines light intensity in a CRT?
A) Voltage
B) Current
C) Momentum of electrons
D) Fluorescent screen

ANSWER: C

13. Effect of negative voltage to the grid is ________
A) No force
B) A gravitational force
C) An attractive force
D) A repulsive force

ANSWER: D

14. How many guns are available for color monitor in shadow mask method?
A) 1
B) 2
C) 3
D) 4

ANSWER: C

15. How many colors can be generated using beam penetration method?
A) 3
B) 4
C) 254
D) 24

ANSWER: B

## UNIT- III

**Graphics Primitives:**

**Scan Conversion:**

It is a process of representing graphics objects a collection of pixels. The graphics objects are continuous. The pixels used are discrete. Each pixel can have either on or off state. 0 is represented by pixel off. 1 is represented using pixel on.

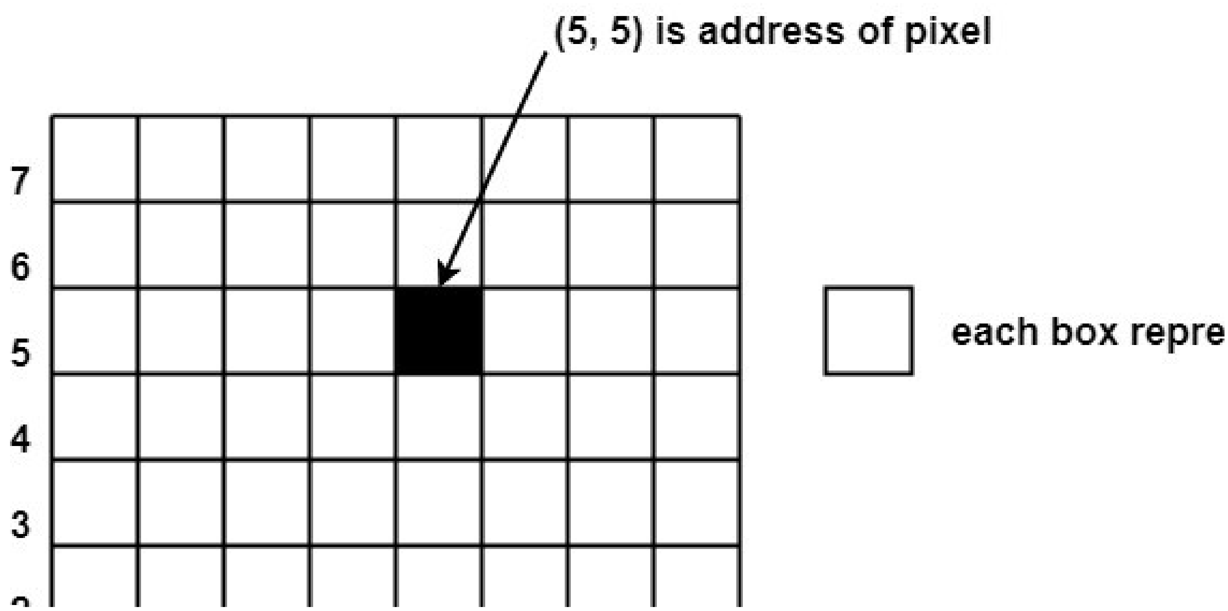
Using this ability graphics computer represent picture having discrete dots.

**Objects that can be scanned conversion:**

1. Point
2. Line
3. Sector
4. Arc
5. Ellipse
6. Rectangle
7. Polygon
8. Characters
9. Filled Regions

The process of converting is also called as rasterization. The algorithms implementation varies from one computer system to another computer system. Some algorithms are implemented using the software. Some are performed using hardware or firmware. Some are performed using various combinations of hardware, firmware, and software.

**Pixel or Pel:**
The term pixel is a short form of the picture element. It is also called a point or dot. It is the smallest picture unit accepted by display devices. A picture is constructed from hundreds of such pixels. Pixels are generated using commands. Lines, circle, arcs, characters; curves are drawn with closely spaced pixels. To display the digit or letter matrix of pixels is used.



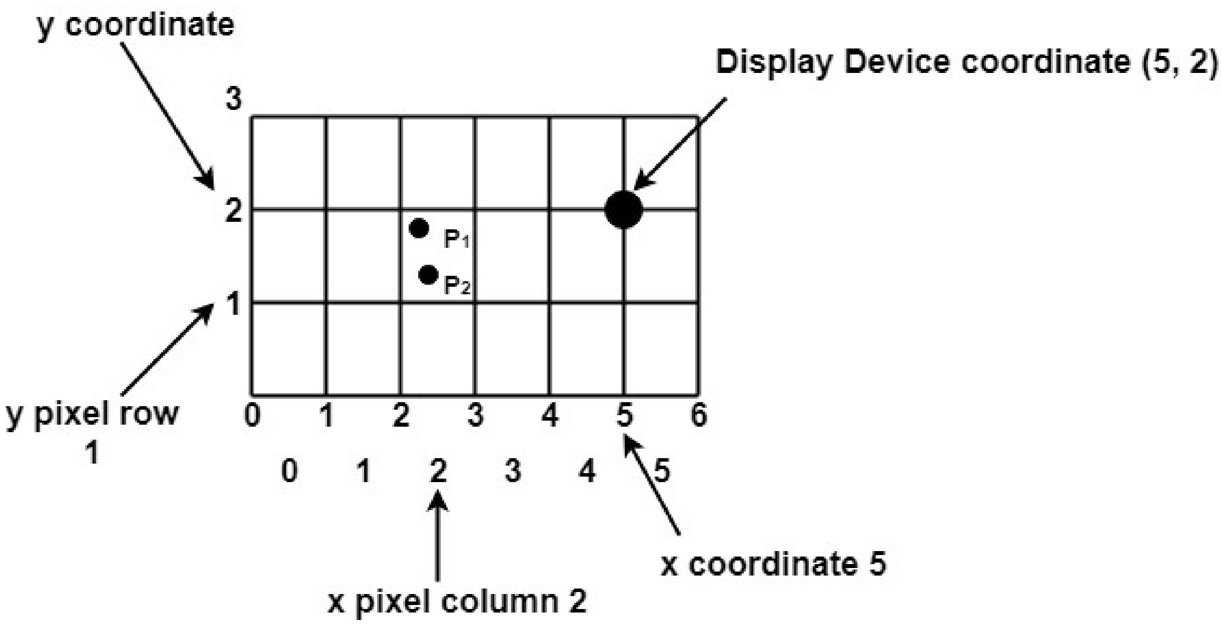

Different graphics objects can be generated by setting the different intensity of pixels and different colors of pixels. Each pixel has some co-ordinate value. The coordinate is represented using row and column.

P (5, 5) used to represent a pixel in the 5th row and the 5th column. Each pixel has some intensity value which is represented in memory of computer called a **frame buffer**. Frame Buffer is also called a refresh buffer.

**Scan Converting a Point:**

Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region which theoretically can contain an infinite number of points. Scan-Converting a point involves illuminating the pixel that contains the point.

**Example:** Display coordinates points as shown in fig would both be represented by pixel (2, 1). In general, a point p (x, y) is represented by the integer part of x & the integer part of y that is pixels [INT ( x ), INT ( y )].




**Line Drawing (scanning):**

A straight line may be defined by two endpoints & an equation. In fig the two endpoints are described by $(x_1,y_1)$ and $(x_2,y_2)$. The equation of the line is used to determine the x, y coordinates of all the points that lie between these two endpoints.

$_2,y_2)$

Using the equation of a straight line, $y = mx + b$ where $m = \frac{\Delta y}{\Delta x}$ & b = the y interrupt, we can find values of y by incrementing x from $x = x_1$, to $x = x_2$. By scan-converting these calculated x, y values, we represent the line as a sequence of pixels.

**Note: A good line drawing algorithm should produce a straight line always.**

**Algorithm for line drawing:**

1. DDA Line Drawing Algorithm
2. Bresenham Line Drawing Algorithm

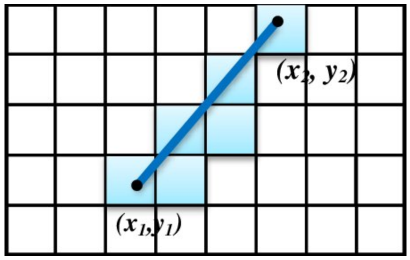
**Digital Differential Analyzer (DDA):**

**Algorithm:**

1. Compute the slope of the line 'm'
2. Increase the current value of x by 1, starting from leftmost coordinate
3. Calculate new value of y using: $y_i=mx_i+c$
4. Draw pixel at (xi, round(yi))

**Case 1:**

When m <=1




**Calculating value of m:**

$y=mx+c$ .......Eq. 1

For $(x_1, y_1)$ and $(x_2, y_2)$ Both will satisfy equation no 1 so

$y_1=mx_1+c$ .......Eq. 2

$y_2=mx_2+c$ .......Eq. 3

Eq. 3 – Eq.2

$(y_2 - y_1)= m(x_2 - x_1)$

$m= (y_2 - y_1)/ (x_2 - x_1)$ or we can say that $m=\Delta y/\Delta x$

**Calculating y' for next step of x+1:**

From Eq. 1 $y_i=mx_{i-1}+c$ …….Eq. 4

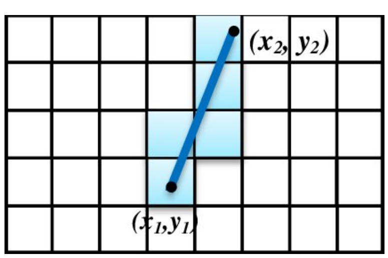
y' next point will obtain by $y'=m(x_{i-1}+1)+c$

$y'= mx_{i-1} + m + c$ by Eq. 4

$y'=y_i+m$ ……Eq. 5

**Case 2:**

When m >1




f the slope is greater than 1 then role of x and y will interchange. We sample y at unit intervals $\Delta y=1$ and compute successive x values.

$m=1/(x_2-x_1)$ or we can say $m=1/(x_{i+1}-x_i)$

$x_{i+1}=x_i+(1/m)$ …….Eq. 6

Case 1 and Case 2 are based on assumption that lines are to be processed from left end point to right end point.

**Case 3:**

If the processing is reversed i.e. if line starts from right to left then

$\Delta x = -1$

$m=(y_{i+1} - y_i)/ -1$

$y_{i+1} = y_i - m$ ……..Eq. 7

**Case 4:**

$\Delta y=-1$

$x_{i+1} = x_i - (1/m)$ ...…..Eq. 8

**Disadvantages:**

1. Lots of floating point calculations are involved
2. Multiplication and division involved in calculation takes more CPU cycles

**Algorithm:**

Step-1 Calculate constant dx, dy

Step-2 Calculate no of steps for iteration as:

nsteps = (dx>dy) ? dx : dy

Step-3 Calculate $x_{inc}$ and $y_{inc}$ as:

$x_{inc}$ = dx / nsteps & $y_{inc}$ = dy / nsteps

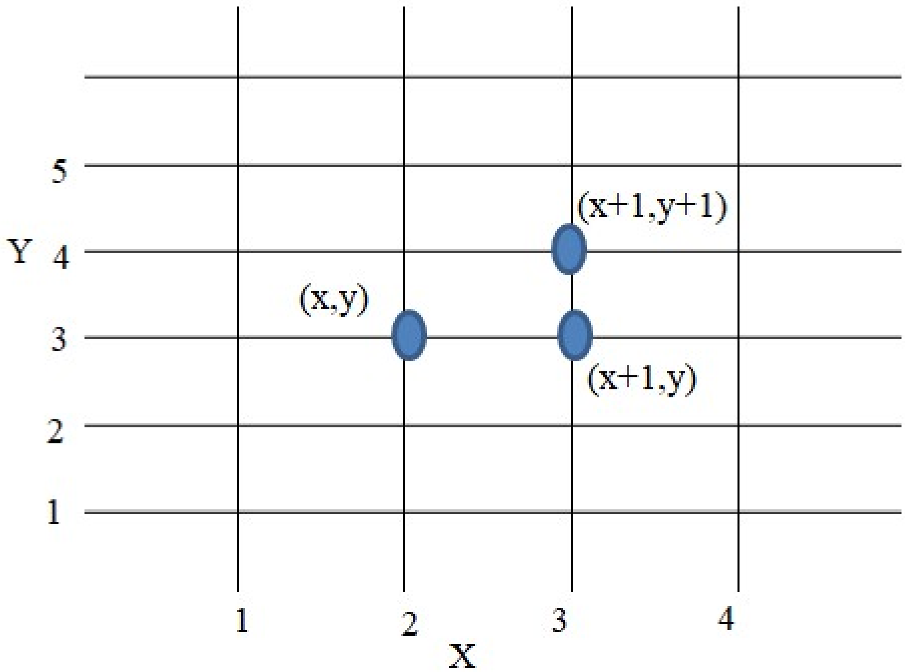
Step-4 Draw all the points by calculating consecutive x and y points as $x_{next} = x_{prev}+x_{inc}$, round-off ($y_{next}=y_{prev}+y_{inc}$)repeat this process n steps times.

**Bresenham's Line Drawing Algorithm:**

This is very efficient and faster algorithm. This algorithm follows the closeness theory to implement line plotting.

$0<m<1$ slope is between 0 to 1

Increment x by unit interval and plot the point whose y value is close to the ideal line path.




From figure, it can be seen that first point is plotted at (xp,yp). Next point will be plotted in the x direction would be xp+1 and for y we have to choose yp or yp+1. This can be decided mathematically by distance d1 and d2.

$Y= mx+c$ ……Eq. 1

$y= m(x_p+1) +c$ ……Eq. 2

$d2= (y_p+1)-y$ …….Eq. 3

by Eq. 3 and 2

$d2= y_p+1=m (x_p+1)-c$ ……Eq. 4

$d1= y-y_p$ …….Eq. 5

By Eq. 5 and 3

$d1=m\ (x_p+1) + c - y_p$ ……Eq. 6

So d1-d2 will be

$\Rightarrow m\ (x_p+1) + c - y_p - y_p - 1 + m\ (x_p+1) + c$

$\Rightarrow 2m\ (x_p+1) - 2y_p + 2c - 1$ …….Eq. 7

To make sure that during calculation of decision parameter we will include only integer calculation we know $m=\Delta y/\Delta x$ so putting this in Eq. 7

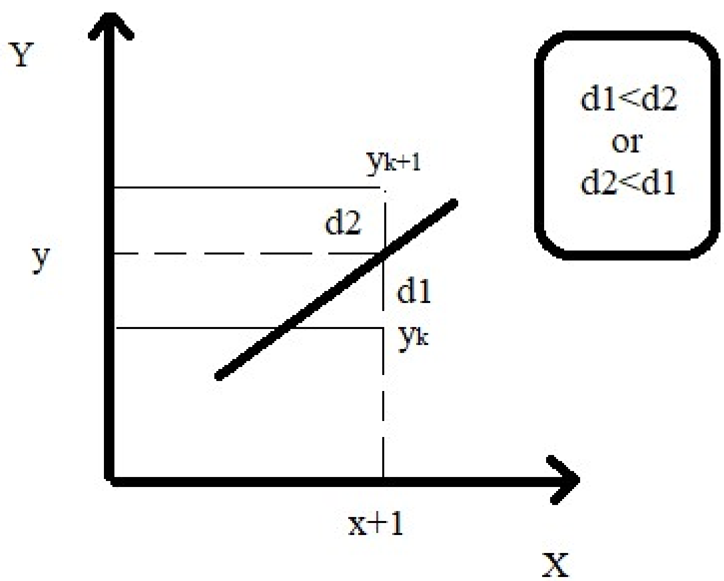
$d= (d1- d2)= 2\Delta y/\Delta x\ (x_p+1) - 2y_p+2c - 1$ Multiplying both the sides by $\Delta x$ will get

$d= \Delta x\ (d1-d2)= 2\Delta y\ x_p+ 2\Delta y - 2\Delta x\ y_p+ \Delta x\ (2c-1)$

$d= 2\Delta y\ x_p- 2\Delta x\ y_p+ c'$ where $c'= 2\Delta y +\Delta x\ (2c-1)$ ….Eq. 8

After this calculation if d is –ve, means d1<d2 so will choose $y_p$ so next point will be $(x_p+1, y_p)$ else will choose $(x_p+1, y_p+1)$

Set $d_{old}=d$




**Case 1:** Select $y_p$

$d_{new} = 2\Delta y\ (x_p +1) - 2\Delta x\ y_p+ c'$ …….Eq. 9

$\Delta d = d_{new} - d_{old}$ Eq. 8 and 9

$\Delta d = 2\ \Delta y$

**Case 2:** select $y_p+1$

$d_{new} = 2\Delta y\ (x_p+1) - 2\Delta x\ (y_p+1) + c'$ ……Eq. 10

$\Delta d = d_{new} - d_{old}$ Eq. 8 and 10

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

**Algorithm:**

1. Input start point and end point of the line consider initial point as (x0,y0) and plot them

2. Calculate the constants $\Delta x$, $\Delta y$, $2\ \Delta y$, $2\ \Delta y - 2\Delta x$ and obtain the starting value for the decision parameter as $d_{start} = 2\ \Delta y - \Delta x$

3. At each step test the value of decision variable

   a. If $d<0$ choose $y_p$ and increment d by $2\ \Delta y$

   · $d_{new} = d_{old} + 2\ \Delta y$

   b. Else choose $y_{p+1}$ and increment d by $2\ \Delta y - 2\Delta x$

   · $d_{new} = d_{old} + 2\ \Delta y - 2\Delta x$

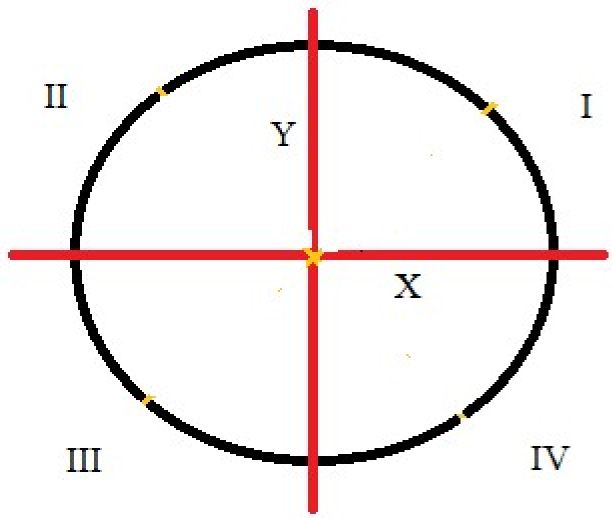
5. Repeat step 3 until last point is reached

Mid Point Circle

A circle is a set of all points that lie at an equal distance from a fixed point called as centre.

**Symmetric figure:**

4- Way symmetric (Quadrant)




Circle Drawing

8- Way symmetric (Octant)

Circle Equation:

$x^2+y^2=r^2$ { centre (0,0)} ……..Eq. 1

Brute force algorithm

$y2=r^2-x^2$

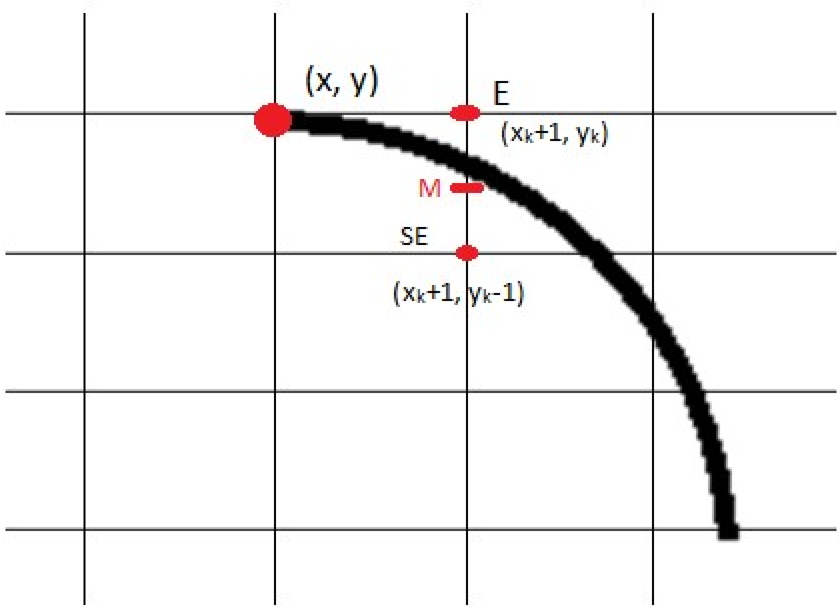
$y=\sqrt{r^2-x^2}$ (finding square root in every calculation is expensive)

**Mid Point Circle:**

$x^2+y^2-r^2=0$

Putting midpoints say (x', y') into the equation will get 3 possible results.

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




Case 1: if result is 0 point lies on the line

Case 2: if result is <0 then point lies inside circle boundary

Case 3: if result is >0 then point les outside the circle boundary

Consider Mid Point coordinates as $(x_m, y_m)$ -> $(x_k+1, y_k-1/2)$

Putting these coordinates in Eq. 1

$P_k = (x_k+1)^2 + (y_k-1/2)^2 – r^2$ …..Eq. 2

$P_{k+1} = (x_{k+1}+1)^2 + (y_{k+1}-1/2)^2 – r^2$ …..Eq. 3

$P_{k+1} - p_k = (x_{k+1}+1)^2 - (x_k+1)^2 + (y_k-1/2)^2 – (y_{k+1}-1/2)^2 + r^2 – r^2$

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

$P_{k+1} - p_k = x_k^2 + 4 + 4x_k – x_k^2 – 1 - 2x_k + y_{k+1}^2 + ¼ - y_{k+1} – y_k^2 – ¼ + y_k$

$P_{k+1} = p_k + 2x_k + 3 + y_{k+1}^2 – y_k^2 - y_{k+1} + y_k$

**if $p_k < 0$** $y_{k+1} = y_k$

$p_{k+1} = p_k + 2x_k + 3 + y_k^2 – y_k^2 - y_k + y_k$

**$p_{k+1} = p_k + 2x_k + 3$**

**if $p_k > 0$** $y_{k+1} = y_k - 1$

**$p_{k+1} = p_k + 2x_k + 5 - 2y_k$**

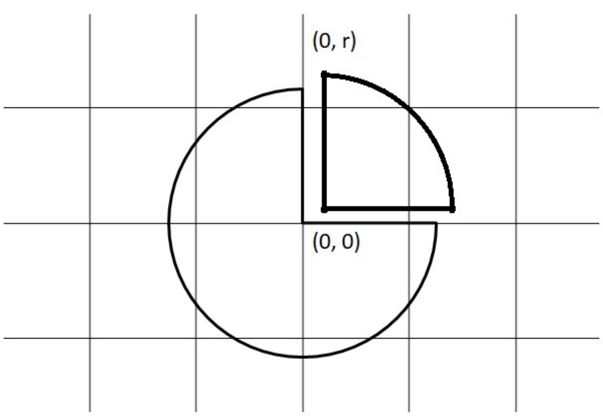
P0 initial decision parameter

$x_k=0$ $y_k=r$

Putting this in Eq. 2

P0 = 5/4 –r

**Algorithm:**



If r is the radius of the circle to be drawn and origin is its centre, then to plot the first octant of the circle, do following:

1. Plot the initial point $(x_i, y_i)$ such that: $x_i = 0$ and $y_i = r$

2. Find initial decision parameter $p_i = 5/4 - r$

3. If $p_i < 0$ then

   $x_{i+1} = x_i + 1$

   $y_{i+1} = y_i$

   $p_{i+1} = p_i + 2x_i + 3$

6. If $p_i > 0$ then

   $x_{i+1} = x_i + 1$

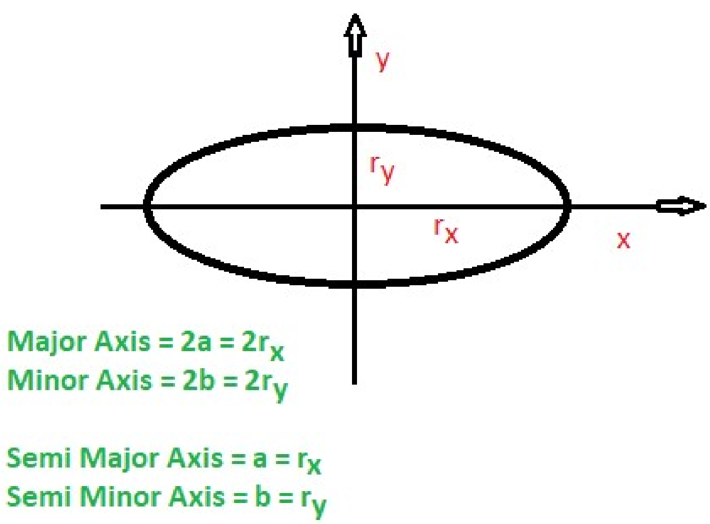
   $y_{i+1} = y_i - 1$

   $p_{i+1} = p_i + 2(x_i - y_i) + 5$

7. Repeat step 3, 4 until x becomes greater than or equal to y

To plot the complete circle, reflect each point of the first octant, onto 7 other octants making use of 8 – way symmetry

**Mid Point Ellipse Drawing:**



$x^2/a^2 + y^2/b^2 = 1$ …….Eq. 1

Simplifying this equation will get

$b^2x^2 + a^2y^2 = a^2b^2$

$r_x^2 y^2 + x^2 r_y^2 - r_x^2 r_y^2 = 0$ …….Eq. 2

Putting any point in Eq. 2 will get 3 cases:

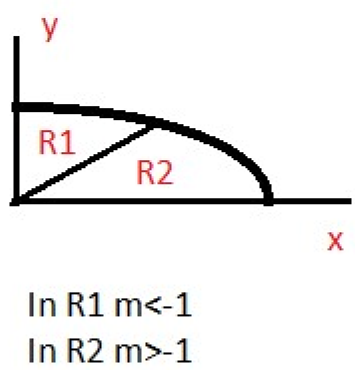
Case 1: if result is 0 point lies on the ellipse boundary

Case 2: if result is <0 then point lies inside ellipse boundary

Case 3: if result is >0 then point les outside the ellipse boundary

**Note:**

1. Circle has 8- way symmetry where as ellipse has 4- way symmetry
2. In circle we need to plot one octant but in case of ellipse we need to plot 2 octant i.e. 1 quadrant




**Quadrant – 1(Region 1)**

- Start point: $(0, r_y)$

- Slope of Curve < -1

- Take unit steps in positive x direction till boundary between 2 regions is reached

**Quadrant- 1 (Region 2)**

- Slope of curve > -1

- Take unit step in negative y direction till the end of the quadrant.

**Note:** On the boundary between 2 region or octant the slope of the curve is -1.

**Slope of curve:**

$dy/dx\ (r_y^2x^2 + r_x^2\ y^2 – r_x^2r_y^2) = 0$

$dy/dx\ (y^2) = dy/dx\ \{(r_x^2r_y^2 - r_y^2x^2)/ r_x^2\}$

$dy/dx\ 2y = dy/dx\ (r_x^2r_y^2 / r_x^2) –dy/dx( r_y^2x^2 / r_x^2)$

$dy/dx = - 2r_y^2x / 2\ r_x^2y$

$- 2r_y^2x / 2\ r_x^2y =-1$

$- 2r_y^2x = -2\ r_x^2y$

**Region -1:**

$(x_k + 1, y_k)$ or $(x_k + 1, y_k - 1)$ so the midpoint will be $(x_k + 1, y_k - 1/2)$

$r_y\ ^2x^2 + r_x^2\ y^2 – r_x\ ^2r_y^2 = 0$

$r_y^2\ (x_k + 1)^2 + r_x^2(y_k-1/2)^2 – r_x^2r_y^2 = p_k$ ……Eq. 3

$r_y^2\ (x_{k+1} + 1)^2 + r_x^2(y_{k+1}-1/2)^2 – r_x^2r_y^2 = p_{k+1}$

$r_y^2\ ((x_k+1) + 1)^2 + r_x^2(y_{k+1}-1/2)^2 – r_x^2r_y^2 = p_{k+1}$ …….Eq. 4

$p_{k+1} - p_k$ -> (Eq. 4 –Eq. 3)

$r_y^2\ ((x_k+1) + 1)^2 + r_x^2(y_{k+1}-1/2)^2 – r_x^2r_y^2 - r_y^2\ (x_k + 1)^2 - r_x^2(y_k-1/2)^2 + r_x^2r_y^2$

$r_y^2\ ((x_k+1) + 1)^2 - r_y^2\ (x_k + 1)^2 + r_x^2(y_{k+1}-1/2)^2 - r_x^2(y_k-1/2)^2$

$r_y^2\ (2(x_k+1) + 1) + r_x^2(y_{k+1}^2 - y_k^2 – y_{k+1} + y_k) = p_{k+1} - p_k$ …...Eq. 5

$\mathbf{p_k < 0}$: $y_{k+1} = y_k$

$p_{k+1}=p_k + r_y^22x_{k+1} + r_y^2$

**$p_k$ >= 0**: $y_{k+1} = y_k - 1$

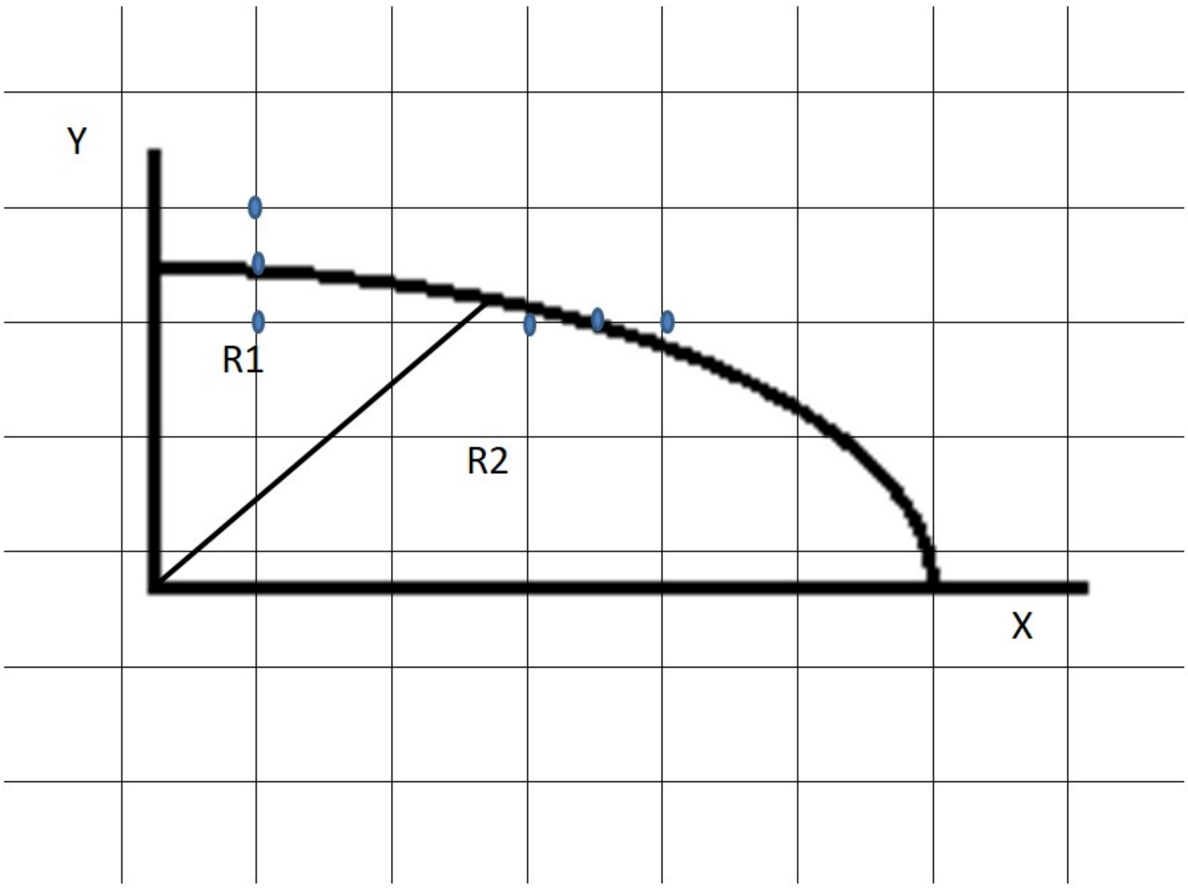
$p_{k+1}=p_k +2x_{k+1}r_y^2 + r_y^2 – 2y_{k+1}r_x^2$

Initial decision parameter

Put $(0, r_y)$in Eq. 3

$P_0 = r_y^2 + r_x^2/4 – r_yr_x^2$

**Region- 2:**




$(x_k , y_k - 1)$ or $(x_k + 1, y_k - 1)$ so the midpoint will be $(x_k + 1/2, y_k - 1)$

$r_y{}^2x^2 + r_x^2 y^2 – r_x{}^2r_y^2 = 0$

$r_y^2 (x_k + 1/2)^2 + r_x^2(y_k - 1)^2 – r_x^2r_y^2 = p2_k$ ……Eq. 6

$r_y^2 (x_{k+1} + 1/2)^2 + r_x^2(y_{k+1}-1)^2 – r_x^2r_y^2 = p2_{k+1}$

$r_y^2 (x_{k+1} + 1/2)^2 + r_x^2((y_k – 1) - 1)^2 – r_x^2 r_y^2 = p2_{k+1}$ …….Eq. 7

Eq. 7 – Eq. 6

$r_y^2 (x_{k+1} + 1/2)^2 - r_y^2 (x_k + 1/2)^2 + r_x^2((y_k – 1) - 1)^2 – r_x^2(y_k - 1)^2 + r_x^2 r_y^2 - r_x^2 r_y^2 = p2_{k+1} - p2_k$

$r_y^2 (x_{k+1}^2 + ¼ + x_{k+1}) - r_y^2 (x_k^2 + ¼ + x_k) + r_x^2[(y_k – 1)^2 + 1 – 2(y_k – 1)]^2 – r_x^2(y_k^2 + 1 +2 y_k)$

$r_y^2 \{x_{k+1}^2 + ¼ + x_{k+1} - x_k^2 - ¼ - x_k\} + r_x^2 \{y_k^2 + 1 – 2y_k + 1 – 2y_k + 2 - y_k^2 – 1 + 2 y_k\}$

$r_y^2 \{x_{k+1}^2 + x_{k+1} - x_k^2 - x_k\} + r_x^2 \{– 2y_{k+1} + 1\}$

$p2_{k+1} - p2_k = r_y^2 \{x_{k+1}^2 + x_{k+1} - x_k^2 - x_k\} + r_x^2 \{1 – 2y_{k+1}\}$

$\mathbf{p_k > 0}$: $x_{k+1} = x_k$

$p2_{k+1} = p2_k – 2y_{k+1} x_k^2 + r_x^2$ ……Eq. 8

$\mathbf{p_k <= 0}$: $x_{k+1} = x_k + 1$

$p2_{k+1} = p2_k + r_y^2 (2x_{k+1}) – 2y_{k+1} r_x^2 + r_x^2$ ……Eq. 9

**Initial Decision Parameter:** will be derived by putting last point of region 1 in the equation

$p2_0 = r_y^2 (x_k + 1/2)^2 + r_x^2(y_k - 1)^2 – r_x^2 r_y^2$ (x==y)

$p2_0 = r_y^2 (x + 1/2)^2 + r_x^2(y - 1)^2 – r_x^2 r_y^2$ …..Eq. 10

**Algorithm:**

1. Read radii $r_x$ and $r_y$
2. Initialize starting point of region 1 as x = 0 and y = $r_y$
3. Calculate
   $P1_0 = r_y^2 + r_x^2/4 – r_y r_x^2$
4. Calculate
   $dx = 2r_y^2 x$ and $dy = 2 r_x^2 y$
5. Repeat while (dx < dy)

```
        Plot(x, y)
        If(p1 < 0){
                x =x+1
                Update dx (2ry^2 x = old dx +2 ry^2)
                P1 = p1 +2ry^2x +ry^2
        }
        else {
```

$x = x + 1$, $y = y - 1$
Update dx ($2r_y^2 x$ = old dx + 2 $r_y^2$)
Update dy ($2r_x^2 y$ = old dy - 2 $r_x^2$)
$p1 = p1 + dx - dy + r_y^2$
}

6. When (dx >= dy) plot region 2 as:
7. Find

$$p2_0 = r_y^2 (x + 1/2)^2 + r_x^2(y - 1)^2 - r_x^2 r_y^2$$

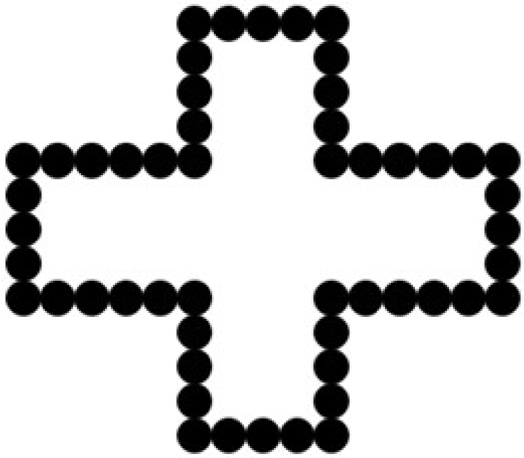
8. Repeat till (y > 0)

Plot (x, y)
If(p2 > 0){
$x = x$
$y = y-1$
Update dy : $2r_x^2 y$
$p2 = p2 - dy + r_x^2$
}
else{
$x = x + 1$
$y = y - 1$
$dy = dy - 2r_x^2$
$dx = dx + 2r_y^2$
$p2 = p2 + dx - dy + r_x^2$
}

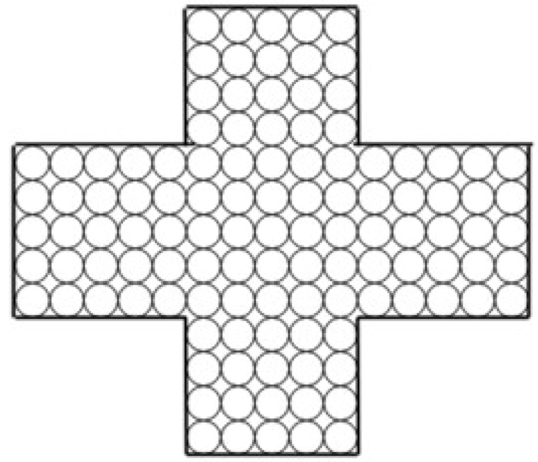
9. END

**Filled Area Primitives:**

Region filling is the process of filling image or region. Filling can be of boundary or interior region as shown in fig. Boundary Fill algorithms are used to fill the boundary and flood-fill algorithm is used to fill the interior.



Boundary Filled Region



Interior or Flood Filled Region

*Boundary Filled Algorithm:*

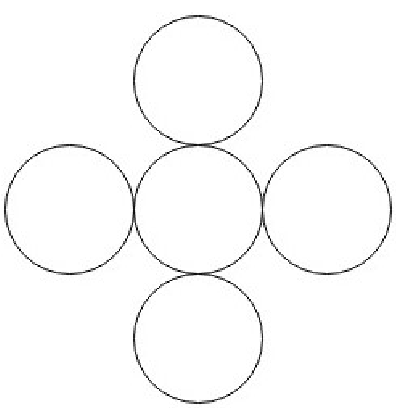
This algorithm uses the recursive method. First of all, a starting pixel called as the seed is considered. The algorithm checks boundary pixel or adjacent pixels are coloured or not. If the adjacent pixel is already filled or coloured then leave it, otherwise fill it. The filling is done using four connected or eight connected approaches.

Four connected approaches is more suitable than the eight connected approaches.

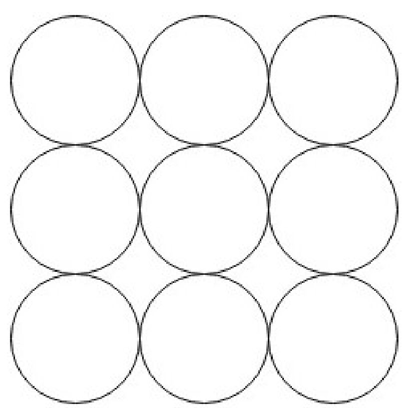
**1. Four connected approaches:** In this approach, left, right, above, below pixels are tested.

**2. Eight connected approaches:** In this approach, left, right, above, below and four diagonals are selected.

Boundary can be checked by seeing pixels from left and right first. Then pixels are checked by seeing pixels from top to bottom. The algorithm takes time and memory because some recursive calls are needed.



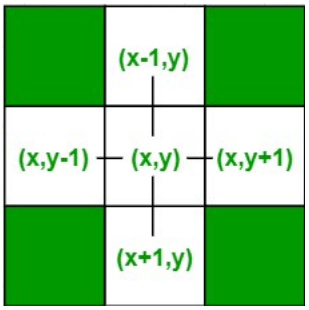
Four Connected



Eight Connected

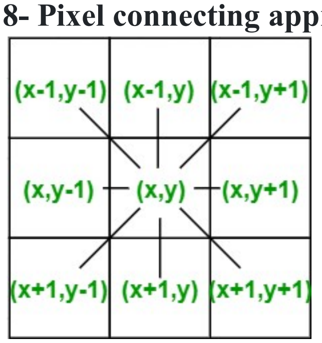
**Algorithm:**

1. Initialize the 4 values namely x, y, fill_color, and default_color.
2. Define the value of boundary pixel color or boundary color.
3. Check if the current pixel is of default color and if yes then repeat step 4 and 5 till the boundary pixels are reached
4. Change the default color with the fill color at the current pixel.
5. Repeat step 3 and 4 for the neighboring 4 pixel.
6. Exit




```
void boundaryFill4(int x, int y, int fill_color,int boundary_color)
{
   if(getpixel(x, y) != boundary_color && getpixel(x, y) != fill_color)
   {
      putpixel(x, y, fill_color);
      boundaryFill4(x + 1, y, fill_color, boundary_color);
      boundaryFill4(x, y + 1, fill_color, boundary_color);
      boundaryFill4(x - 1, y, fill_color, boundary_color);
      boundaryFill4(x, y - 1, fill_color, boundary_color);
   }
}
```

**8- Pixel connecting approach:**




```
void boundaryFill8(int x, int y, int fill_color,int boundary_color)
{
   if(getpixel(x, y) != boundary_color && getpixel(x, y) != fill_color)
   {
      putpixel(x, y, fill_color);
```

```
        boundaryFill8(x + 1, y, fill_color, boundary_color);
        boundaryFill8(x, y + 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y, fill_color, boundary_color);
        boundaryFill8(x, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y + 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y + 1, fill_color, boundary_color);
    }
}
```

***Flood Filled Algorithm:***

In this method, a point or seed which is inside region is selected. This point is called a seed point. Then four connected approaches or eight connected approaches is used to fill with specified color.

The flood fill algorithm has many characters similar to boundary fill. But this method is more suitable for filling multiple colors boundary. When boundary is of many colors and interior is to be filled with one color we use this algorithm.

**Algorithm:**

Flood-Fill (node, target-color, replacement-color)

1. If target-color is equal to replacement-color, return.
2. If the color of node is not equal to target-color, return.
3. Set the color of the node to replacement-color.

    Perform Flood-Fill (one step to the south of node, target-color, replacement-color)
    Perform Flood-Fill (one step to the north of node, target-color, replacement-color).
    Perform Flood-Fill (one step to the west of node, target-color, replacement-color).
    Perform Flood-Fill (one step to the east of node, target-color, replacement-color)

5. Return.

***4- Way connected:***

```
if(getpixel(x,y)==defaultColor)
   {
      putpixel(x,y,fillColor);
      flood(x+1,y,fillColor,defaultColor);
      flood(x-1,y,fillColor,defaultColor);
      flood(x,y+1,fillColor,defaultColor);
      flood(x,y-1,fillColor,defaultColor);
   }
```

***8- Way connected:***

```
if(current==old)
            {
                        putpixel(x,y,newcol);
                        floodfill(x+1,y,old,newcol);
                        floodfill(x-1,y,old,newcol);
                        floodfill(x,y+1,old,newcol);
                        floodfill(x,y-1,old,newcol);
                        floodfill(x+1,y+1,old,newcol);
                        floodfill(x-1,y+1,old,newcol);
                        floodfill(x+1,y-1,old,newcol);
                        floodfill(x-1,y-1,old,newcol);
            }
```

**Disadvantage:**

1. Very slow algorithm
2. May be fail for large polygons
3. Initial pixel required more knowledge about surrounding pixels.

**Assignment-3**

1. Derive all the equations of circle drawing using Bresenham's circle drawing algorithm with example.

## UNIT- IV

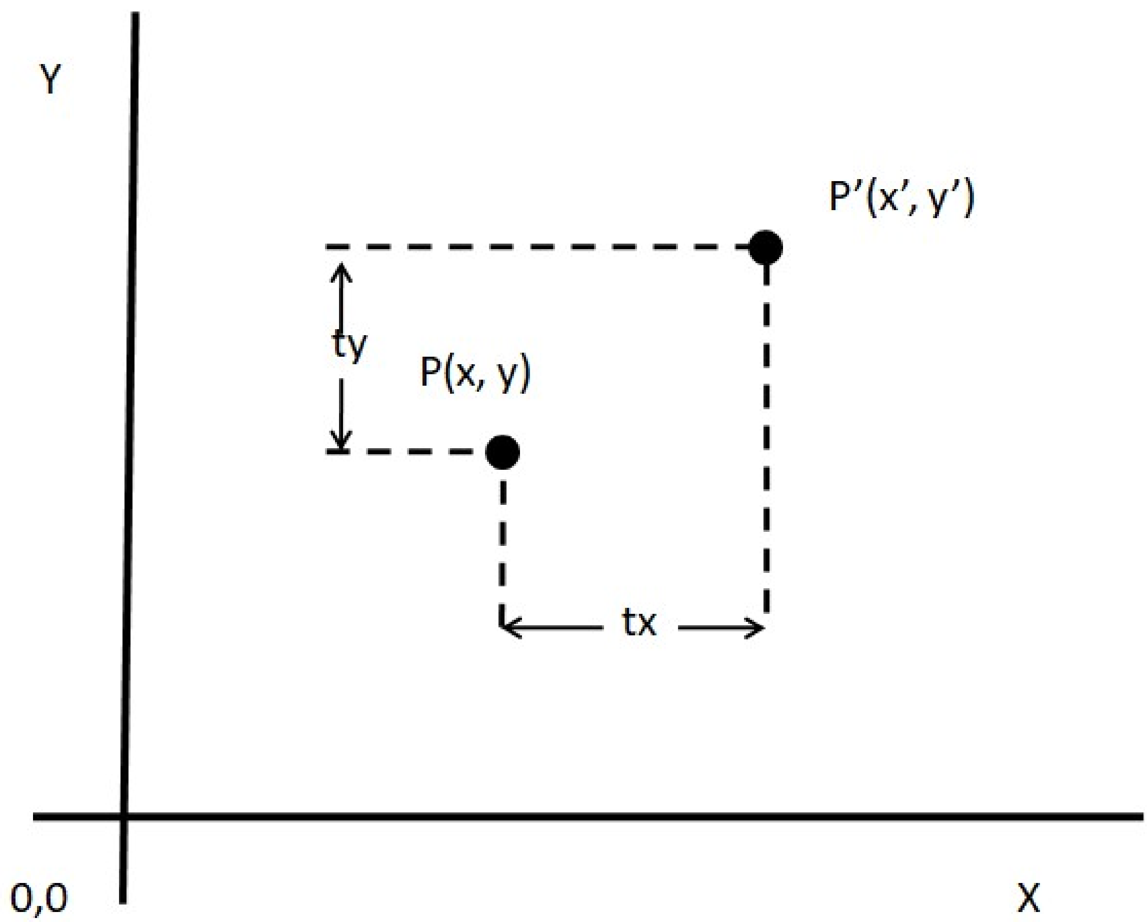
**Translation**

**Transformation:**

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

**Translation:**




A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate ($t_x$, $t_y$) to the original coordinate X, Y to get the new coordinate X′, Y′.

From the above figure, you can write that −

$$X' = X + t_x$$

$$Y' = Y + t_y$$

The pair ($t_x$, $t_y$) is called the translation vector or shift vector. The above equations can also be represented using the column vectors.

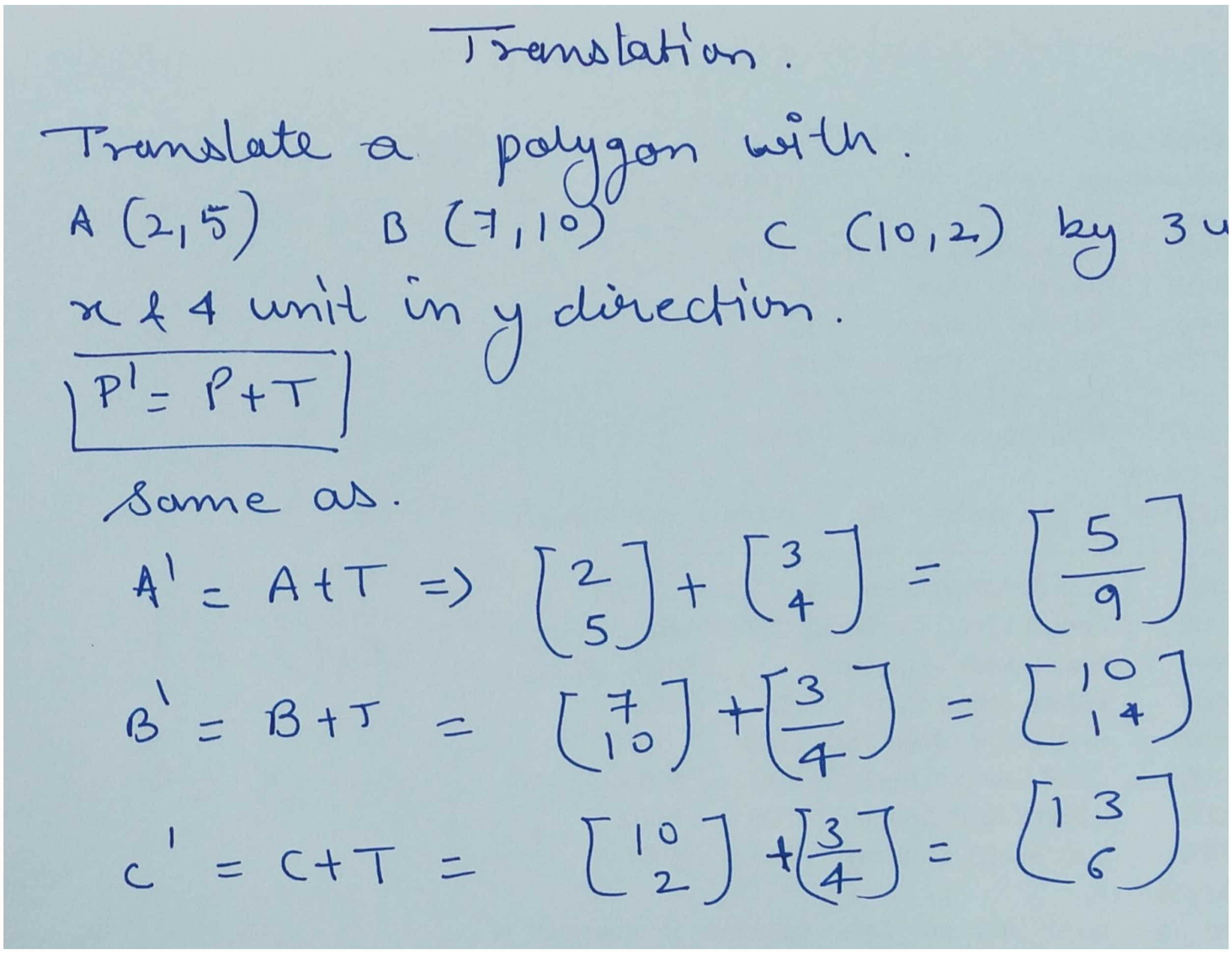
P=[X] / [Y]

P' = [X′] / [Y′]

T = [tx] / [ty]

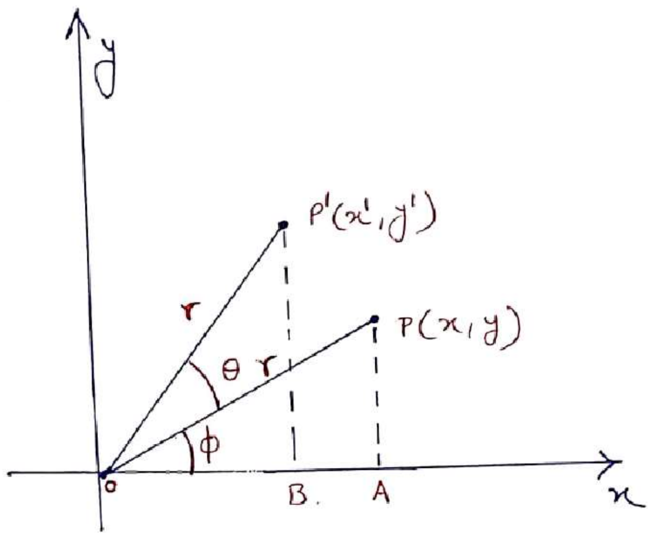
We can write it as −

**P’ = P + T**



# Rotation






$OP' = OP = r$ constant distance from origin to object.

$\phi$ = Original Angle

$\theta$ = Rotation Angle

To calculate $x$, & $y$.

$\cos\phi = \frac{\text{Base}}{\text{Hyp.}} = \frac{x}{r} \Rightarrow x = r\cos\phi$ . ---- (1)

$\sin\phi = \frac{\text{Prep.}}{\text{Hyp.}} = \frac{y}{r} \Rightarrow y = r\sin\phi$ ---- (2)

To calculate $x'$, $y'$ (Angle will be $(\phi+\theta)$)

$\cos(\phi+\theta) = \frac{x'}{r} \Rightarrow x' = r\cos(\phi+\theta)$

$= r[\cos\phi\cos\theta - \sin\phi\sin\theta]$

$x' = r\cos\phi\cos\theta - r\sin\phi\sin\theta$ . ---- (3)

$\sin(\phi+\theta) = \frac{y'}{r} \Rightarrow y' = r(\sin(\phi+\theta))$

$y' = r\cos\phi\sin\theta + r\cos\theta\sin\phi$ ——— (4)

using eq 3, 2 & 1. we can write.

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

using eq. 4, 2, & 1 we can write

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

Represent following in Matrix format

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

$P' = P \cdot R$

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

= If Rotation is Anticlockwise then Angle is +ve
- else Angle will be -ve.

$$R = \begin{bmatrix}\cos(-\theta) & \sin(-\theta) \\ -\sin(-\theta) & \cos(-\theta)\end{bmatrix} \Rightarrow \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$$

Q

(3)

P (4, 3) ∠ = 45° calculate P' & R.

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

$$P' = [4, 3] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad \cos 45^\circ = \sin 45^\circ = 1/\sqrt{2}$$

$$P' = (4, 3) \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$P' = \begin{bmatrix} \frac{4}{\sqrt{2}} - \frac{3}{\sqrt{2}} & \frac{4}{\sqrt{2}} + \frac{3}{\sqrt{2}} \end{bmatrix}$$

$$P' = \begin{bmatrix} 1/\sqrt{2} & \cdot\, 7/\sqrt{2} \end{bmatrix}$$

# Scaling

Scaling factor in
x direction $S_x$
y direction $S_y$.

If intial point is at say $A(x, y)$ we have to find out new points $A'(x', y')$.

$x' = x \cdot S_x$

$y' = y \cdot S_y$  Representing it into Matrix

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

By the value of $S_x$ & $S_y$ we can have 4 cases:

1: $S_x$ & $S_y > 1$ object enlarge
2: $S_x$ & $S_y < 1$ object decrease
3: $S_x$ & $S_y == 1$ No change.
4: $S_x(+/-)$ & $S_y(-/+)$ differential Image.

Q. Givine a polygon.

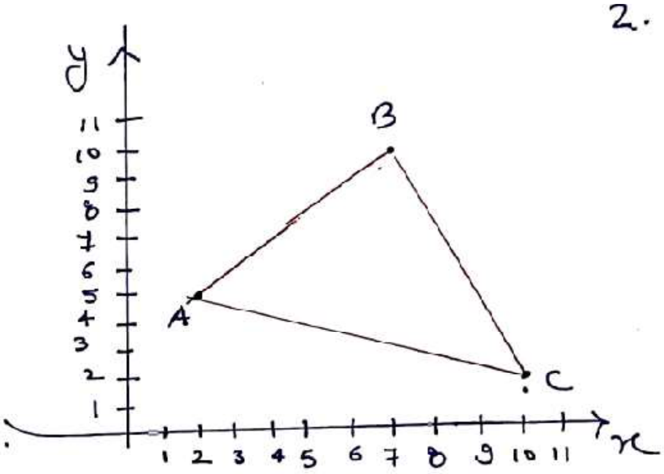
A(2,5) B(7,10)

C(10,2) is

Scaled by.

2 units in x direction

4 2 units in y direction.




Calculate New cordinates ~~by~~

A', B', C'

$$P' = P \cdot S. \qquad S = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\begin{bmatrix} x_1' & y_1' \\ x_2' & y_2' \\ x_3' & y_3' \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

**Transformation Numerical:**

**Translation**:

1. Given a circle C with radius 10 and centre coordinates (1, 4). Apply the translation with distance 5 towards X axis and 1 towards Y axis. Obtain the new coordinates of C without changing its radius.

2. Given a square with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the translation with distance 1 towards X axis and 1 towards Y axis. Obtain the new coordinates of the square.

**Rotation:**

1. Given a line segment with starting point as (0, 0) and ending point as (4, 4). Apply 30 degree rotation anticlockwise direction on the line segment and find out the new coordinates of the line.

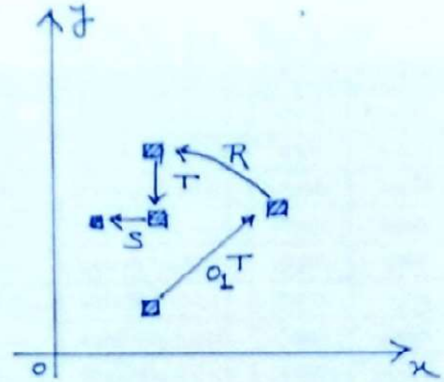
2. Given a triangle with corner coordinates (0, 0), (1, 0) and (1, 1). Rotate the triangle by 90 degree anticlockwise direction and find out the new coordinates.



**Scaling:**

1. Given a square object with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the scaling parameter 2 towards X axis and 3 towards Y axis and obtain the new coordinates of the object.

## HOMOGENEOUS COORDINATES (1)




- To perform Multiple operation on any object we have to perform them separately. to avoid this we can use the concept of homogeneous coordinates.
- In Homogeneous coordinate system, two dimensional coordinates postions $(x, y)$ are represented by triple-coordinates.
- For 2-dimensional geomatric transformation we can choose homogeneous parameter 'h' to any non-zero value. For our convenince take it as one.

$$(x, y, 1)$$

1) Translation.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- which we can write in abbreveated form.

$$P' = T(t_x, t_y) \cdot P.$$

- Inverse of Translation Matrix is obtained by replacing $t_x$ & $t_y$ with their negatives $-t_x$ $-t_y$

2.) Rotation.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

- we get inverse rotation matrix when $\theta$ is replaced by $-\theta$.

3.) Scaling

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

$$P' = S(s_x, s_y) \cdot P.$$

- we can get inverse scaling matrix when $(s_x, s_y)$ is replaced with invers multiplicative $\left(\frac{1}{s_x}, \frac{1}{s_y}\right)$

# COMPOSITE TRANSFORMATION (3)

By using homogeneous representation of matrix we can set up a matrix of any sequence of transformations as a composit transformation matrix by calculating the matrix product of individual transformation.

1) Translation.

- If 2 successive translation vector $(t_{x_1}, t_{y_1})$ and $(t_{x_2}, t_{y_2})$ are applied to a coordinates position P, the final P' is calculated as:

$$P' = T(t_{x_2}, t_{y_2}) \cdot \{T(t_{x_1}, t_{y_1})\}$$

$$P' = \{T(t_{x_2}, t_{y_2}) \cdot T(t_{x_1}, t_{y_1})\} \cdot P$$

- composite transformation matrix for this sequence of translation is.

$$\begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & t_{x_1}+t_{x_2} \\ 0 & 1 & t_{y_1}+t_{y_2} \\ 0 & 0 & 1 \end{bmatrix}$$

or

$$= T(t_{x_1}+t_{x_2}, t_{y_1}+t_{y_2}, 1)$$

2) <u>Rotation</u>.

- Two successive rotation applied to point P produce the transformation position.

$$P' = R(\theta_2) \cdot \{R(\theta_1) \cdot P\}$$
$$= \{R(\theta_2) \cdot R(\theta_1)\} \cdot P$$

- By Multiplying the 2 rotation matrices, we can verify that the two successive rotations are additive

$$R(\theta_2) \cdot R(\theta_1) = R(\theta_1 + \theta_2)$$

So the final rotated coordinates can be calculated with the composite Rotation matrix as $P' = R(\theta_1 + \theta_2) \cdot P$.

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2\cos\theta_1 - \sin\theta_1\sin\theta_2 & -\cos\theta_2\sin\theta_1 - \sin\theta_2\cos\theta_1 & 0 \\ \cos\theta_1\sin\theta_2 + \sin\theta_1\cos\theta_2 & -\sin\theta_1\sin\theta_2 + \cos\theta_1\cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} (x, y, 1)$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = R(\theta_1+\theta_2) \cdot P.$$

3.) Scaling

2 sucessive scaling operations produces the following composit scaling matrix.

$$\begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

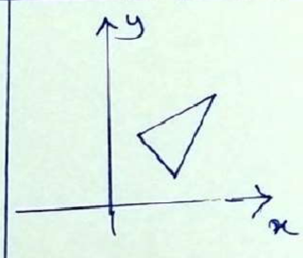
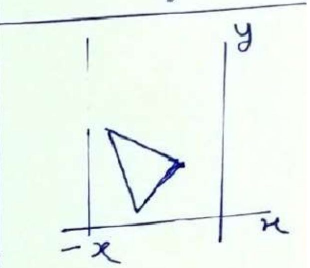
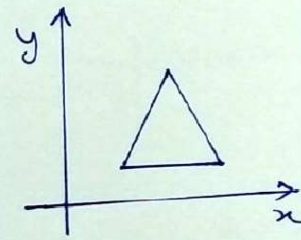
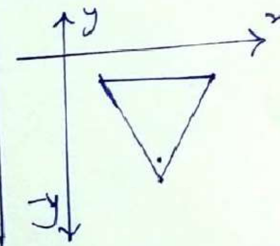
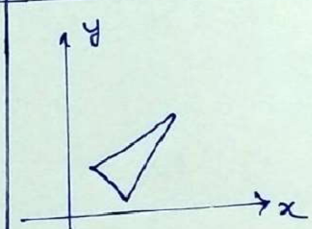
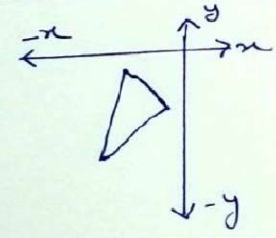
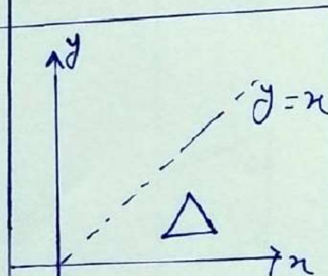
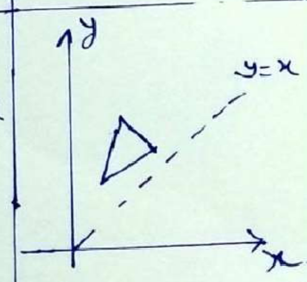
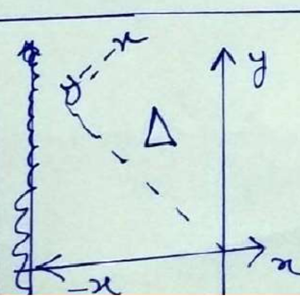
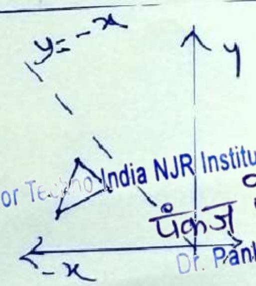
$$S.(Sx_1 \cdot Sx_2 , Sy_1 \cdot Sy_2)$$

Q1 Magnify a triangle A(0,0), B(1,1) and C(5,2) to twice of its size.

Q.2 Magnify a triangle place on A(0,0), B(1,1) and C(5,2) to twice of its size keeping point C(5,2) fixed.

# Reflection



| Reflection about | Transformation Matrix | Original Image | Reflected Image |
|---|---|---|---|
| y Axis | $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |   |   |
| x Axis | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |   |   |
| about origin. | $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |   |   |
| about line y=x | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |   |   |
| about line y = -x | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |   |   |

# Shear

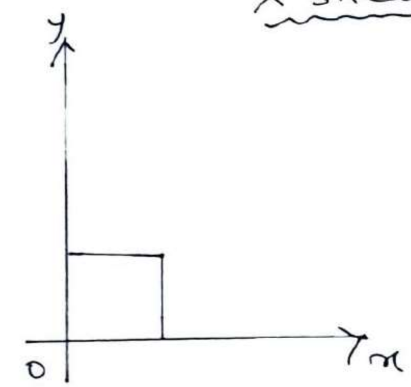


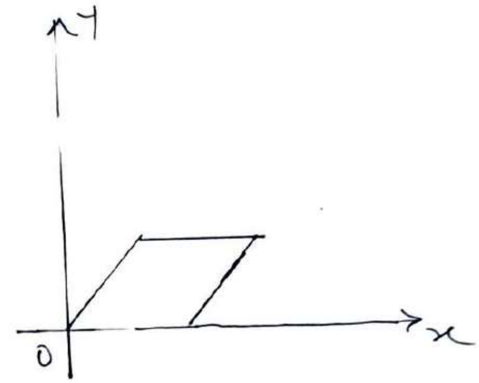
2 types of shear.

X shear (x will chance y will not change)

Y shear (y will chang x will not change)

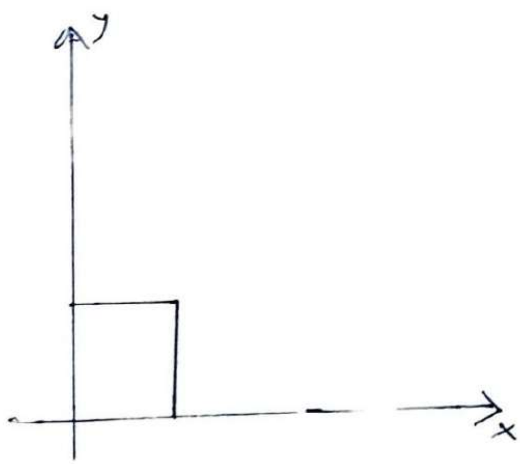
## X shear

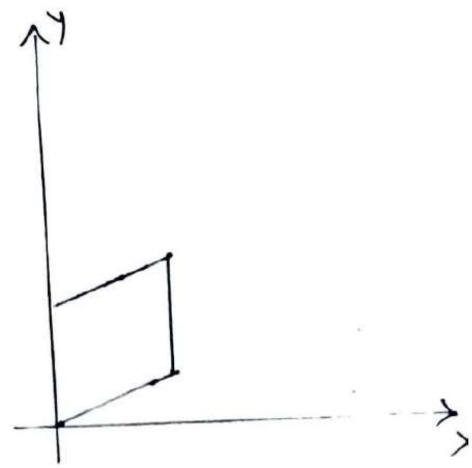



$$X.sh = \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x + Sh_x \cdot y$$
$$y' = y$$




$$Y.Sh = \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x$$
$$y' = y + Sh_x \cdot x$$

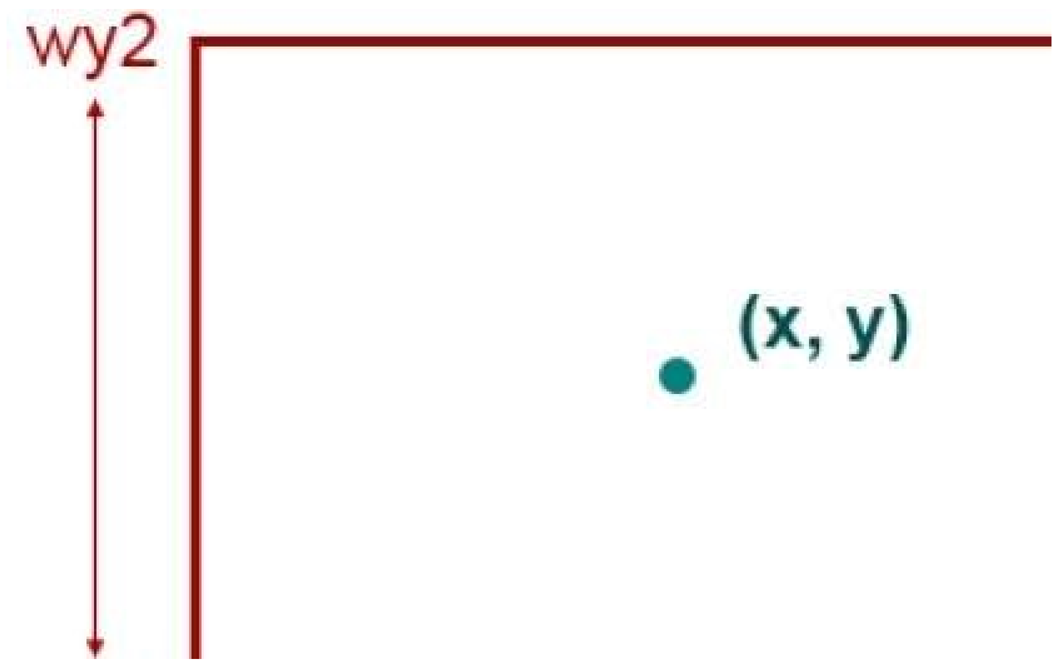
**Viewing & Clipping:**

The primary use of clipping in computer graphics is to remove objects, lines, or line segments that are outside the viewing pane.

**Point Clipping:**

Clipping a point from a given window is very easy. Consider the following figure, where the rectangle indicates the window. Point clipping tells us whether the given point X,YX,Y is within the given window or not; and decides whether we will use the minimum and maximum coordinates of the window.

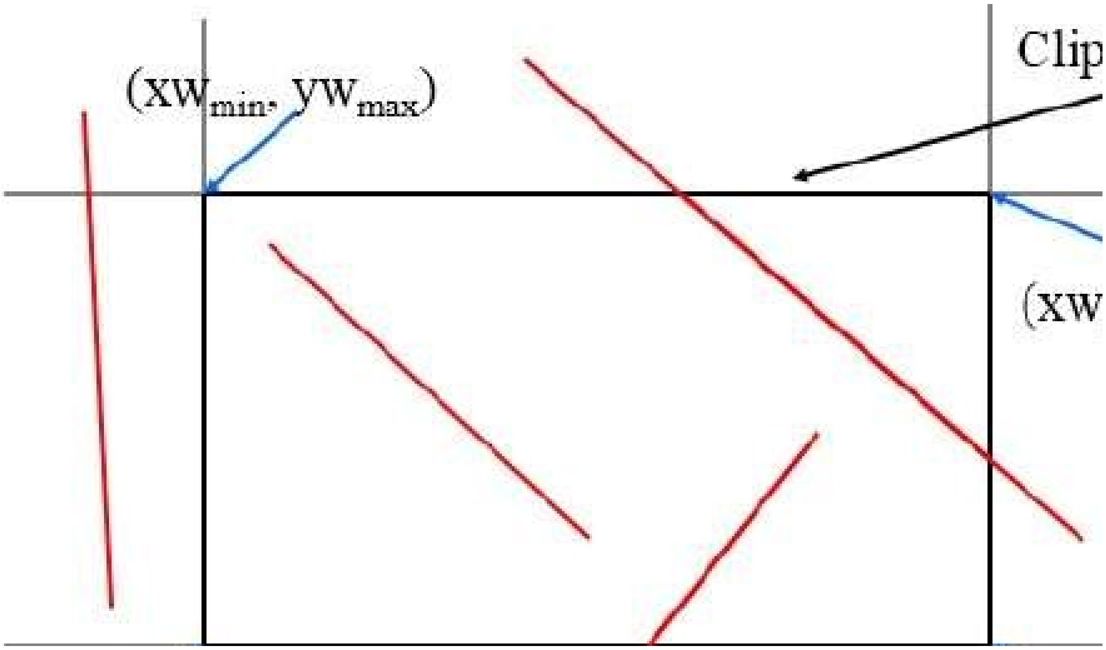
The X-coordinate of the given point is inside the window, if X lies in between $Wx1 \leq X \leq Wx2$. Same way, Y coordinate of the given point is inside the window, if Y lies in between $Wy1 \leq Y \leq Wy2$.



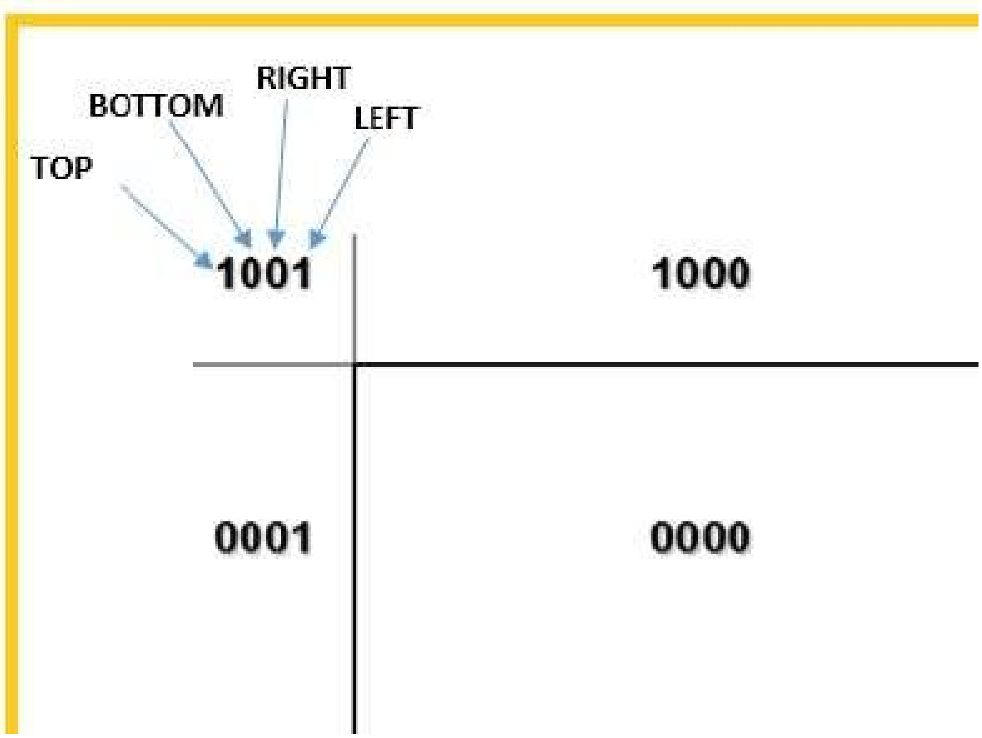

Line Clipping

**Cohen- Sutherland Line Clipping:**

This algorithm uses the clipping window as shown in the following figure. The minimum coordinate for the clipping region is (XWmin,YWmin)(XWmin,YWmin) and the maximum coordinate for the clipping region is (XWmax,YWmax)(XWmax,YWmax).




We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the **TOP** and **LEFT** bit is set to 1 because it is the **TOP-LEFT** corner.




Algorithm

**Step 1** − Assign a region code for each endpoints.

**Step 2** − If both endpoints have a region code **0000** then accept this line.

**Step 3** − Else, perform the logical **AND**operation for both region codes.

**Step 3.1** − If the result is not **0000,** then reject the line.

**Step 3.2** − Else you need clipping.

**Step 3.2.1** − Choose an endpoint of the line that is outside the window.

**Step 3.2.2** − Find the intersection point at the window boundary baseonregioncode.

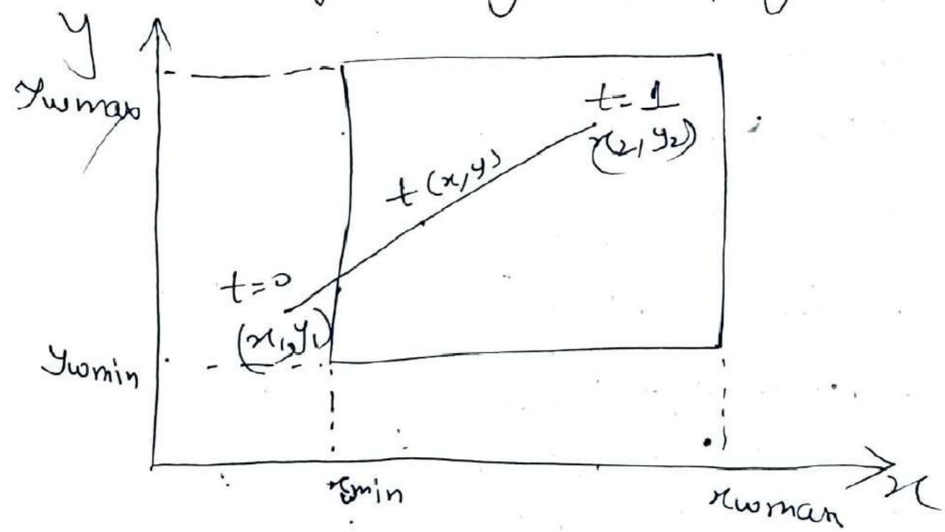
**Step 3.2.3** − Replace endpoint with the intersection point and update the region code.

**Step 3.2.4** − Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

**Step 4** − Repeat step 1 for other lines.

# Liang - Barsky Line clipping.

(1)




Line Parametric eqn.

$x = (1-t)x_1 + tx_2 \Rightarrow x = x_1 + t(x_2 - x_1) \Rightarrow x = x_1 + t\Delta x$

$y = (1-t)y_1 + ty_2 \Rightarrow y = y_1 + t(y_2 - y_1) \Rightarrow y = y_1 + t\Delta y$

As Per point clipping Algorithm.

$x_{wmin} \le x \le x_{wmax}$

$y_{wmin} \le y \le y_{wmax}$

Putting values of x & y at any point into above eqn.

$x_{wmin} \le x_1 + t\Delta x \le x_{wmax}$

$y_{wmin} \le y_1 + t\Delta y \le y_{wmax}$

we can rewrite above eqn as.

$x_1 + t\Delta x \ge x_{wmin}$

$x_1 + t\Delta x \le x_{wmax}$

$y_1 + t\Delta y \ge y_{wmin}$

$y_1 + t\Delta y \le y_{wmax}$

$t\Delta x \geq x_{wmin} - x_1$ — (1)

$t\Delta x \leq x_{wmax} - x_1$

$t\Delta y \geq y_{wmin} - y_1$ — (2)

$t\Delta y \leq y_{wmax} - y_1$

Multiply eq$^n$ (1) & (2) with -ve.

$-t\Delta x \leq x_1 - x_{wmin}$

$t\Delta x \leq x_{wmax} - x_1$

$-t\Delta y \leq y_1 - y_{wmin}.$

$t\Delta y \leq y_{wmax} - y_1$

General equation forms of above eq$^n$s.

$t\,P_k \leq q_k \quad \{k = 1, 2, 3, 4\}.$

| | |
|---|---|
| $P_1 = -\Delta x$ | $q_1 = x_1 - x_{wmin}$ |
| $P_2 = \Delta x$ | $q_2 = x_{wmax} - x_1$ |
| $P_3 = -\Delta y$ | $q_3 = y_1 - y_{wmin}.$ |
| $P_4 = \Delta y$ | $q_4 = y_{wmax} - y_1.$ |

if $p_k = 0$ Line parallel
if $q_k = 0$ Line is outside. $\{k = 1, 2, 3, 4\}$

if $p_k$ is Non zero

{

if $p_k < 0$ then

{

value to t will be

$$t = \max\left(0, \frac{q_k}{p_k}\right)$$

}

else ($p_k > 0$) then

{

$$t = \min\left(1, \frac{q_k}{p_k}\right)$$

}

Now check $t_1 > t_2$

Then line is completely outside; Reject the line

If $t_1 < t_2$ then

$x = x_1 + t\Delta x$

$y = y_1 + t\Delta y$.

Q window A(20,20) . B(90,20) , C(90,70) . D(20,70)

line $P_1(10,30)$ , $P_2(80,90)$.

$x_{wmin} = 20$ $x_1 = 10$ $x_2 = 80$

$y_{wmin} = 20$ $y_1 = 30$ $y_2 = 90$

$x_{wmax} = 90$ $\Delta x = 80 - 10 = 70$

$y_{wmax} = 70$ $\Delta y = 90 - 30 = 60$

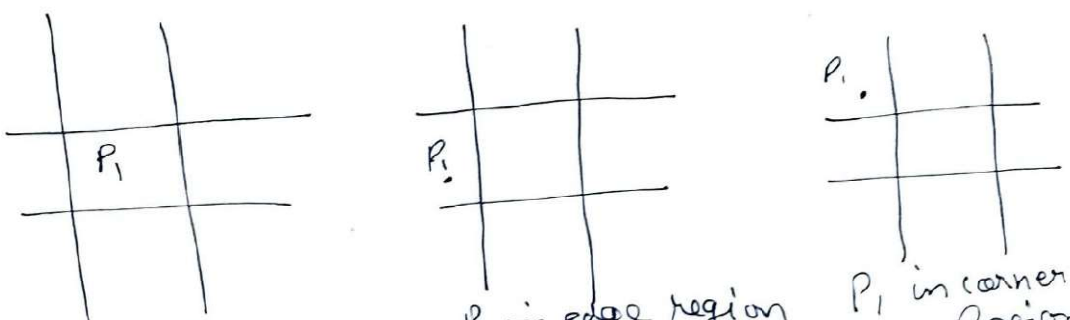
| | |
|---|---|
| $P_1 = -70$ | $q_1 = 10 - 20 = -10$ |
| $P_2 = 70$ | $q_2 = 90 - 10 = 80$ |
| $P_3 = -60$ | $q_3 = 30 - 20 = 10$ |
| $P_4 = 60$ | $q_4 = 70 - 30 = 40$ |

| $P_k < 0$ | $P_k > 0$ |
|---|---|
| $P_1, P_3$ | $P_2, P_4$ |
| $t_1 = \max(0, \frac{-10}{-70}, \frac{10}{-60})$ | $t_2 = \min(1, \frac{80}{70}, \frac{40}{60})$ |
| $t_1 = \frac{1}{7}$ | $t_2 = \frac{2}{3}$. |
| $x_1 = 10 + \frac{1}{7} \times 70$ | $x_1 = 10 + \frac{2}{3} \times 70 = 56.67$. |
| $x_1 = 20$ | $y_1 = 30 + \frac{2}{3} \times 60 = 70$. |
| $y_1 = 30 + \frac{1}{7} \times 60$ | |
| $y_1 = 38.57$ | |

$P_1'(20, 38.57)$ $P_2'(56.67, 70)$.

# Nicholl-lee-Nicholl (NLN).

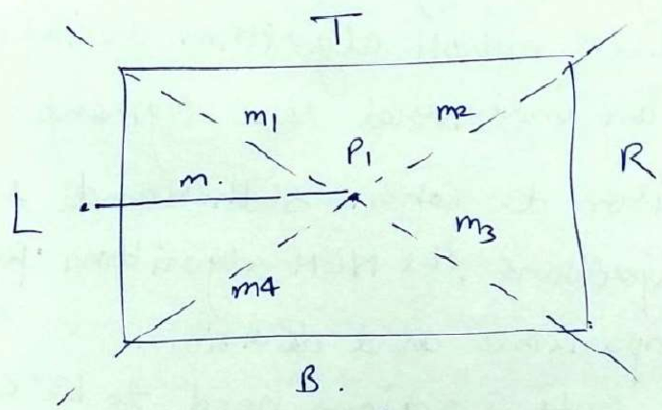
- By creating more regions around the clip window the nicholl-lee nicholl algorithm avoids multiple clipping of an individual line segment.
- In comparison to Cohen-sutherland & Liang-Barskey algorithms the NLN algorithm performs fewer comparisons and decision.
- In NLP only 3 regions need to be considered.




$P_1$ in window — $P_1$ in edge region — $P_1$ in corner Region.

- If $P_1$ lies in any other region we can move it to one of the 3 region by simple transformat-ion.
- If both the end points $P_1$ and $P_2$ are inside the clipping window. We Simply save the entire line.
- If $P_1$ is inside the clip window and $P_2$ is outside. Then set up the four region (L, T, R, B) The intersection with the appropriate window boundary is then calculated depending on

which one of the 4 region (T, B, L, R) contains $P_2$.

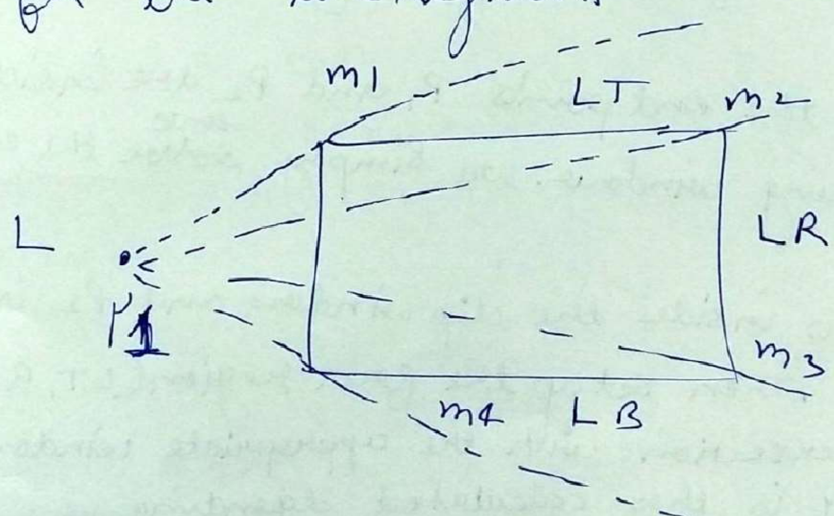



1) $m_1 < m < m_4$ → Left
2) $m_1 < m < m_4$ ⇒ TOP.
3) $m_2 < m < m_3$ → Right.
4) $m_3 < m < m_4$ → Bottom.

if $P_1$ is in the region to the left of the window

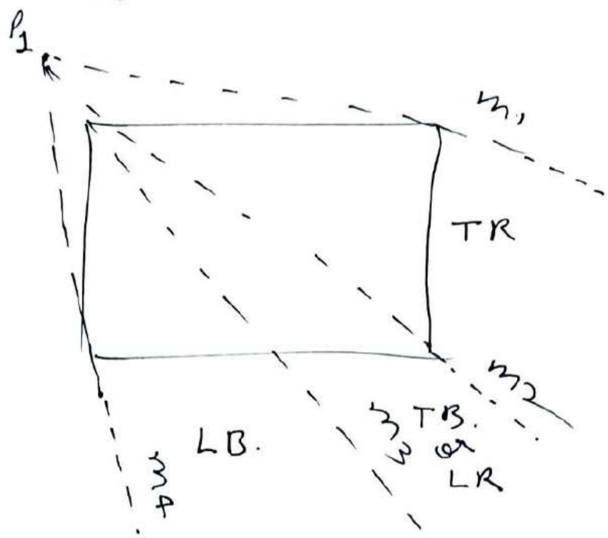
Then we set 4 region L, LT, LR, LB.

These 4 regions determines a unique boundary for the line segment



(1) $m_1 < m < m_2$ = left → Ltop
(2) $m_2 < m < m_3$ = left → Lright
(3) $m_3 < m < m_4$ = left → L Bottom
(4) $m_4 < m < m_1$ = discard

- If $P_1$ is to the left and above the clip window we use the clipping region in T, L, TR, TB, LR or LB this determines a unique clip window edge for the intersection calculation.

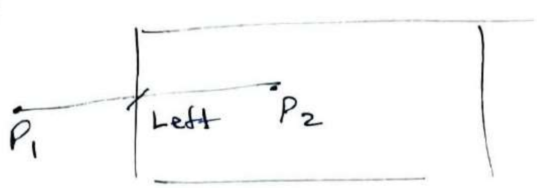



- Intersection Point calculation.

Parametric eqn.

$$x = x_1 + (x_2 - x_1)U$$
$$y = y_1 + (y_2 - y_1)U$$

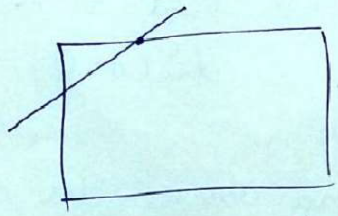
Case 1:




$x = x_L$ $\qquad U = \dfrac{x_L - x_1}{x_2 - x_1}$

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_L - x_1)$$

case 2:



$$y = y_T \qquad U = \frac{y_T - y_1}{y_2 - y_1}$$

$$x = x_1 - \frac{(x_2 - x_1)}{(y_2 - y_1)}(y_T - y_1)$$

**UNIT- V**

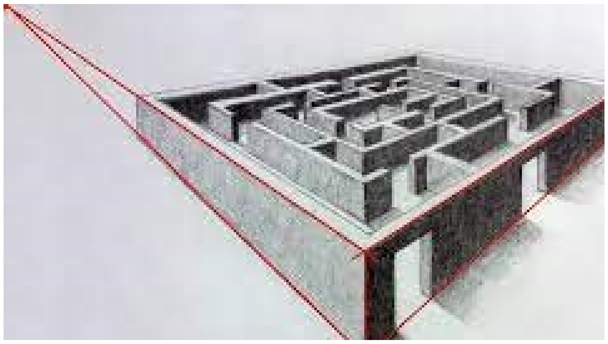
## 3D Display Method

In the 2D system, we use only two coordinates X and Y but in 3D, an extra coordinate Z is added. 3D graphics techniques and their application are fundamental to the entertainment, games, and computer-aided design industries. It is a continuing area of research in scientific visualization.

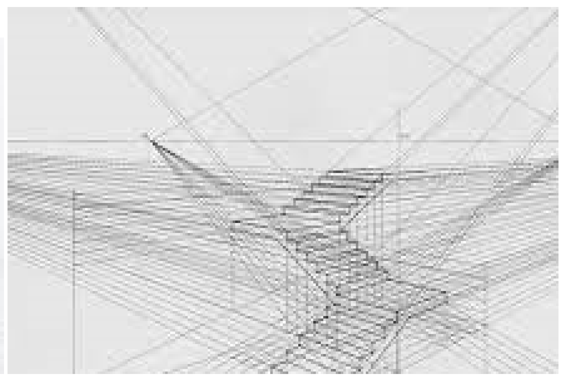
### Perspective Views:

1-Point Perspective


2- Point Perspective




3- Point Perspective

Polygon Surfaces

- Polygon surface can be thought of the surface composed of polygon faces

- The most commonly used boundary representation for a 3-D object is  a set of polygon surfaces that enclose the object interior

Methods of polygon surface representations are:

1. Polygon Table

2. Plane equation

3. Polygon Meshes

**Polygon Tables:** Representation of vertex coordinates, edges and other property of polygon into table form is called polygon table

Polygon data tables can be organized into two groups:

(i) Geometric Table

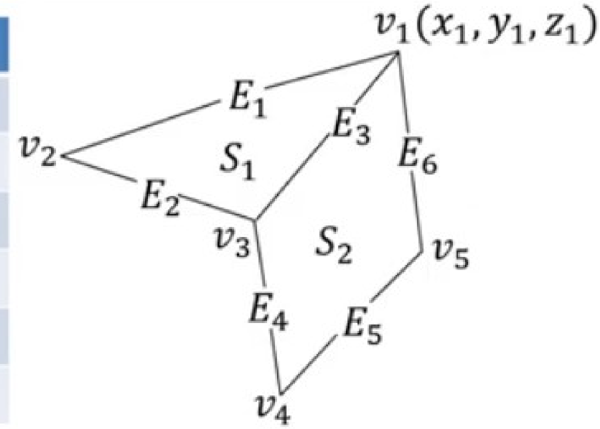
(ii) Attribute Table

**Geometric Table:** Contains vertex coordinate and the other parameter which specify geometry of polygon.

**Attribute Table:** stores other information like color, transparency etc.

| Vertex Table | |
|---|---|
| $v_1$: | $x_1, y_1, z_1$ |
| $v_2$: | $x_2, y_2, z_2$ |
| $v_3$: | $x_3, y_3, z_3$ |
| $v_4$: | $x_4, y_4, z_4$ |
| $v_5$: | $x_5, y_5, z_5$ |

| Edge Table | |
|---|---|
| $E_1$: | $v_1, v_2$ |
| $E_2$: | $v_2, v_3$ |
| $E_3$: | $v_3, v_1$ |
| $E_4$: | $v_3, v_4$ |
| $E_5$: | $v_4, v_5$ |
| $E_6$: | $v_5, v_1$ |




| Polygon Surface Table | |
|---|---|
| $S_1$: | $E_1, E_2, E_3$ |
| $S_2$: | $E_3, E_4, E_5, E_6$ |

**Plane Equation:**

General equation of plane is give =n as,

$$Ax + By + Cz + D = 0$$

Where (x,y,z) is any point on tha plane and A, B, C and D are constant.

We can obtain the values of A, B, C, and D by solving a set of three plane equations using the coordinate values for three non collinear points in the plane. Let us assume that three vertices of the plane are (x1, y1, z1), (x2, y2, z2) and (x3, y3, z3).

Let us solve the following simultaneous equations for ratios A/D, B/D, and C/D. You get the values of A, B, C, and D.

A/D x1 + B/D y1 + C/D z1 = -1

A/D x2 + B/D y2 + C/D z2 = -1

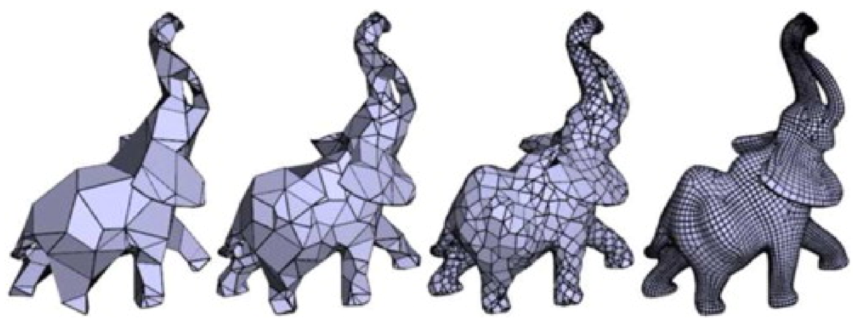
A/D x3 + B/D y3 + C/D z3 = -1

**Polygon Meshes:**

3D surfaces and solids can be approximated by a set of polygonal and line elements. Such surfaces are called polygonal meshes. In polygon mesh, each edge is shared by at most two polygons. The set of polygons or faces, together form the “skin” of the object.

This method can be used to represent a broad class of solids/surfaces in graphics. A polygonal mesh can be rendered using hidden surface removal algorithms. The polygon mesh can be represented by three ways −

- Explicit representation
- Pointers to a vertex list
- Pointers to an edge list



Curves in Computer Graphics

Representation of curves
B-Spline Curve
Bezier Curve
Parametric Curve
Spline Curve

# CURVE & REPRESENTATION

**WHAT IS CURVE :-** There are lots of definitions for Curve but we will focus on 2 main definitions for our understanding.

**DEF I :-** When sets of points infinite or finite are Joined Continous then what we get is called Curve.

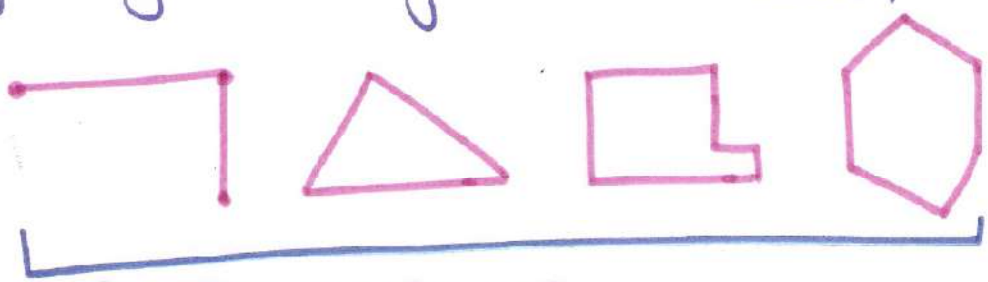
**DEF 2-** When we start from a point for drawing a geometrical figure and end at some other point without any GAP, so what we get is called CURVE.

**One Question comes in mind that as per definition is LINE ALSO A CURVE?**

Curve ?

YES, Mathematically a line is also Curve.

If LINE is a Curve then all the geometrical figures generated by Line also a Curve?

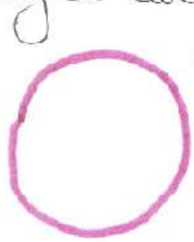


Are they all Curve?

As the Mathematics Says all the above figures are Curve.

But we focus here on other definition as well which Says

In Mathematics, a Curve is generally speaking, an object Similar to a line but that need not to be Straight. Thus, a Curve is generalization of a line, in that it may be Curved (bend, Smoothness).

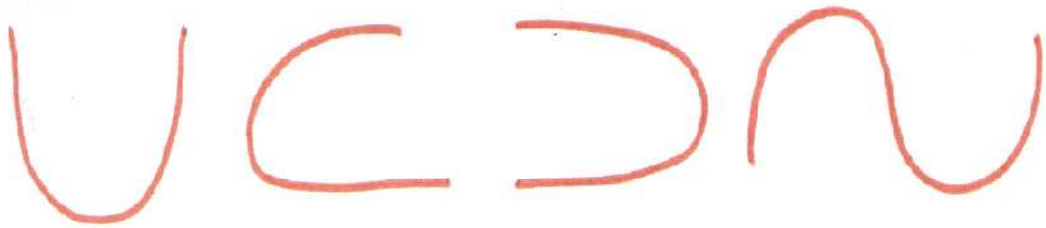


This Circle is Curve

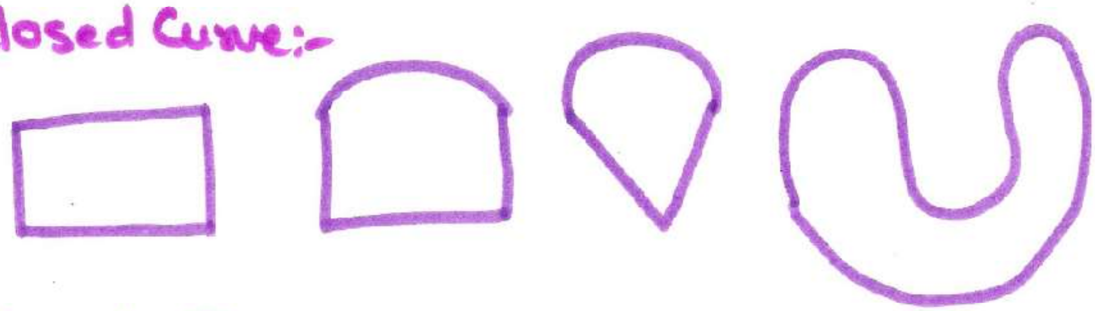
but this is Not as there is GAP.

There are many types of curves like:-

Open Curve:-



Closed Curve:-



Crossing Curve:-

# REPRESENTING CURVES :-

In Computer Graphics we daily need to draw or design different types of objects which are not flat but have bends and deviations and most importantly Smoothness.

Like Human face, Automobiles designs and many more.

So Computer need to Calculate or Compute all Curves so they can provide the Smoothness in Curve.

We Can represent basically Curves by 3 mathemati-cal function

Curve Representation

- Explicit Fn for Curve
- Implicit fn
- Parametric Curve fn.

## Explicit Representation of Curves:-

→ In this the dependent Variable has been given "Explicitly" in terms of the independent Variable. denoted as

$y = F(x)$ **Example:-** $y = ax^n + bx \ldots$

$y = 5x^3 + 2x + 1$

or $5x^2 + x$

Like a line $y = mx + c$

→ Explicit representation is Single valued for each Value of x only a Single value of y is Computed.

## Implicit Representation of Curves:-

In this dependent Variable is not expressed in terms of Some independent Variables.

$f(x, y) = 0$ $x^2 + y^2 - 1 = 0$

$y^4 + x^3 + 18 = 0$

It Can represent multivalued Curves (Multiple y Values for an x value) $x^2 + y^2 - R^2 = 0$ Circle.

Although you can convert a implicit $f^n$ into Explicit $f^n$ but generally it should not be done. b/c.

The new explicit function becomes very complex and some times also gives two different function branch.

For example :- If we convert implicit curve $x^2+y^2+1=0$ to explicit curve it will give us

$$y = \pm\sqrt{1-x^2}$$

Now new explicit $f^n$ become very complex and some times it gives us 2 branches.

here y has 2 branch one is +ve & second is -ve.

# PARAMETRIC CURVES:-

→ Most of the Curve representation's Follow the parametric form.

→ Curves having parametric form are called parametric Curves.

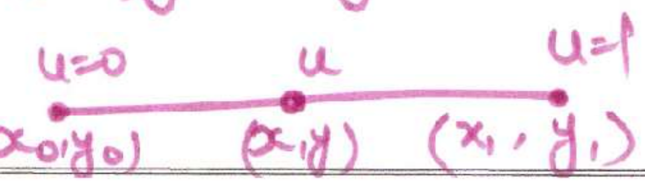
→ There are many Curves which we Cannot write down as a Single equation in terms of only $x$ and $y$.

→ Instead of defining $y$ in terms of $x$ $(y=f(x))$ or $x$ in terms of $y$ $(x=h(y))$ we defining both $x$ and $y$ in terms of a third variable called a Parameter

$$x = f_x(u) \quad u \text{ is parameter.}$$
$$y = f_y(u)$$

Like line Parametric equation is

$$x = (1-u)x_0 + ux_1$$
$$y = (1-u)y_0 + uy_1$$



# B-SPLINE CURVE

We have Some limitations in Bezier Curve like →

1) The Bezier Curve produced by Bernstein basis fn has limited Flexibility.

Numbers of Control points decides the degree of the Polynomial Curve. Ex:- 4 Control points results a Cubic polynomial Curve.

So only one way to reduce the degree of the Curve is to reduce the no. of Controlpoints and vice versa.

2) The Second limitation is that the Value of the blending fn is non-zero for all parameter values over the entire Curve.

Due to this Change in one vertex, Changes the entire Curve and this eliminates the ability to produce a local Change with in a Curve.

So B-Spline Curve — Basis-Spline Curve is Solution of this limitations of Bezier Curve.

## Properties of B-Spline Curve :-

1) B-Spline basis is non-global (LOCAL) effect.
In this each Control point affects the shape of the Curve Only over range of parameter Values where its associated basis fn is non zero.

2) B-Spline Curve made up of n+1 Control point

3) B-Spline Curve let us specify the order of basis (k) fn and the degree of the resulting Curve is independent on the no. of vertices.

4) It is possible to change the degree of the resulting Curve With out changing the no. of control points.

5) B-Spline can be used to define both open & close Curves.

6) Curve generally follows the shape of defining polygon
If we have order $k=4$ then degree will be 3 $P(k)=x^3$

7) The Curve line with in the Convex hull of its defining Polygon.

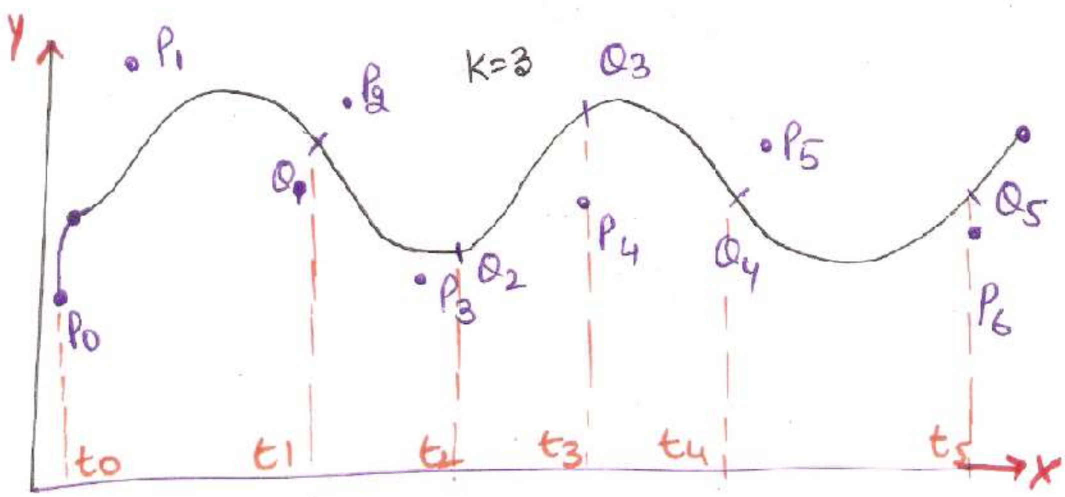
In B-Spline we Segment out the whole Curve which is decided by the order (K). by Formula 'n-K+2'

for Example:-

If we have 7 control points and order of Curve $k=3$ then $n=6$

and this B-Spline Curve has Segments

$6-3+2=5$




Five Segments $Q_1, Q_2, Q_3, Q_4, Q_5$

| Segment | Control points | Parameter |
|---|---|---|
| $Q_1$ | $P_0$ $P_1$ $P_2$ | $t_0=0$, $t_1=1$ |
| $Q_2$ | $P_1$ $P_2$ $P_3$ | $t_1=1$, $t_2=2$ |
| $Q_3$ | $P_2$ $P_3$ $P_4$ | $t_2=2$, $t_3=3$ |
| $Q_4$ | $P_3$ $P_4$ $P_5$ | $t_3=3$ $t_4=4$ |
| $Q_5$ | $P_4$ $P_5$ $P_6$ | $t_4=4$ $t_5=5$ |

There will be a join point or knot between $Q_{i-1}$ & $Q_i$ for $i \geq 3$ at the parameter value $t_i$ know as KNOT VALUE [X].

If $P(u)$ be the position vectors along the curve as a fn of the parameter $u$, a B-Spline curve is given by

$$P(u) = \sum_{i=0}^{n} P_i \, N_{i,k}(u) \qquad 0 \leq u \leq n-k+2$$

$N_{i,k}(u)$ is B-Spline basis fn

$$N_{i,k}(u) = \frac{(u - x_i) N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u) N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}}$$

The values of $x_i$ are the elements of a knot vector satisfying the relation $x_i \leq x_{i+1}$.

The parameter $u$ varies from 0 to $n-k+2$ along the $P(u)$

So there are Some Conditions for finding the KNOT VALUES [X]

$X_i \ (0 \leq i \leq n+k) \rightarrow$ Knot Values

$X_i = 0$ if $i < k$

$X_i = i-k+1$ if $k \leq i \leq n$

$X_i = n-k+2$ if $i > n$

So as B-Spline Curve has Recursive Eqn. so we stop at

$$N_{i,k}(u) = 1 \text{ if } x_i \leq u \ x_{i+1}$$

$$= 0 \text{ otherwise}$$

Example :-

$n=5, \ k=3$

then $X_i \ (0 \leq i \leq 8)$ Knot Values

$X_i \{0, 0, 0, 1, 2, 3, 4, 4, 4\}$ ($X_0, X_1, X_2, X_3, \ldots, X_8$)

$$N_{0,3}(u) = (1-u)^2 \cdot N_{2,1}(u)$$

After Calculation.

→ when $i=0, k=3$ so $i<k$ is true
$X_0 = 0$

→ $i=1, k=3$ $X_1 = 0$
$i=2, k=3$ $X_2 = 0$
$i=3, k=3$ $X_3 = i-k+1 = 3-3+1$
$X_3 = 1$
$i=4, k=3$ $X_4 = i-k+1 \Rightarrow 4-3+1$
$X_4 = 2$
⋮
$i=8, k=3$ $X_8 = i>n$ $n-k+2 = 5-3+2 = 4$
$X_8 = 4$

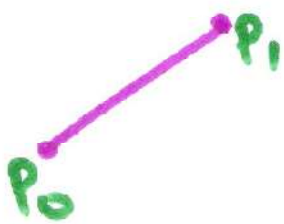
In this way we will Calculate.

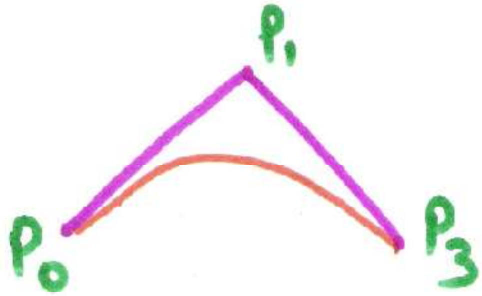
# BEZIER CURVE

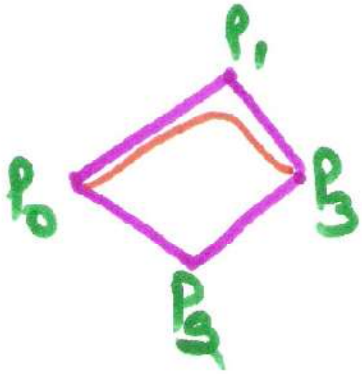
→ Bezier Curve is another approach for the Construction of the Curve.

→ It is approximate spline Curve.

→ Instead of endspoints and tangents, we have four Control points in the Case of Cubic Bezier Curve.




→ Bezier splines are widely used in various CAD System, COREL DRAW Packages and many more Graphic packages.

→ As with Splines, a bezier Curve Can be specified with boundary Conditions with a characterizing matrix or with blending $f^n$. For general bezier Curves, the blending Function specification is most Convenient.

Let Suppose we are given (n+1) control points positions. then $P_i = (x_i, y_i, z_i)$ with i varying from 0 to n. These coordinate points Can be blended to produce the following position vector P(u), which Describes the path of an approximation So Bezier polynomial Fn b/w $P_0$ to $P_n$ is

$$P(u) = \sum_{i=0}^{n} P_i \, B_{i,n}(u) \quad 0 \leq u \leq 1$$

$P_i$ Control Points

$B_{i,n}$ or $BEZ_{i,n}$ is Bezier Fn or Barstein Polynomials.

→ The Bernstein polynomial or the Bezier fn is very important fn which will dictate the smoothness of this curve & the weight will be dictated by boundary conditions.

$$BEZ_{i,n}(u) = {}^{n}C_i \cdot u^i (1-u)^{n-i}$$

where

$${}^{n}C_i = \frac{n!}{i!\,(n-i)!} \quad \text{[Binomial Coefficient]}$$

For Individual Coordinates

$$X(u) = \sum_{i=0}^{n} x_i \, BEZ_{i,n}(u)$$

$$Y(u) = \sum_{i=0}^{n} y_i \, BEZ_{i,n}(u)$$

$$Z(u) = \sum_{i=0}^{n} z_i \, BEZ_{i,n}(u)$$

# Bezier Curve For

## 3 points

$Q(u) = P_0 B_{0,2}(u) + P_1 B_{1,2}^{(u)} + P_2 B_{2,2}(u)$

Now Calculate $B_{0,2}$

$B_{0,2}^{(u)} = {}^2C_0 u^0 (1-u)^{2-0}$

$= \frac{2!}{0!\,2-0!} (1-u)^2 \cdot 1$

$= \frac{2\times 1}{2!} (1-u)^2$

$= 1 \cdot (1-u)^2 \Rightarrow (1-u)^2$

Now $B_{1,2}(u)$ in Same way

$= 2(1-u)u$

$B_{2,2}^{(u)} = u^2$

Now using in main Equation

$Q(u) = (1-u)^2 P_0 + 2 \cdot (1-u) \cdot u\, P_1 + u^2 P_2$

$x(u) = (1-u)^2 x_0 + 2 \cdot (1-u) u\, x_1 + u^2 x_2$

## 4 Points

$Q(u) = P_0 B_{0,3}^{(u)} + P_1 B_{1,3}^{(u)} + P_2 B_{2,3}^{(u)} + P_3 B_{3,3}^{(u)}$

Now We will Calculate $B_{0,3}^{(u)}$, $B_{1,3}(u)$ - - as we have Calculated & get

$B_{0,3}^{(u)} = (1-u)^3$

$B_{1,3}^{(u)} = 3u \cdot (1-u)^2$

$B_{2,3}^{(u)} = 3u^2 (1-u)$

$B_{3,3}^{(u)} = u^3$

Now Putting them in main Equation(s)

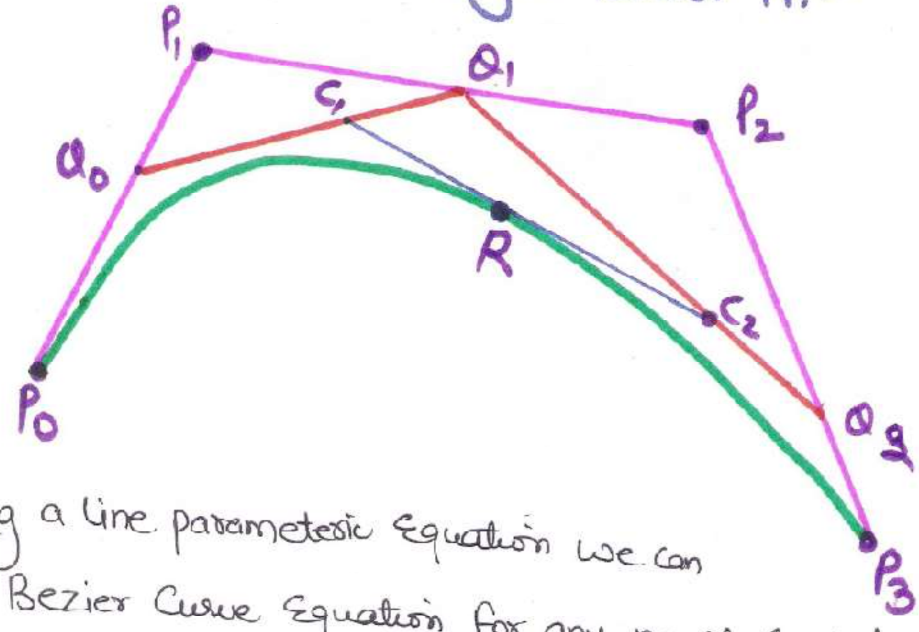
$Q(u) = P_0 (1-u)^3 + P_1\, u (1-u)^2 + P_2 \cdot 3u^2 (1-u) + u^3 P_4$

$x(u) = (1-u)^3 x_0 + u \cdot (1-u)^2 \cdot x_1 + 3u^2 (1-u) x_2 + x_3 \cdot u^3$

$y(u) =$ In Same way

$z(u) =$

Let See the main way of Calculating the Bezier Curve or where from we get Bezier fn:-




By using a line parametric Equation we can drive Bezier Curve Equation for any no. of Control points:-

$Q_0 = (1-u) P_0 + u P_1$ [ $Q_0$ Point on $P_0 \rightarrow P_1$
$Q_1 = (1-u) P_1 + u P_2$ $Q_1$ Point on $P_1 \rightarrow P_2$
$Q_2 = (1-u) P_2 + u P_3$ $Q_2$ Point on $P_2 \rightarrow P_3$ ]

$C_1 = (1-u) Q_0 + u \cdot Q_1$ [ $C_1$ Point on $Q_0 \rightarrow Q_1$
$C_2 = (1-u) Q_1 + u \cdot Q_2$ $C_2$ Point on $Q_1 \rightarrow Q_2$ ]

$R = (1-u) C_1 + u \cdot C_2$ [ R point on $C_1 \rightarrow C_2$ ]

Now we will use $C_1, C_2, Q_0, Q_1, Q_2$ Values in R:-

$$R(u) = (1-u)\,C_1 + u \cdot C_2$$

$$= (1-u)\left[(1-u)\,Q_0 + u \cdot Q_1\right] + u\left[(1-u)\,Q_0 + u \cdot Q_2\right]$$

$$= (1-u)^2 Q_0 + (1-u) \cdot u \cdot Q_1 + (1-u) \cdot u \cdot Q_1 + u^2 \cdot Q_2$$

$$= (1-u)^2\left[(1-u)P_0 + uP_1\right] + 2(1-u) \cdot u \cdot Q_1 + u^2\left((1-u)P_2 + uP_3\right)$$

$$= (1-u)^3 P_0 + (1-u)^2 \cdot u \cdot P_1 + 2(1-u) \cdot u \cdot \left[(1-u)P_1 + u \cdot P_2\right] + u^2\left((1-u)P_2 + u \cdot P_3\right)\Big]$$

$$= (1-u)^3 P_0 + (1-u)^2 \cdot u \cdot P_1 + 2(1-u)^2 \cdot u \cdot P_1 + 2(u-1) \cdot u^2 P_2 + (1-u) \cdot u^2 \cdot P_2 + u^3 \cdot P_3$$

$$= (1-u)^3 P_0 + 3(1-u)^2 \cdot u \cdot P_1 + 3(1-u) \cdot u^2 \cdot P_2 + u^3 \cdot P_3$$

For x, y, z coordinate

$$(1-u)^3 x_0 + 3(1-u)^2 \cdot u \cdot x_1 + 3(1-u) \cdot u^2 \cdot x_2 + u^3 \cdot x_3$$

Same equation which we get from Bernstein Polynomial form:—

## Propeties of Bezier Curves:-

(i) A very useful property of Bezier Curve is that it always Passes through the first and last Control points.

$$P(0) = P_0$$
$$P(1) = P_n$$

(ii) They generally follow the shape of the Control polygon which Consists of the Segments joining the Control points.

(iii) The Curve is contained within Convex hull of defining Polygon.

(iv) The degree of the Polynomial defining the Curve Segment is one less than the number of defining Control polygon points. For 4 Control points the degree of polynomial is 3. i.e Cubic Bezier Curve.

(v) It is quite easy to implement.

## Drawback:-

→ The degree of Bezier Curve depends on number of Control Points

# PARAMETRIC CURVE

The Parametric representation for Curves is :-

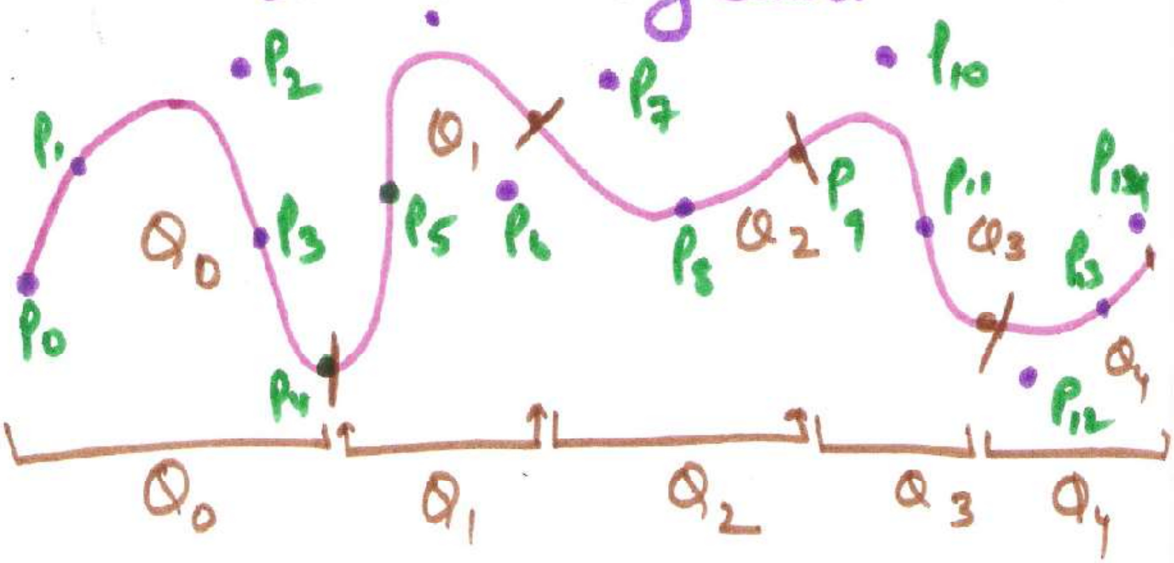
$$x = x(t)$$
$$y = y(t)$$
$$z = z(t)$$

A Curve is approximated by a piecewise polynomial Curve instead of piece linear Curve

| Piecewise Linear Curve | piecewise polynomial Curve. |
| --- | --- |
|  |  |
| by Polylines & using Linear Equation | Represented by Polynomial Equation. |

For drawing Curve we need to specify some points through which it may or maynot Completely Follow:-

let take a big Curve:-




$Q_0$, $Q_1$, $Q_2$ $Q_3$ & $Q_4$ are the Sections or Segment of big curve. & P's are Sample or Control Points

Each Segment Q of the overall Curves is given by three 3 Functions x, y, z which are Cubic polynomials in the Parameter t or u.

***

Cubic means here is that the polynomial Eq. Which is used to represent the Curve is has degree of 3

The Cubic polynomials that define a Curve Segment

$$Q(t) = [x(t) \;\; y(t) \;\; z(t)]$$

$$\left.\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_z t \cdot d_z \end{aligned}\right\} \quad (1)$$

$$0 \leq t \leq 1$$

where $T = [t^3 \; t^2 \; t \; 1]$

The Coefficient matrix is defined as

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \quad - (2)$$

So we can Rewrite equ. (1) as

$$Q(t) = [x(t) \; y(t) \; z(t)] = T \cdot C \quad \text{—} (3)$$

In General:-

In this C can be further be divided

$$C = M \cdot G$$

where $M = [m_g]_{4 \times 4}$ & $G = [g_1 \; g_2 \; g_3 \; g_4]^t$

M is a 4x4 basis matrix and G is a four element Column Vector of geometric constants, Called the geometric vector.

So $Q(t) = T \cdot M \cdot G$.

The Curve is a weighted Sum of the elements of the geometry matrix.

The weights are each Cubic polynomials of t, and are called the blending functions:-

$$B = T \cdot M$$

---

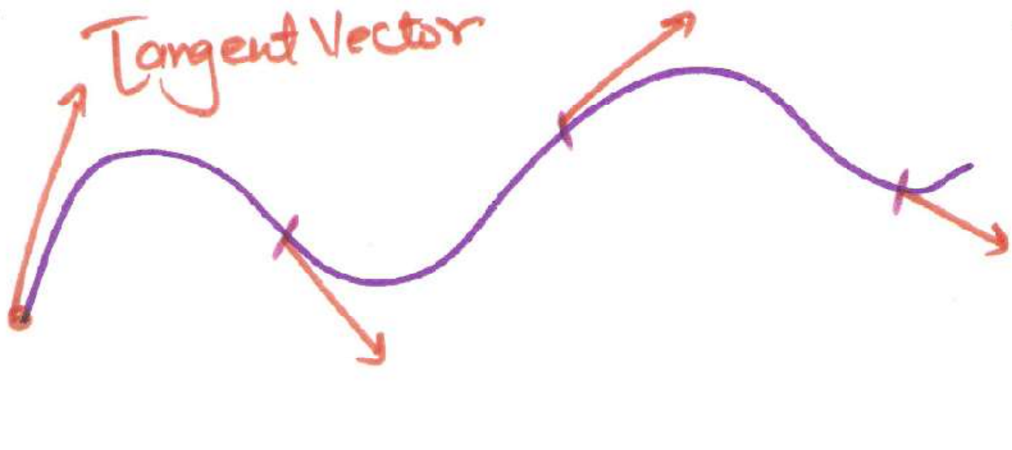
The parametric tangent-vector to the Curve is

$$\frac{d\,Q(t)}{d(t)} = Q'(t) = \left[\frac{d\,x(t)}{dt} \quad \frac{d\,y(t)}{dt} \quad \frac{d\,z(t)}{dt}\right]$$

$$= \frac{d\,T \cdot C}{dt}$$

$$= \left[3t^2 \quad 2t \quad 1 \quad 0\right] \cdot C$$

$$= \left[3a_x t^2 + 2b_x t + c_x \quad 3a_y t^2 + 2b_y t + c_y \quad 3a_z t^3 + 2b_z t + c_z\right]$$



# SPLINE CURVE

**Spline** :- Spline is a flexible strip which was long ago used for designing the Ships.

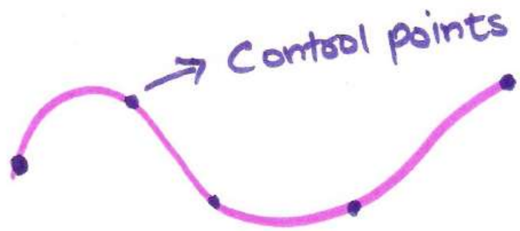
**Spline Curve**:- A Spline Curve is mathematical representation for which it is easy to build an interface that will allow a user to design and Control the Shape of Complex Curves & Surfaces.

Spline Curve mathematically described with a piecewise Cubic polynomial function whose first & Second derivatives are continuous across the various Curve Section. $C^1$ & $C^2$ Continuity.

**Control points:-** We specify a spline Curve by giving a Set of **Coordinate positions,** Called Control points. which indicates the general Shape of the Curve. These Control points are then fitted with piece wise Continuous parametric polynomial functions in on the 2 ways :-

## Interpolate or Interpolation Spline :-

When polynomial Sections are Fitted so that the Curve passes through all Control points , then the resulting Curve is said to be Interpolate the set of Control points.




## Approximate or Approximation Spline :-

When the polynomials are fitted to the path which is not necessarily passing through all Control points , the resulting Curve is Said to approximate the Set of Control points.

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Approximation Spline

Approximation Curves are Commonly used as design tools to Structure object Surface.

**3d viewing pipeline**

---

**Introduction:** In two-dimensional graphics applications, viewing operations transfer positions from the world-coordinate plane to pixel positions in the plane of the output device. Using the rectangular boundaries for the world-coordinate window and the device viewport, a two-dimensional package maps the world scene to device coordinates and clips the scene against the four boundaries of the viewport. For three-dimensional graphics applications, the situation is a bit more involved, since we now have more choices as to how views are to be generated. First of all, we can view an object from any spatial position: from the front, from above, or from the back.

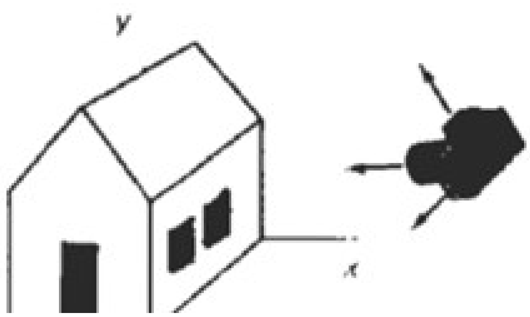



*Figure 12-1*
Photographing a sce

The steps for computer generation of a view of a three-dimensional scene are somewhat analogous to the processes involved in taking a photograph. To take a snapshot, we first need to position the camera at a particular point in space. Then we need to decide on the camera orientation (Fig. 12-1): Which way do we point the camera and how should we rotate it around the line of sight to set the up direction for the picture? Finally, when we snap the shutter, the scene is cropped to the size of the "window" (aperture) of the camera, and light from the visible surfaces is projected onto the camera film. We need to keep in mind, however, that the camera analogy can be carried only so far, since we have more flexibility and many more options for generating views of a scene with a graphics package than we do with a camera.

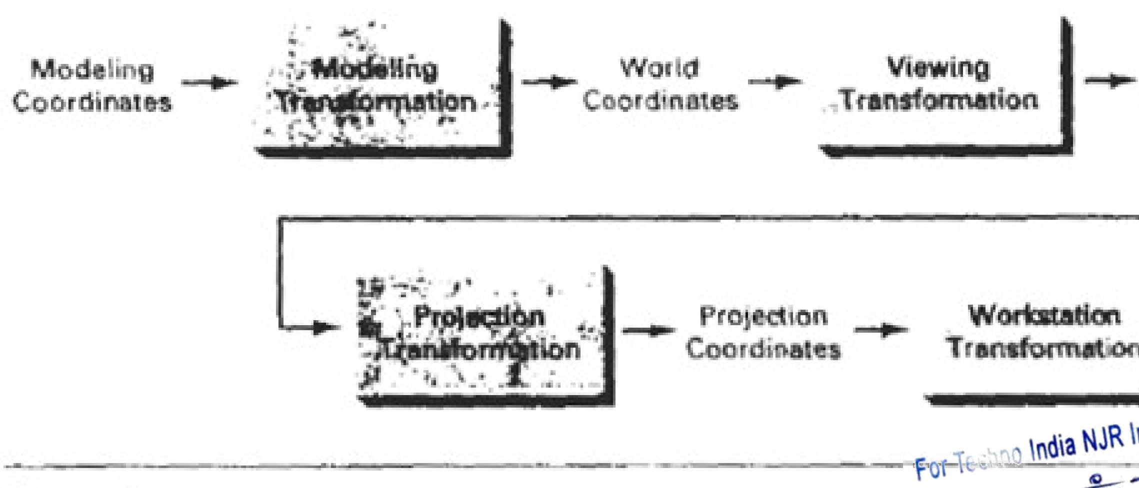
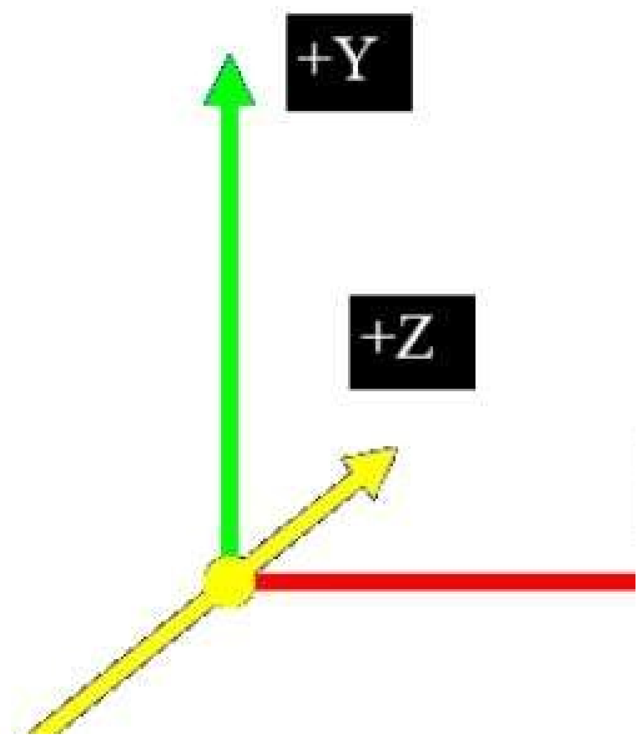


Figure 12-2 shows the general processing steps for modeling and converting a world-coordinate description of a scene to device coordinates. Once the scene has been modeled, world-coordinate positions are converted to viewing coordinates. The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane, which we can think of in analogy with the camera film plane. Next, projection operations are performed to convert the viewing-coordinate description of the scene to coordinate positions on the projection plane, which will then be mapped to the output device. Objects outside the specified viewing limits are clipped h m further consideration, and the remaining objects are processed through visible-surface identification and surface-rendering procedures to produce the display within the device viewport.

Projection:

In the 2D system, we use only two coordinates X and Y but in 3D, an extra coordinate Z is added. 3D graphics techniques and their application are fundamental to the entertainment, games, and computer-aided design industries. It is a continuing area of research in scientific visualization.

Furthermore, 3D graphics components are now a part of almost every personal computer and, although traditionally intended for graphics-intensive software such as games, they are increasingly being used by other applications.

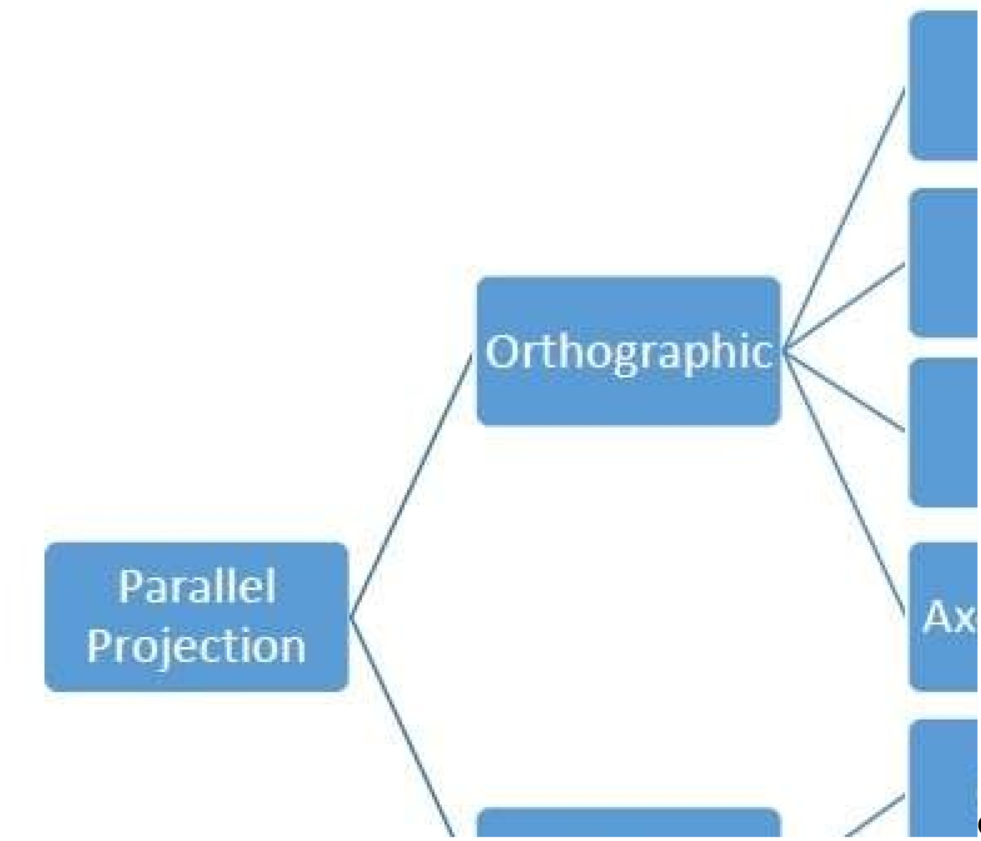



Parallel Projection

Parallel projection discards z-coordinate and parallel lines from each vertex on the object are extended until they intersect the view plane. In parallel projection, we specify a direction of projection instead of center of projection.

In parallel projection, the distance from the center of projection to project plane is infinite. In this type of projection, we connect the projected vertices by line segments which correspond to connections on the original object.

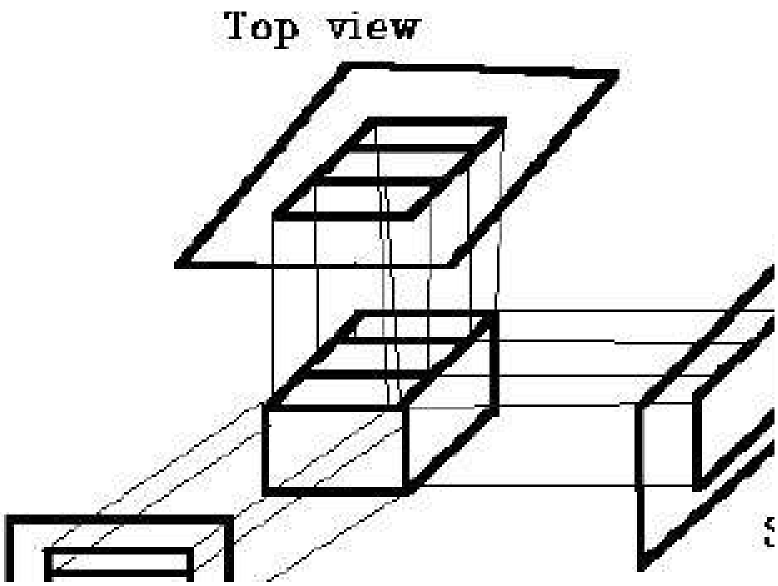
Parallel projections are less realistic, but they are good for exact measurements. In this type of projections, parallel lines remain parallel and angles are not preserved. Various types of parallel projections are shown in the following hierarchy.




Orthographic Projection

In orthographic projection the direction of projection is normal to the projection of the plane. There are three types of orthographic projections −

- Front Projection
- Top Projection
- Side Projection

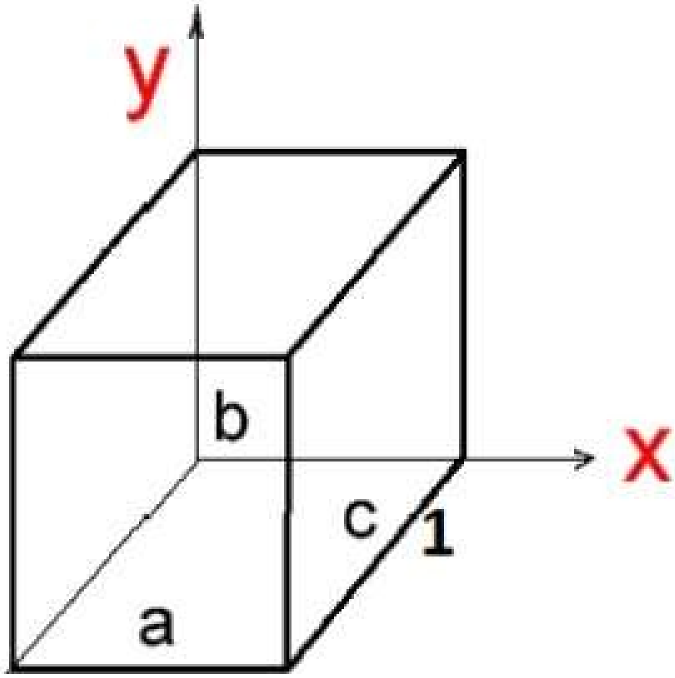


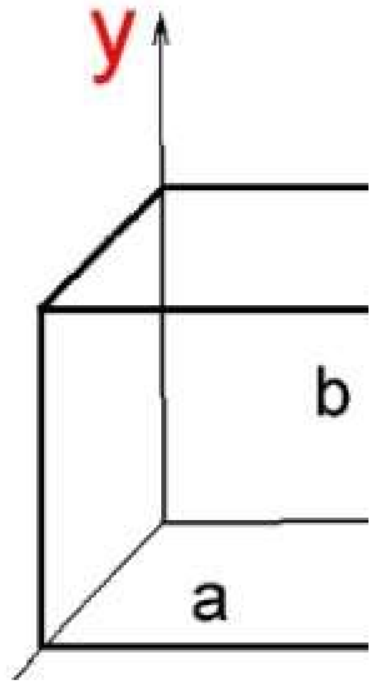

Oblique Projection

In oblique projection, the direction of projection is not normal to the projection of plane. In oblique projection, we can view the object better than orthographic projection.

There are two types of oblique projections − **Cavalier** and **Cabinet**. The Cavalier projection makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.

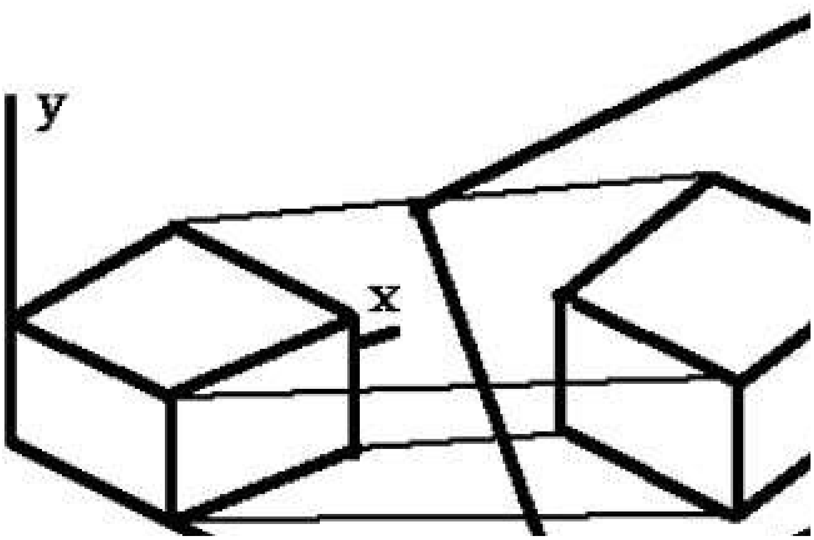
The Cabinet projection makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface are projected at ½ their actual length. Both the projections are shown in the following figure −




Isometric Projections

Orthographic projections that show more than one side of an object are called **axonometric orthographic projections**. The most common axonometric projection is an **isometric projection** where the projection plane intersects each coordinate axis in the model coordinate system at an equal distance. In this projection parallelism of lines are preserved but angles are not preserved. The following figure shows isometric projection −

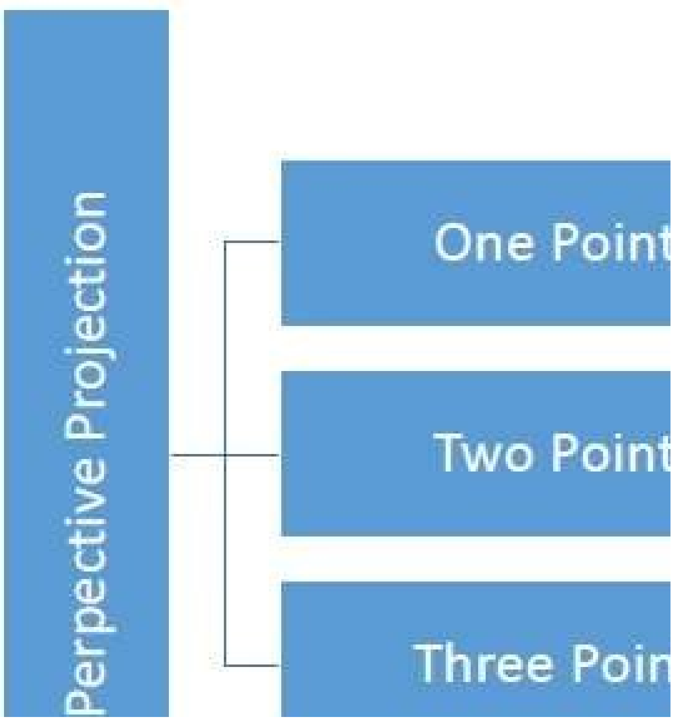



Perspective Projection

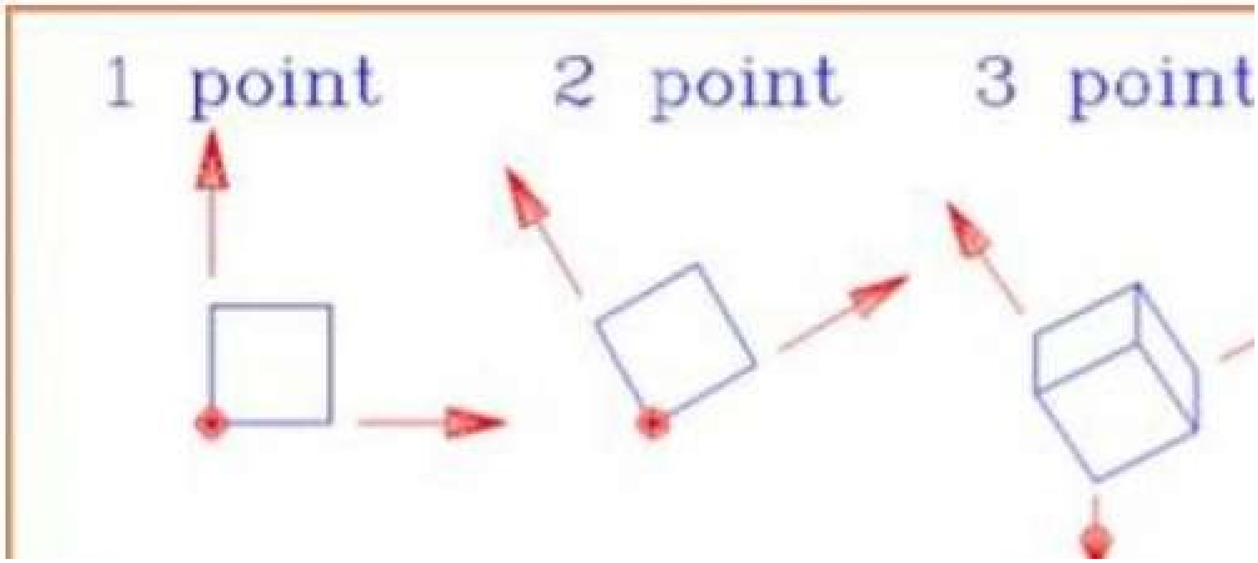
In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic.

The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called **center of projection** or **projection reference point**. There are 3 types of perspective projections which are shown in the following chart.

- **One point** perspective projection is simple to draw.
- **Two point** perspective projection gives better impression of depth.
- **Three point** perspective projection is most difficult to draw.



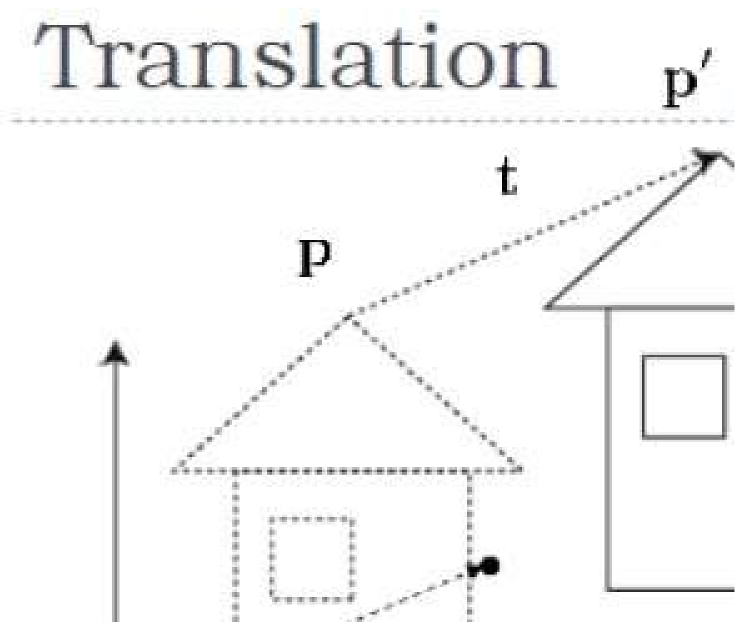

The following figure shows all the three types of perspective projection −




Translation

In 3D translation, we transfer the Z coordinate along with the X and Y coordinates. The process for translation in 3D is similar to 2D translation. A translation moves an object into a different position on the screen.

The following figure shows the effect of translation −




A point can be translated in 3D by adding translation coordinate $(tx,ty,tz)$(tx,ty,tz) to the original coordinate $X,Y,Z$X,Y,Z to get the new coordinate $X',Y',Z'$X′,Y′,Z′.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$P' = P \cdot T$$

$$[X' \;\; Y' \;\; Z' \;\; 1] = [X \;\; Y \;\; Z \;\; 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

3D Transformation

The geometric transformations play a vital role in generating images of three Dimensional objects with the help of these transformations. The location of objects relative to others can be easily expressed. Sometimes viewpoint changes rapidly, or sometimes objects move in relation to each other. For this number of transformation can be carried out repeatedly.

**1. Translation**

It is the movement of an object from one position to another position. Translation is done using translation vectors. There are three vectors in 3D instead of two. These vectors are in x, y, and z directions. Translation in the x-direction is represented using $T_x$. The translation is y-direction is represented using $T_y$. The translation in the z- direction is represented using $T_z$.

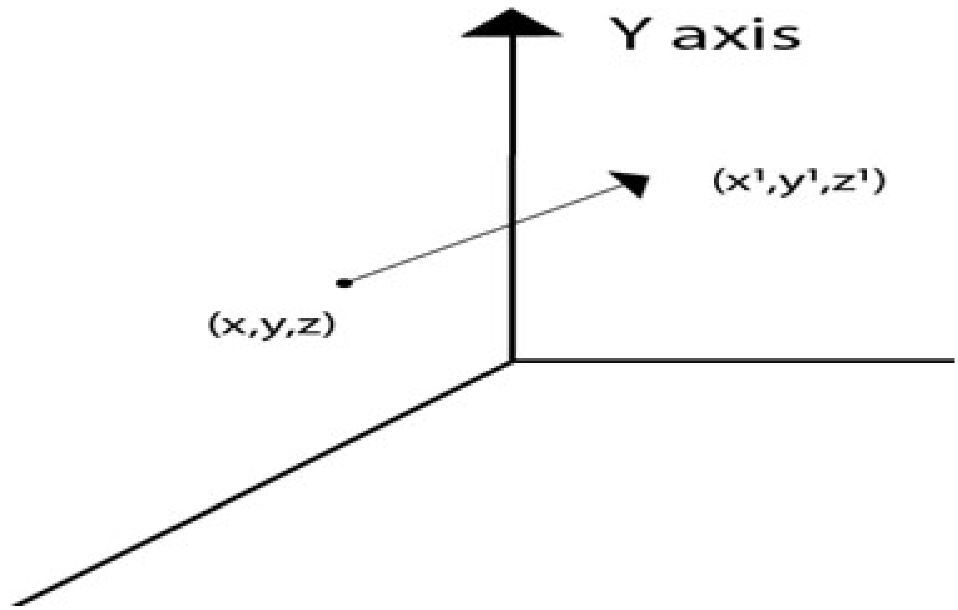
If P is a point having co-ordinates in three directions (x, y, z) is translated, then after translation its coordinates will be ($x^1$ $y^1$ $z^1$) after translation. $T_x$ $T_y$ $T_z$ are translation vectors in x, y, and z directions respectively.

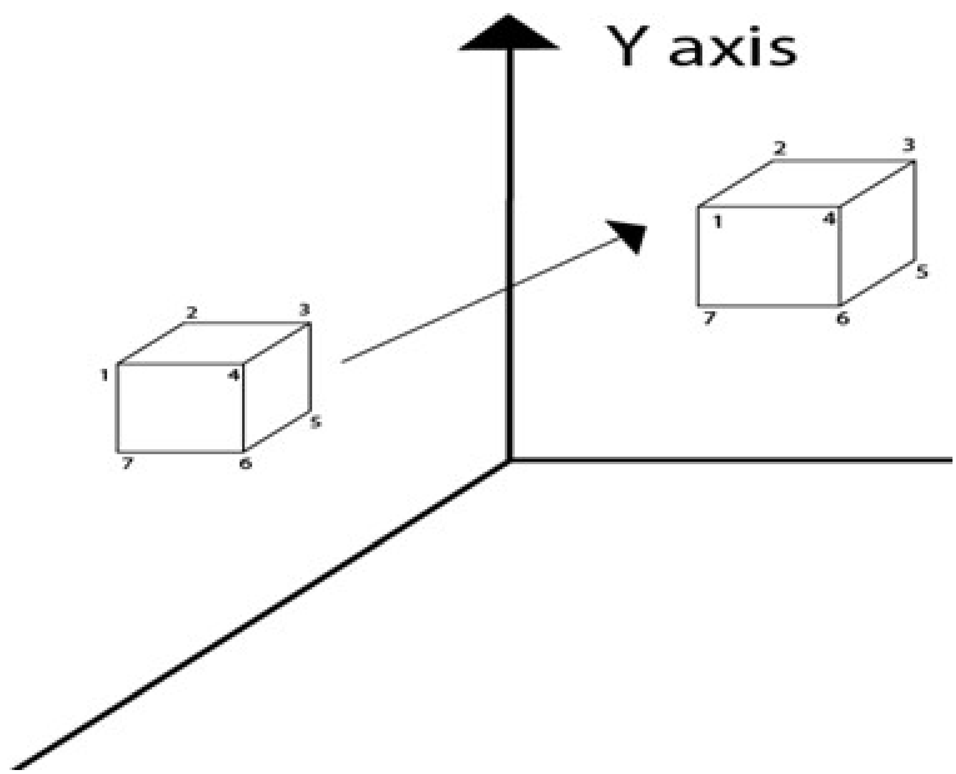
$$x^1 = x + T_x$$

$$y^1 = y + T_y$$

1

Three-dimensional transformations are performed by transforming each vertex of the object. If an object has five corners, then the translation will be accomplished by translating all five points to new locations. Following figure 1 shows the translation of point figure 2 shows the translation of the cube.




Matrix for translation

$$\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{Bmatrix} \text{ or } \begin{Bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{Bmatrix}$$

Matrix representation of point translation

Point shown in fig is (x, y, z). It become $(x^1, y^1, z^1)$ after translation. $T_x$ $T_y$ $T_z$ are translation vector.

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \end{pmatrix}$$

**Example:** A point has coordinates in the x, y, z direction i.e., (5, 6, 7). The translation is done in the x-direction by 3 coordinate and y direction. Three coordinates and in the z- direction by two coordinates. Shift the object. Find coordinates of the new position.

**Solution:** Co-ordinate of the point are (5, 6, 7)
Translation vector in x direction = 3
Translation vector in y direction = 3
Translation vector in z direction = 2
Translation matrix is

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Multiply co-ordinates of point with translation matrix

$$(x^1y^1z^1) = (5,6,7,1)\begin{pmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{pmatrix}$$

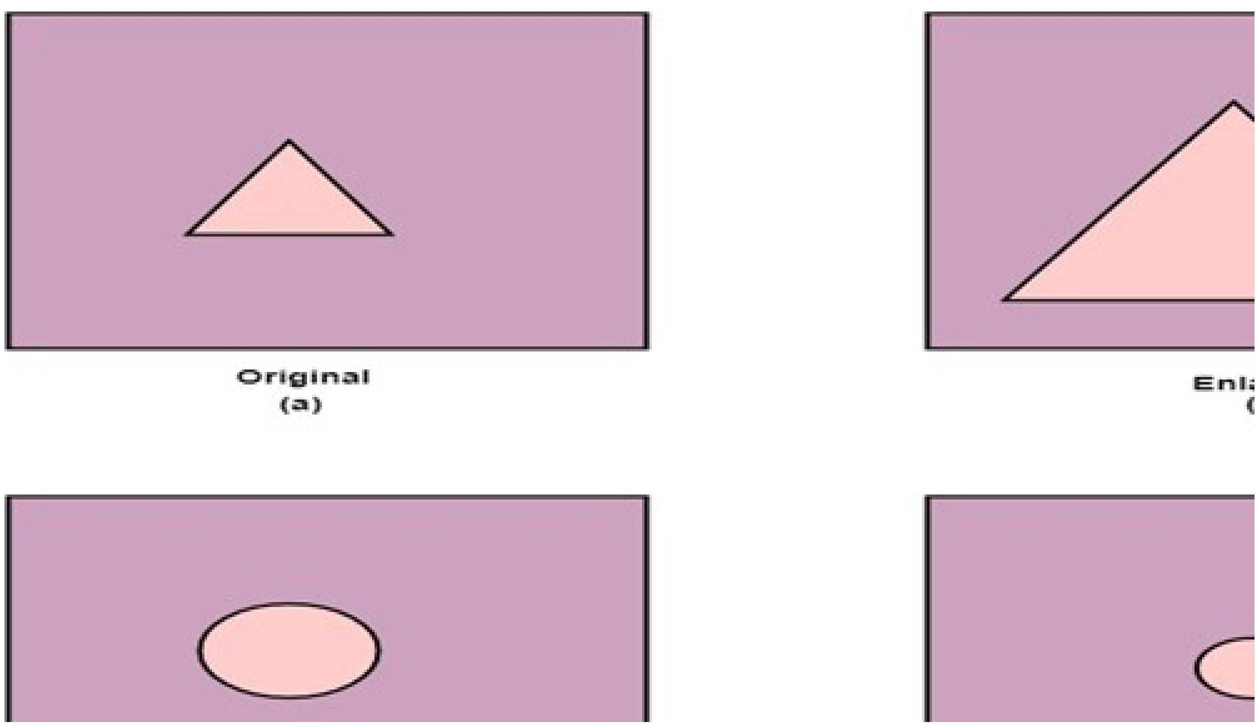
= [5+0+0+30+6+0+30+0+7+20+0+0+1] = [8991]

x becomes $x^1$=8
y becomes $y^1$=9
z becomes $z^1$=9

**2. Scaling**

Scaling is used to change the size of an object. The size can be increased or decreased. The scaling three factors are required $S_x$ $S_y$ and $S_z$.

$S_x$=Scaling factor in x- direction
$S_y$=Scaling factor in y-direction
$S_z$=Scaling factor in z-direction




Matrix for Scaling

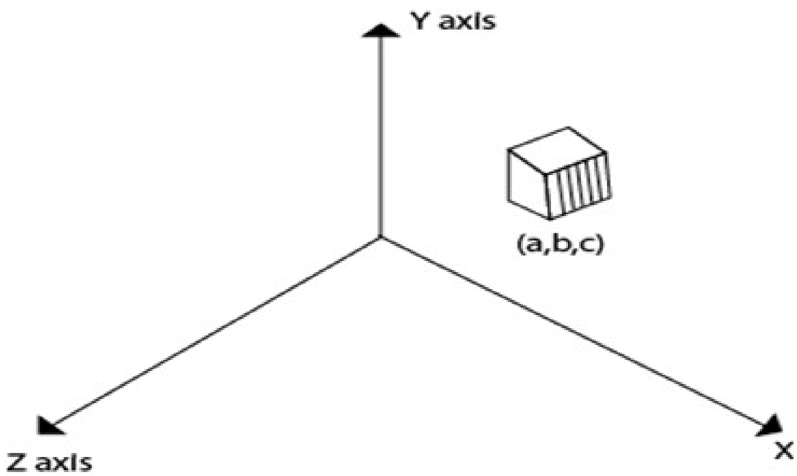
$$
\begin{Bmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0
\end{Bmatrix}
$$

## Scaling of the object relative to a fixed point

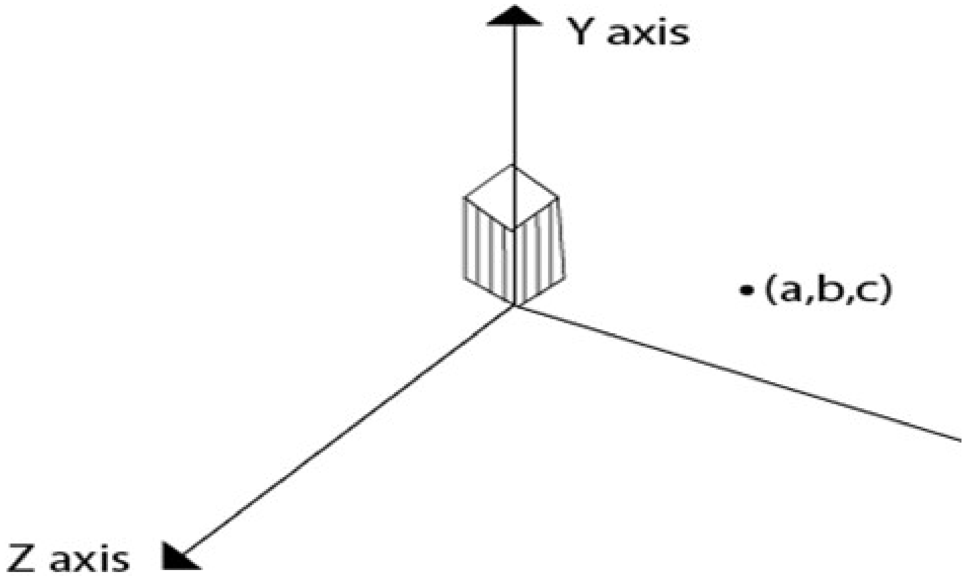
Following are steps performed when scaling of objects with fixed point (a, b, c). It can be represented as below:

1. Translate fixed point to the origin
2. Scale the object relative to the origin
3. Translate object back to its original position.

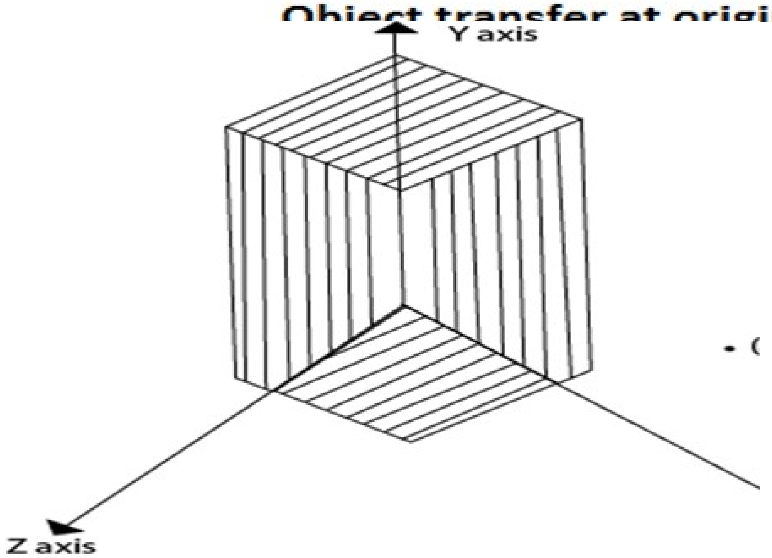
In figure (a) point (a, b, c) is shown, and object whose scaling is to done also shown in steps in fig (b), fig (c) and fig (d).

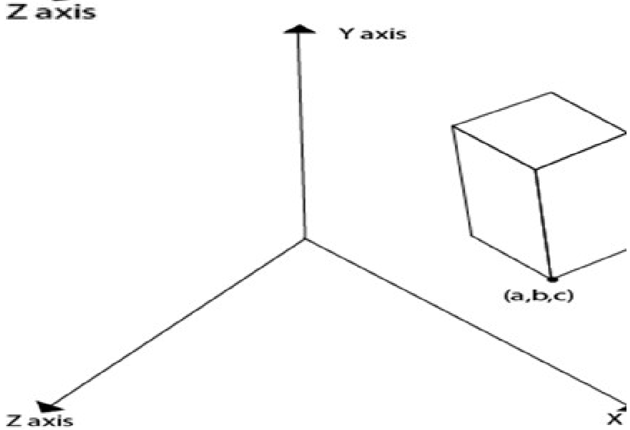



**Object & point for scaling**




Object transfer at origin

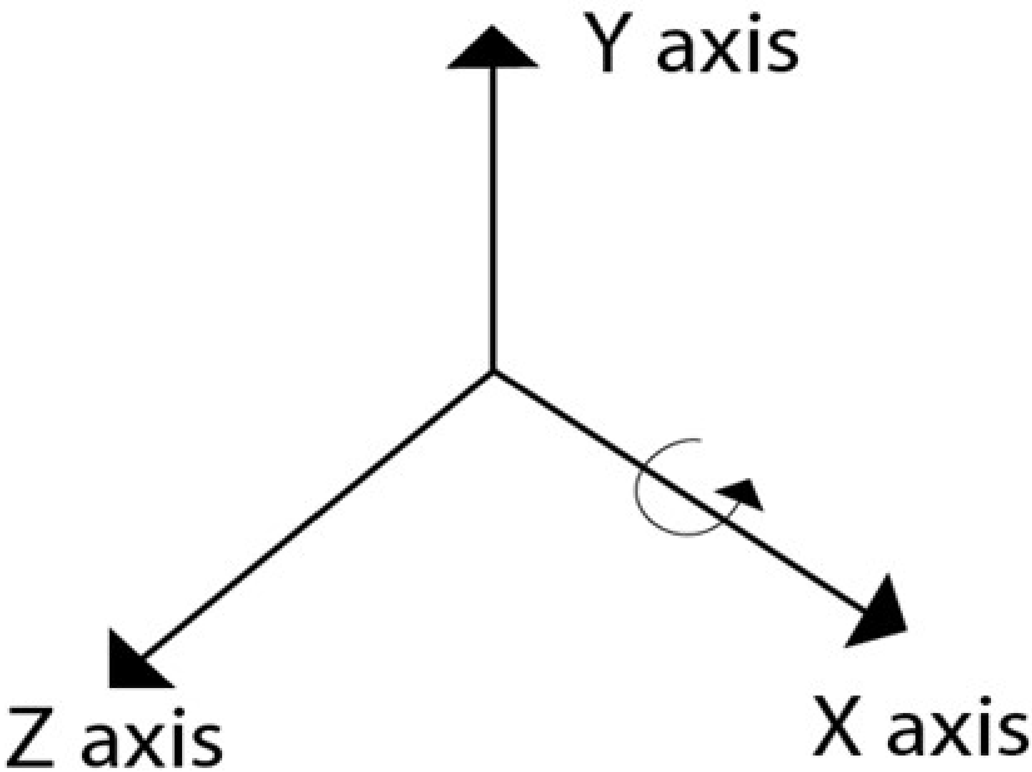


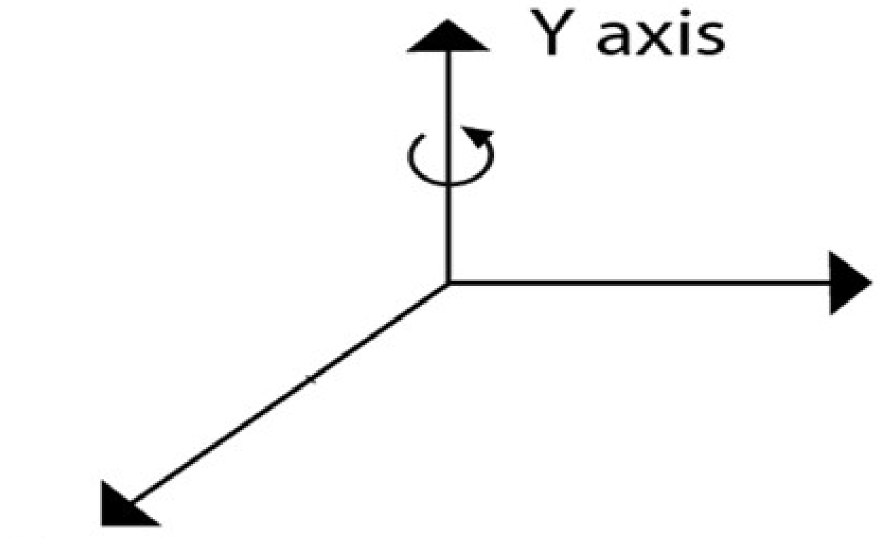



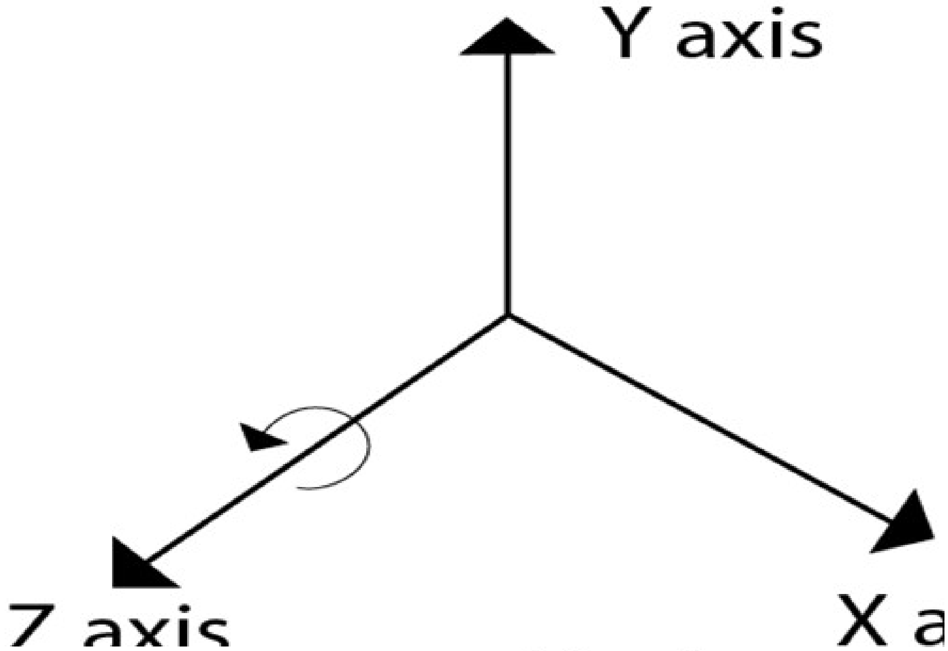
**3. Rotation**

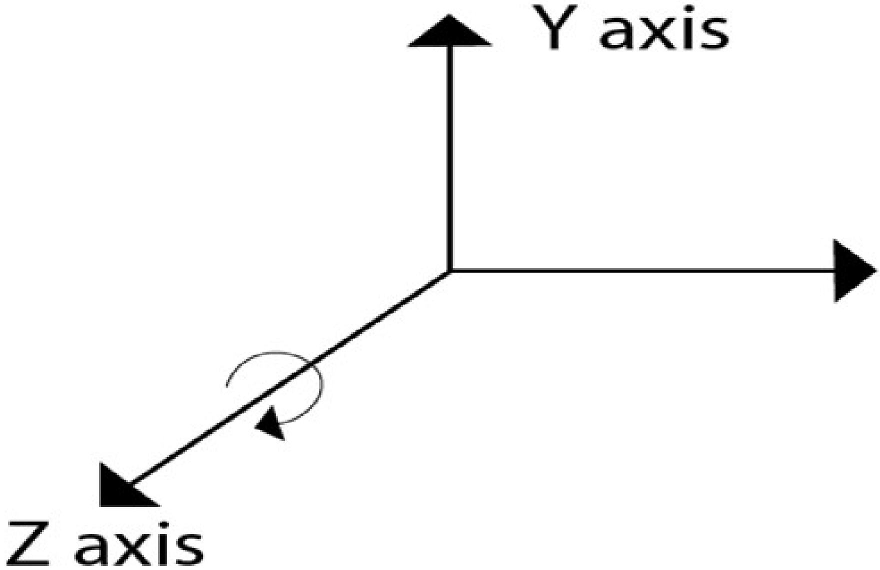
It is moving of an object about an angle. Movement can be anticlockwise or clockwise. 3D rotation is complex as compared to the 2D rotation. For 2D we describe the angle of rotation, but for a 3D angle of rotation and axis of rotation are required. The axis can be either x or y or z.

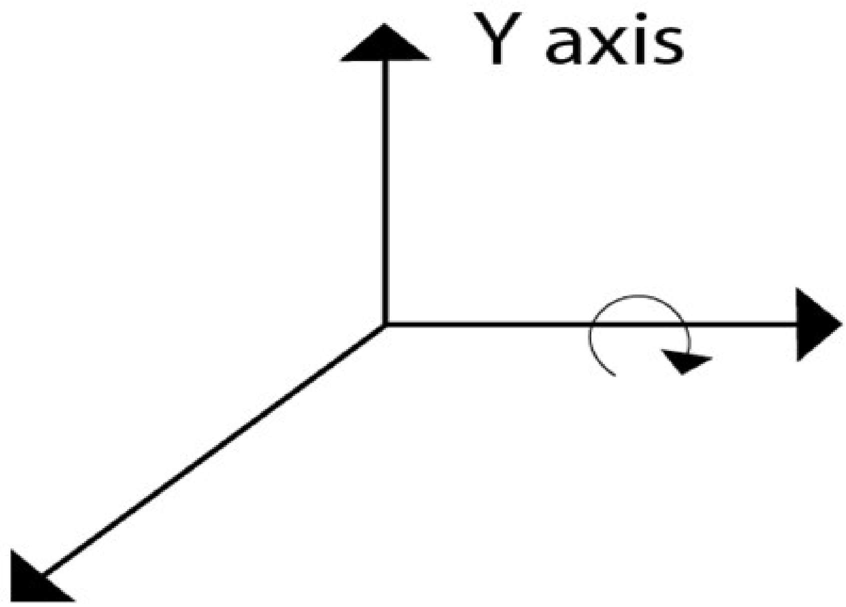
**Following figures shows rotation about x, y, z- axis**



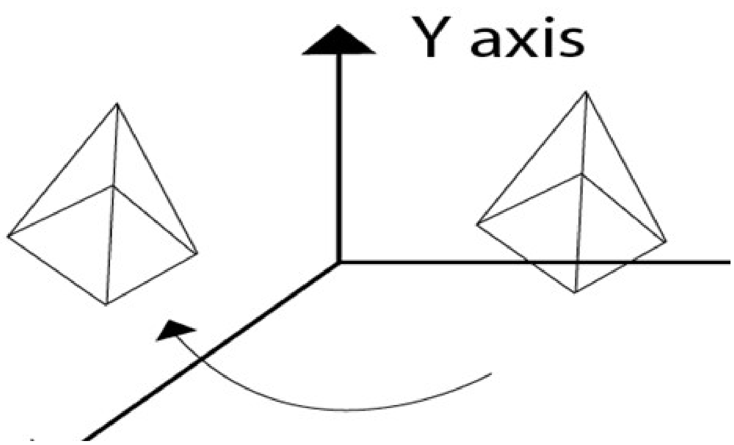




Y axis
Z axis
X a

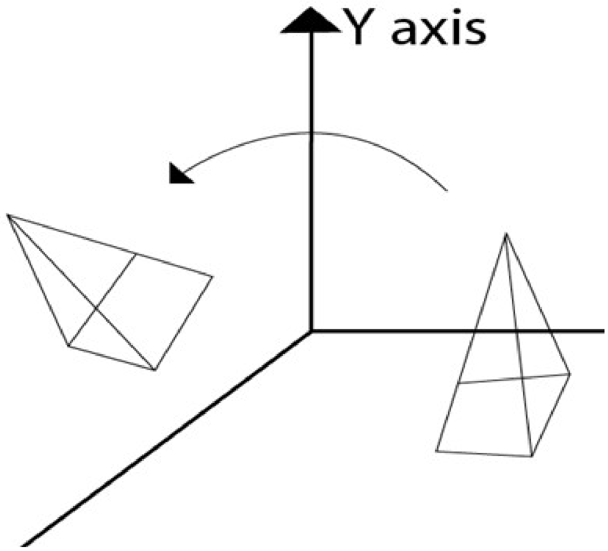

Y axis
Z axis




Following figure show rotation of the object about the Y axis




Following figure show rotation of the object about the Z axis




**UNIT- VI**

**Illumination Model:**

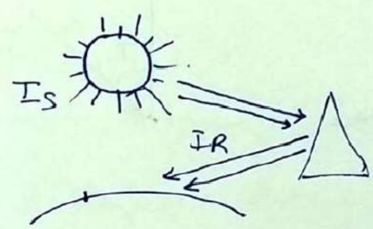
# Illumination Model

(1)

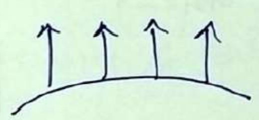
Keywords: Diffuse illumination, Diffuse Reflection. Cofficient of Reflection, Lambert's Law.

- An Illumination model is also called as lighting model, is used to calculate the intensity of light that we should see at a given point of time. on the surface of an object.

* Diffuse illumination (Ambient light)




* Diffuse Reflection



Intensity is equal at every point irrespective of view direction

$[I = I_S K_d N.S]$

* Cofficient of Reflection

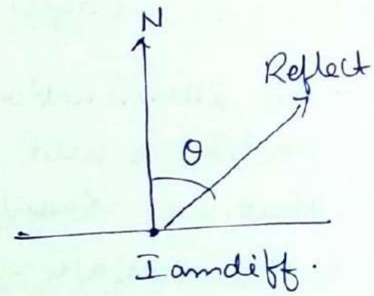
$R = \frac{I_R}{I_S}$  R = 0 to 1

0: Max absorb

1: Max Reflect.

## Lambert's Law

$I_{adiff} = K_a I_a$



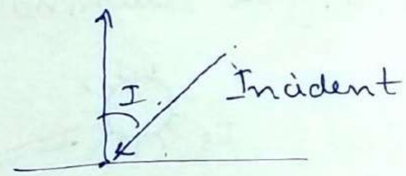

where

$K_a$ = cofficient or ambient Reflection cofficient (material)

$I_a$ : Intensity of ambient.

for some point source

$I_p = K_d I_d \cos I$



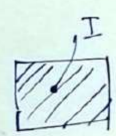

$I_{diff} = K_a I_a + K_d I_d \cos I$

---

Note:

Illumination : How to color a single point

shading : How to color whole object.




Three Types of shadings

1) Flat
2) Gouraud
3) Phong.

# Gouraud Shading
## (Intensity Interpolation Method)

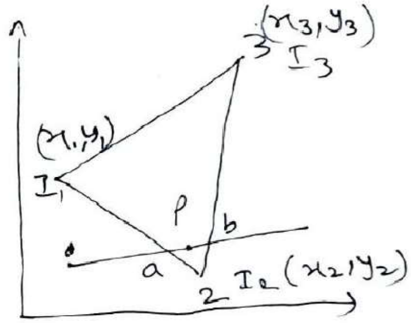
(1) Determine the Average unit vector at each polygon vector

(2) Apply Iu Model to each polygon vector to determine polygon vertex intensity.

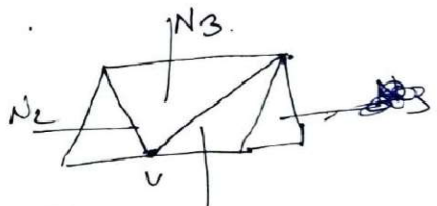
(3) Linearly interpolate the vertex intensities over the surface of polygon.

1) $N_v = \dfrac{\sum_{i=1}^{n} N_i}{\left|\sum_{i=1}^{n} N_i\right|}$ (magnitude)

$n$ = no of surfaces sharing that vertex.




2) By applying Illumination we get intensity of each point.




3) Intensity of $I_a = \dfrac{y_a - y_2}{y_1 - y_2} I_1 + \dfrac{y_1 - y_a}{y_1 - y_2} I_2$

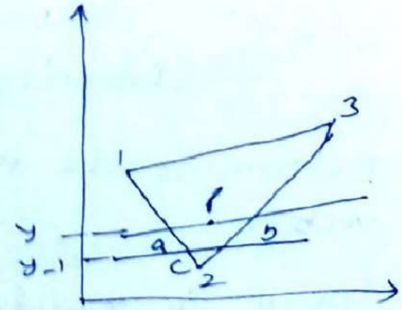
Intensity of $I_b = \dfrac{y_b - y_3}{y_2 - y_3} I_3 + \dfrac{y_3 - y_b}{y_2 - y_3} I_2$

for point $p = I_p = \dfrac{x_b - x_p}{x_b - x_a} I_a + \dfrac{x_p - x_a}{x_b - x_a} I_b$

$$I_c = I_a + \left[\frac{I_2 - I_1}{y_1 - y_2}\right]$$

where

$\left[\frac{I_2 - I_1}{y_1 - y_2}\right]$ will be constant for line (1, 2)




Advantages.

- Remove discontinuity of flat surface

Disadvantages.

- Mach - band (problem of streak line)
- one can miss specular Reflection (Highlights)

# PHONG Shading

(5)

(Normal vector Interpolation shading)

Steps of phong shading

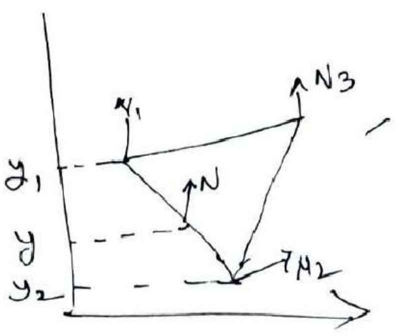
1) Determine the Average unit Normal vector at each. polygon ~~vertex~~. vertex

2) Linearly interpolate the vector normals over the surface of polygon

3) Apply the illumination model along each scan to determine projected pixel intensity of surface point.

(1) $$\frac{\sum_{i=1}^{n} N_i}{\left|\sum_{i=1}^{n} N_i\right|}$$ where n = no of surface shared by vertex.

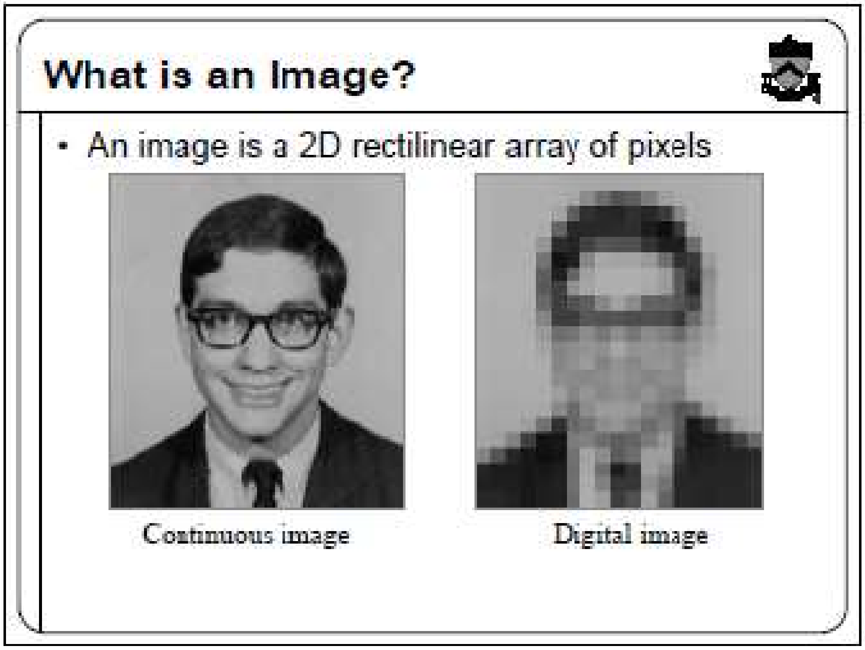
(2)

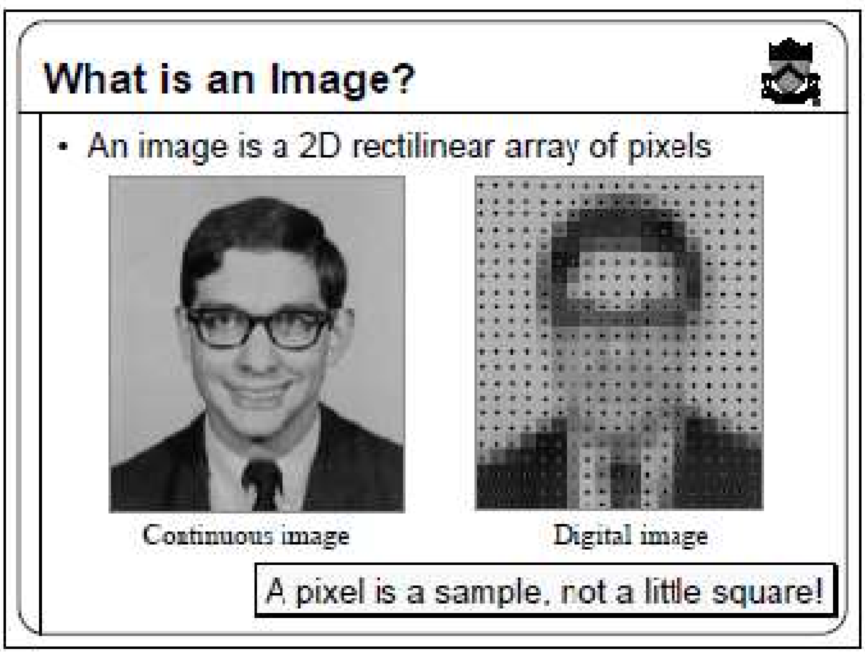



$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

(3) By applying Illumination we get intensity of each point.

Halftone and Dither



## What is an Image?

- An image is a 2D rectilinear array of pixels

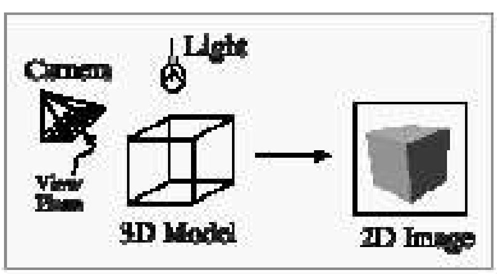
Continuous image

Digital image

A pixel is a sample, not a little square!

## Image Acquisition

- Pixels are samples from continuous function
  - Photoreceptors in eye
  - CCD cells in digital camera
  - Rays in virtual camera



## Image Resolution

- Intensity resolution
  - Each pixel has only "Depth" bits for colors/intensities
- Spatial resolution
  - Image has only "Width" x "Height" pixels
- Temporal resolution
  - Monitor refreshes images at only "Rate" Hz

Typical Resolutions

| | Width x Height | Depth | Rate |
|---|---|---|---|
| NTSC | 640 x 480 | 8 | 30 |
| Workstation | 1280 x 1024 | 24 | 75 |
| Film | 3000 x 2000 | 12 | 24 |
| Laser Printer | 6600 x 5100 | 1 | - |

## Sources of Error

- Intensity quantization
  - Not enough intensity resolution
- Spatial aliasing
  - Not enough spatial resolution
- Temporal aliasing
  - Not enough temporal resolution

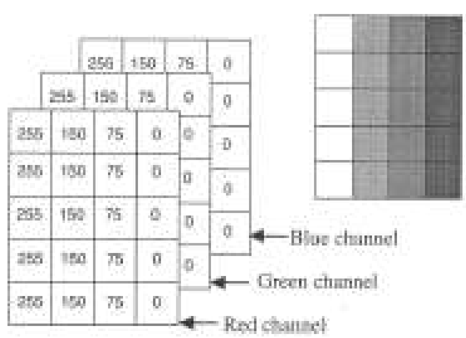
$$E^2 = \sum_{(x,y)} \left(I(x,y) - P(x,y)\right)^2$$

# Overview

- Image representation
  - What is an image?
- Quantization
  - Errors due to limited intensity resolution
- Halftoning and Dithering
  - Reduce effect of quantization errors

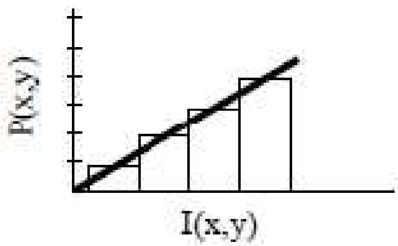
# Quantization

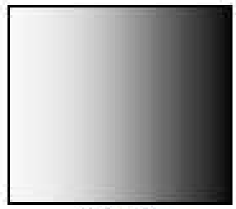
- Artifacts due to limited intensity resolution
  - Frame buffers have limited number of bits per pixel
  - Physical devices have limited dynamic range



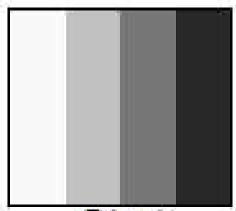
## Uniform Quantization

$$P(x, y) = \text{trunc}(I(x, y) + 0.5)$$



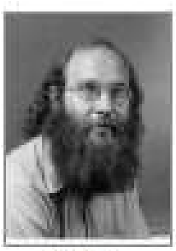
I(x,y)



P(x,y)
(4 bits per pixel)

## Uniform Quantization

- Images with decreasing bits per pixel:



8 bits

4 bits

2 bits

1 bit

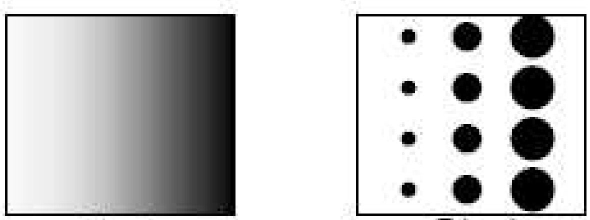
Notice contouring

## Overview

- Image representation
  - What is an image?
- Quantization
  - Errors due to limited intensity resolution
- » Halftoning and Dithering
  - Reduce effects of quantization

## Classical Halftoning

- Use dots of varying size to represent intensities
  - Area of dots proportional to intensity in image



I(x,y) P(x,y)

## Classical Halftoning

Newspaper Image

From New York Times, 9/21/99

## Halftone patterns

- Use cluster of pixels to represent intensity
  - Trade spatial resolution for intensity resolution

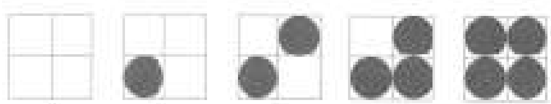



Figure 14.37 from H&D

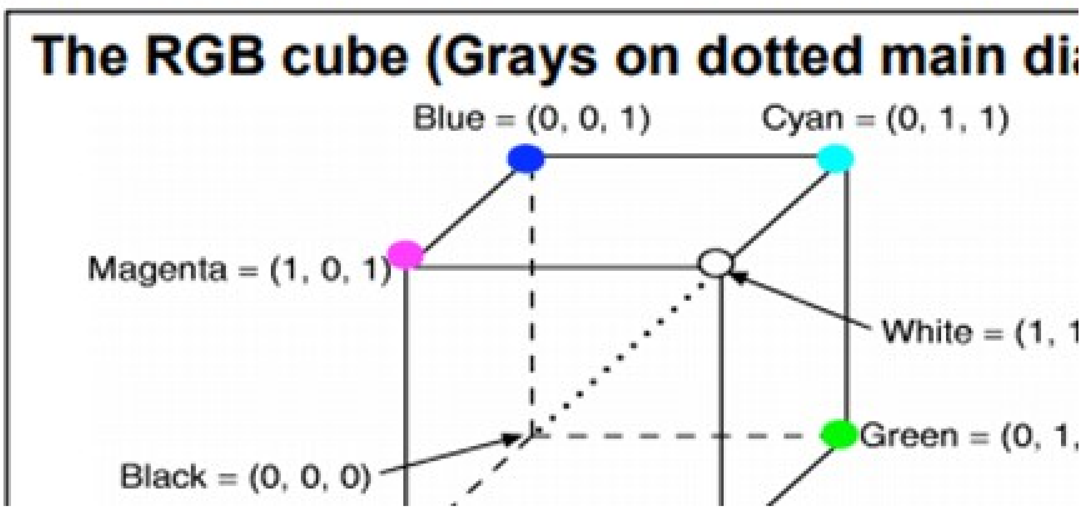
Color Models

Color spaces are the mathematical representation of a set of colors. There are many color models. Some of them are RGB, CMYK, YIQ, HSV, and HLS, etc. These color spaces are directly related to saturation and brightness. All of these color spaces can be derived using RGB information using devices such as cameras and scanners.

## RGB Color Space

**RGB** stands for **Red**, **Green**, and **Blue**. This color space is widely used in computer graphics. RGB are the main colors from which many colors can be made.

RGB can be represented in the 3-dimensional form:

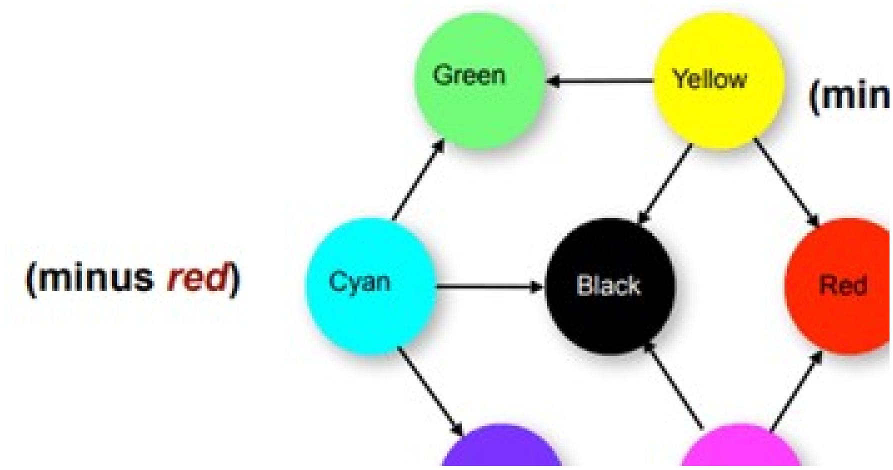



Below table is 100% RGB color bar contains values for 100% amplitude, 100% saturated, and for video test signal.

| | Nominal Range | White | Yellow | Cyan | Green | Magenta | Red |
|---|---|---|---|---|---|---|---|
| R | 0 to 255 | 255 | 255 | 0 | 0 | 255 | 255 |
| G | 0 to 255 | 255 | 255 | 255 | 255 | 0 | 0 |

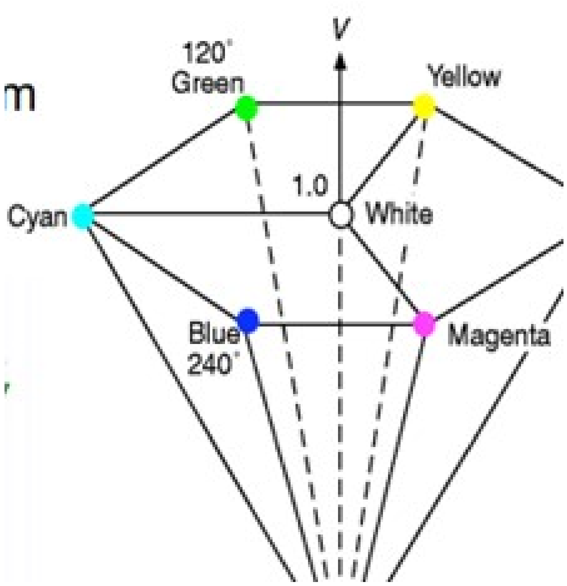
## CMYK Color Model

**CMYK** stands for **Cyan**, **Magenta**, **Yellow** and **Black**. CMYK color model is used in electrostatic and ink-jet plotters which deposits the pigmentation on paper. In these model, specified color is subtracted from the white light rather than adding blackness. It follows the Cartesian coordinate system and its subset is a unit cube.




## HSV Color Model

**HSV** stands for **Hue**, **Saturation**, and **Value (brightness)**. It is a hexcone subset of the cylindrical coordinate system. The human eye can see 128 different hues, 130 different saturations and number values between 16 (blue) and 23 (yellow).




## HLS Color Model

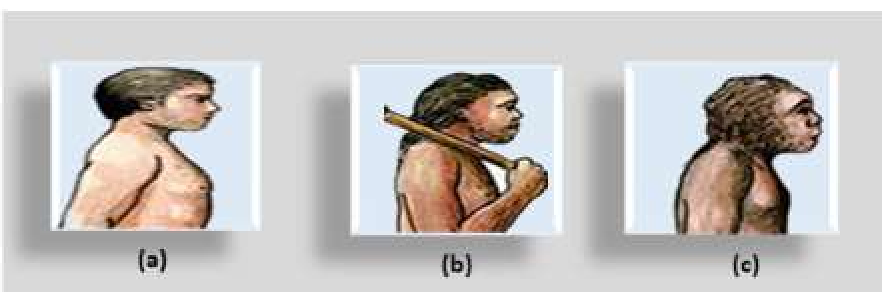
**HLS** stands for **Hue Light Saturation**. It is a double hexcone subset. The maximum saturation of hue is S= 1 and L= 0.5. It is conceptually easy for people who want to view white as a point.

## Animation Functions

**1. Morphing:** Morphing is an animation function which is used to transform object shape from one form to another is called Morphing. It is one of the most complicated transformations. This function is commonly used in movies, cartoons, advertisement, and computer games.

**For Example:**

**1.Human Face is converted into animal face as shown in fig:**




**2. Face of Young person is converted into aged person as shown in fig:**

**The process of Morphing involves three steps:**

1. In the first step, one initial image and other final image are added to morphing application as shown in fig: I$^{st}$ & 4$^{th}$ object consider as key frames.
2. The second step involves the selection of key points on both the images for a smooth transition between two images as shown in 2$^{nd}$ object.

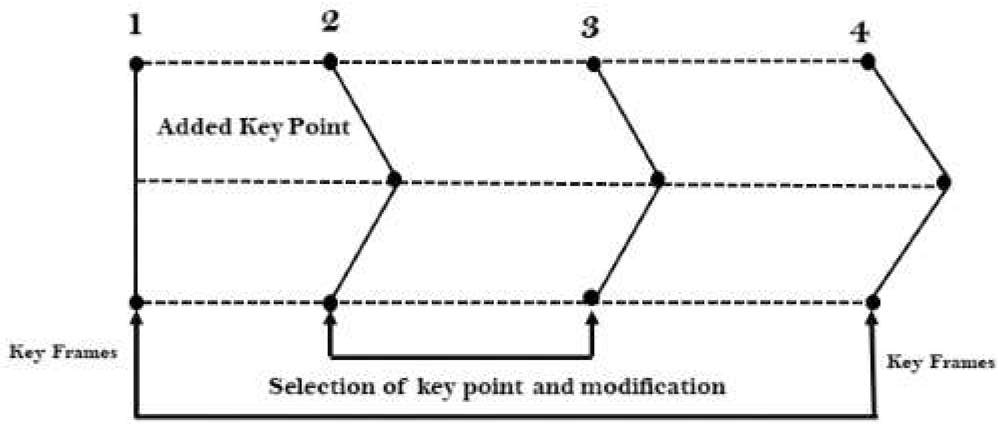



Fig: Process of Morphing

3. In the third step, the key point of the first image transforms to a corresponding key point of the second image as shown in 3$^{rd}$ object of the figure.

2. **Wrapping:** Wrapping function is similar to morphing function. It distorts only the initial images so that it matches with final images and no fade occurs in this function.

3. **Tweening:** Tweening is the short form of 'inbetweening.' Tweening is the process of generating intermediate frames between the initial & last final images. This function is popular in the film industry.

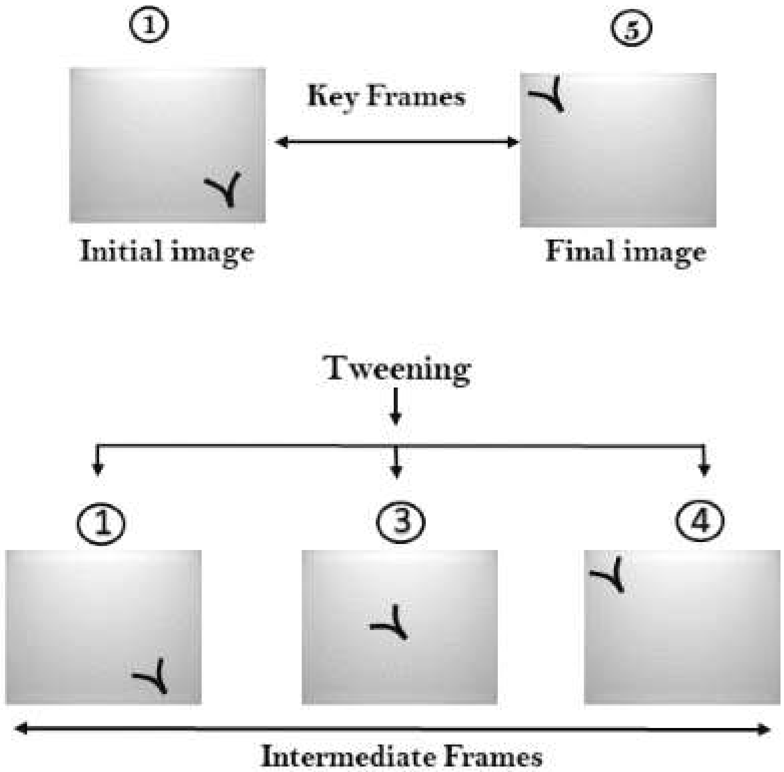



Fig: Tweening

4. **Panning:** Usually Panning refers to rotation of the camera in horizontal Plane. In computer graphics, Panning relates to the movement of fixed size window across the window object in a scene. In which direction the fixed sized window moves, the object appears to move in the opposite direction as shown in fig:


Fig: Panning

If the window moves in a backward direction, then the object appear to move in the forward direction and the window moves in forward direction then the object appear to move in a backward direction.

5. **Zooming:** In zooming, the window is fixed an object and change its size, the object also appear to change in size. When the window is made smaller about a fixed center, the object comes inside the window appear more enlarged. This feature is known as **Zooming In**.

When we increase the size of the window about the fixed center, the object comes inside the window appear small. This feature is known as **Zooming Out**.

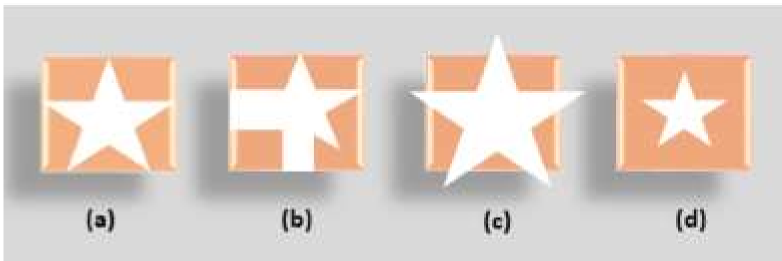



Fig: Zooming in & Zooming Out

6. **Fractals:** Fractal Function is used to generate a complex picture by using Iteration. Iteration means the repetition of a single formula again & again with slightly different value based on the previous iteration result. These results are displayed on the screen in the form of the display picture.

Types of animations

Vector vs Raster

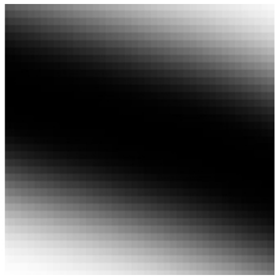
Simply put, vector and raster graphics are the two most common ways of handling digital images.

- Vector images are made up of mathematical formulas that express points and curves to create lines and shapes of single colors.

Vector graphics are excellent for the web, architectural design or anything that requires precision of line like technical drawings that can be easily transferred to a machine for moulds or 3d printing

They are also used heavily in 2d animation as a particular style that was initially popularized by Flash but has now become ubiquitous. Many animation programs now have the ability to work in vectors.

- Raster images are made up of individual pixels of separate colors which, when combined make up an image



Scanned photographs are a good example of raster images, as the computer has to break the real photo down into individual pixels of separate colors.

**Which one is better?**

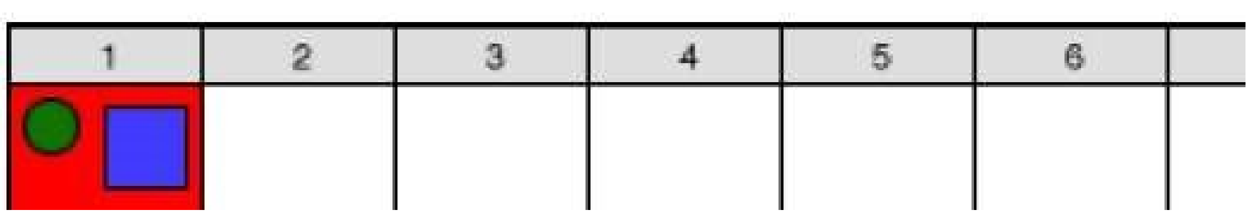
Like most things in life, it is about using the right tool for right job. There will be cases where vectors are better because the cleaner lines and ability to scale up to a higher resolution give the flexibility required, and other situations where raster or bitmaps are the way to go because they can handle realistic images, photographs and artistic effects much better.

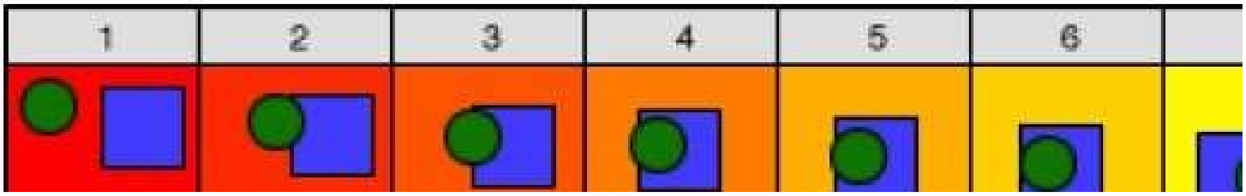
Key Frame Systems & Motion Specification

Key Framing

A key-frame is a frame where we define changes in animation. Every frame is a key-frame 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 key-frames.

Key-frames 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.






*Motion Specification*

There are several ways to specify the motion in any animation system.

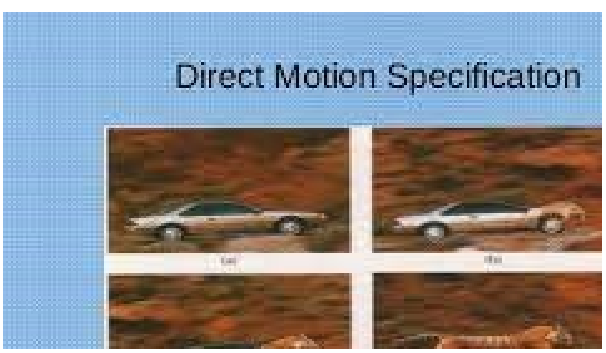
The most general methods are:

- Direct motion specification
- Goal directed specification
- Kinematics and Dynamics specification

**1. Direct Motion Specification:**

- Explicit parameters are provided
- Explicitly rotation angle are given object in any frame
- Explicitly translation vectors are given
- Geometric transformation are applied to transform coordinate position

Example: Ping-Pong ball game in which bouncing ball changes its potions and size.




**Advantages:**

- Easily and explicitly parameters are provided to any object.
- Coordinate positions are easily applied to transform the object

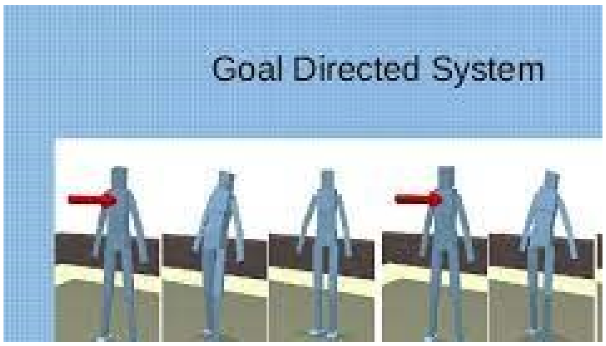
**Disadvantage:**

- Acceleration of any object is not possible.

**2. Goal directed system:**

- Provide general term specification of the motion.
- Abstractly describe the action expressing a quality or characteristics apart from any specific object.
- These are referred as goals directed because they provide specific motion of parameter.

Example: Dancing and running.




**3. Kinematics and Dynamics:**

- **Kinematics:** Motions parameter such as position, velocity and acceleration are specified without reference to the forces.
- **Dynamic**:
    - The forces that produce the velocities and accelerations are specified (Physically based modeling)
    - It uses laws such as Newton's law of motion.

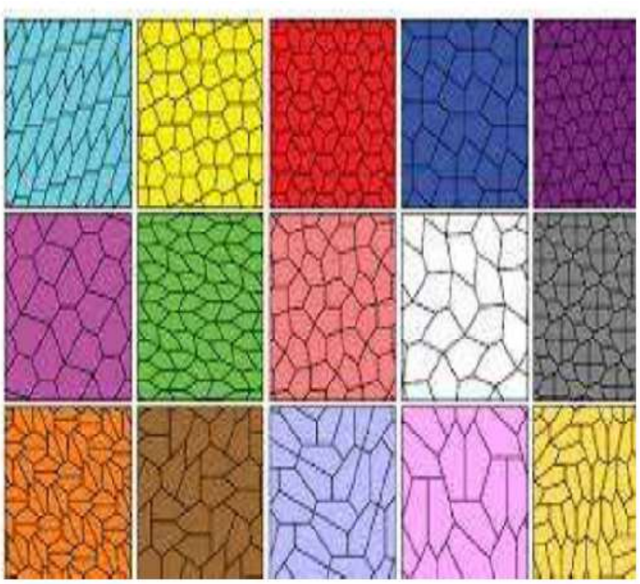
Computer Graphics and Realism

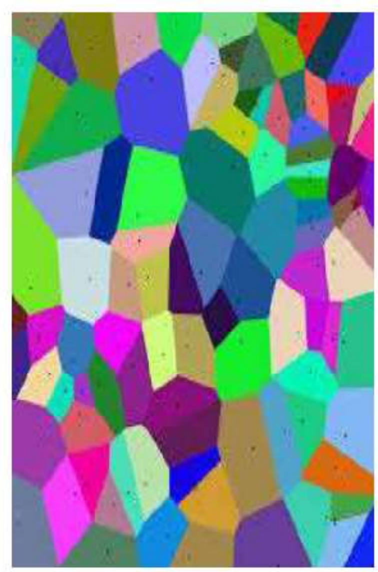
- Image is a visual representation of scene, it represent selected properties of scene to viewer with varying degree of realism.

# Tiling the Plane

- Use one or more geometric shapes
- Tessellation(without gaps) of flat surface
- Shape repeated
- Moving infinity
- Covering entire plane
- Used arts,mosaics,wall papers,tiled floor

# Tiling the Plane





# Tiling the Plane

# Types of tiling

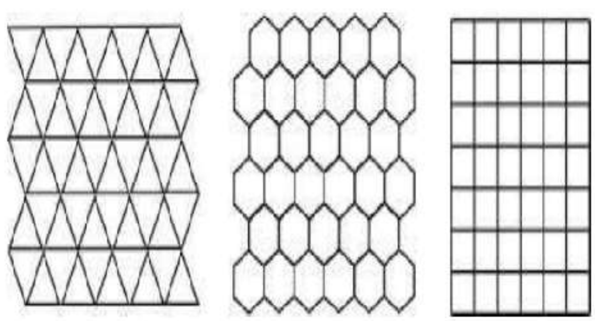
- Monohedral tiling
- Dihedral tiling
- Drawing tiling
- Reptiles

# Monohedral tiling

- Based on single polygon
- **Types**

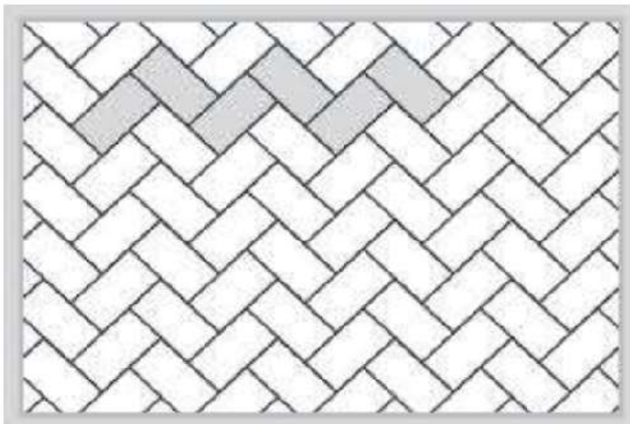
1. Regular tiling
2. Patterns
3. Cario tiling
4. Polymino
5. Polyiamond

# Regular Tiling



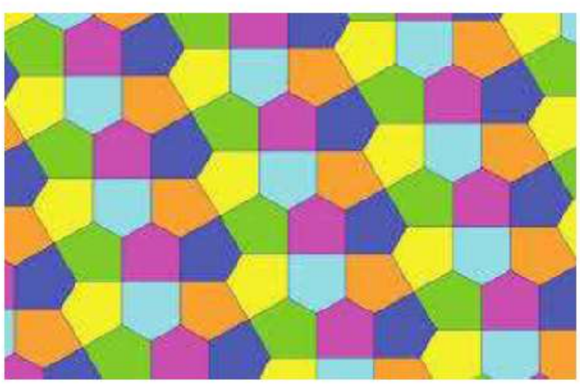
# Patterns

- Shifting the tessellation in particular direction

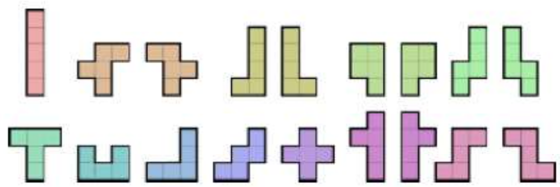


# Cairo tiling

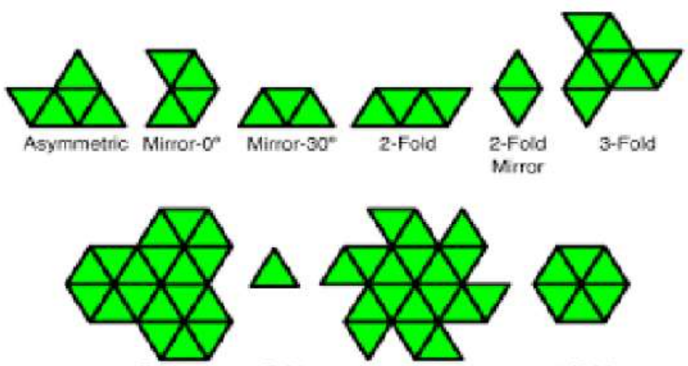
- Four pentagon fit together to form hexagon
- Used to tile the plane
- Many street in cairo,Egypt in this pattern



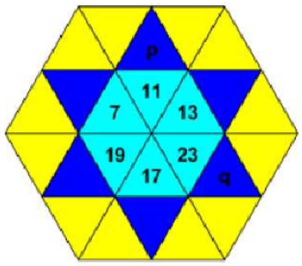
# Polymino

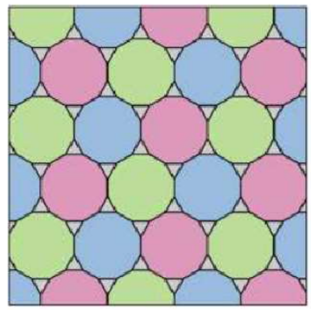


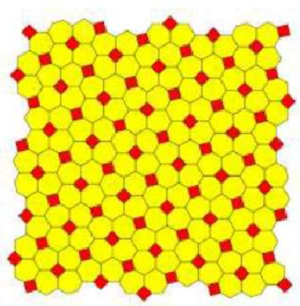
# Polyaimonds



# Dihedral Tiling

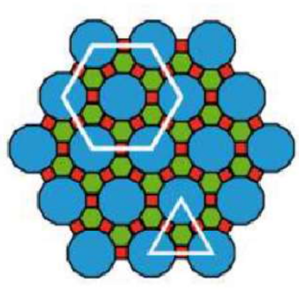




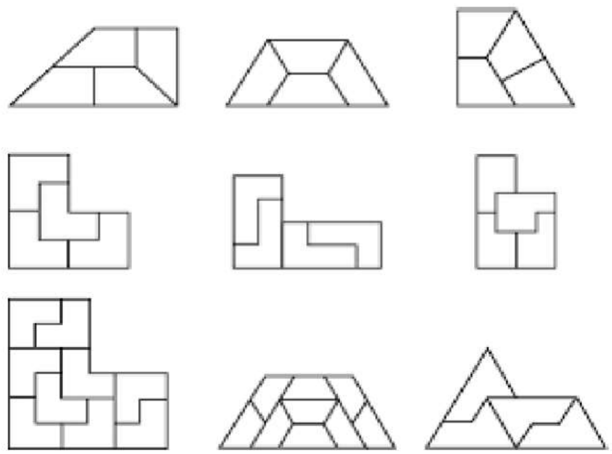
# Drawing Tiling

- Large window setup
- Tiles grouped together into single figure
- Single figure drawn again and again
- Non periodic figure include
- Small to large and large to small



# Reptiles

- Non periodic tiling
- Based on square, equilateral triangle



# Application of tiles


# Fractals

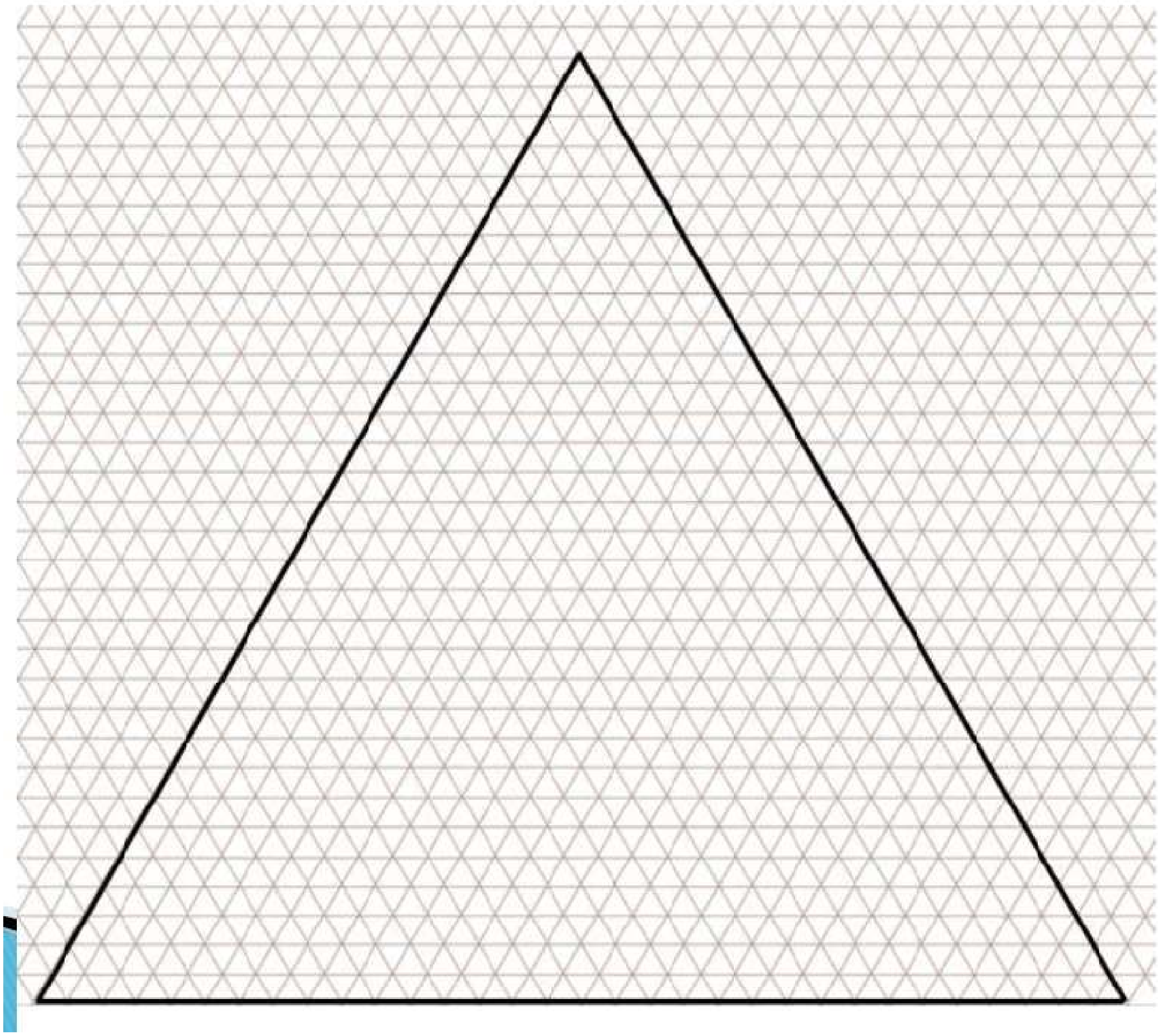
- A fractal is a never-ending pattern.
- Fractals are infinitely complex patterns that are self-similar across different scales.
- They are created by repeating a simple process over and over in an on-going feedback loop.

# Types of Fractals

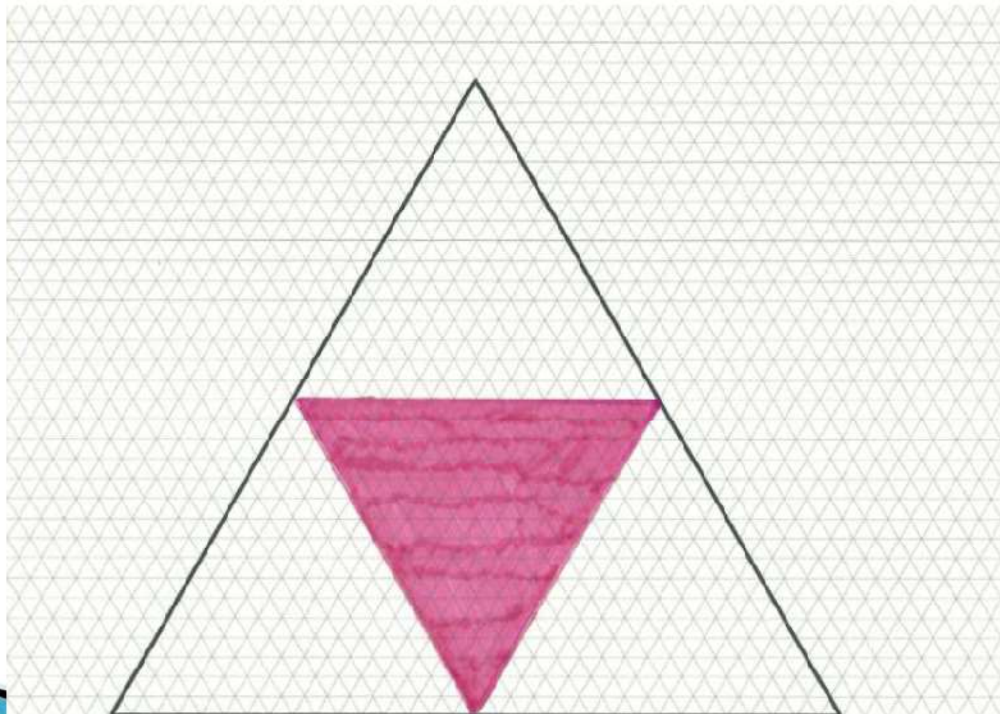
- Self Similar fractals
- Self Affine fractals
- Invariant fractals

# Self Similar fractals

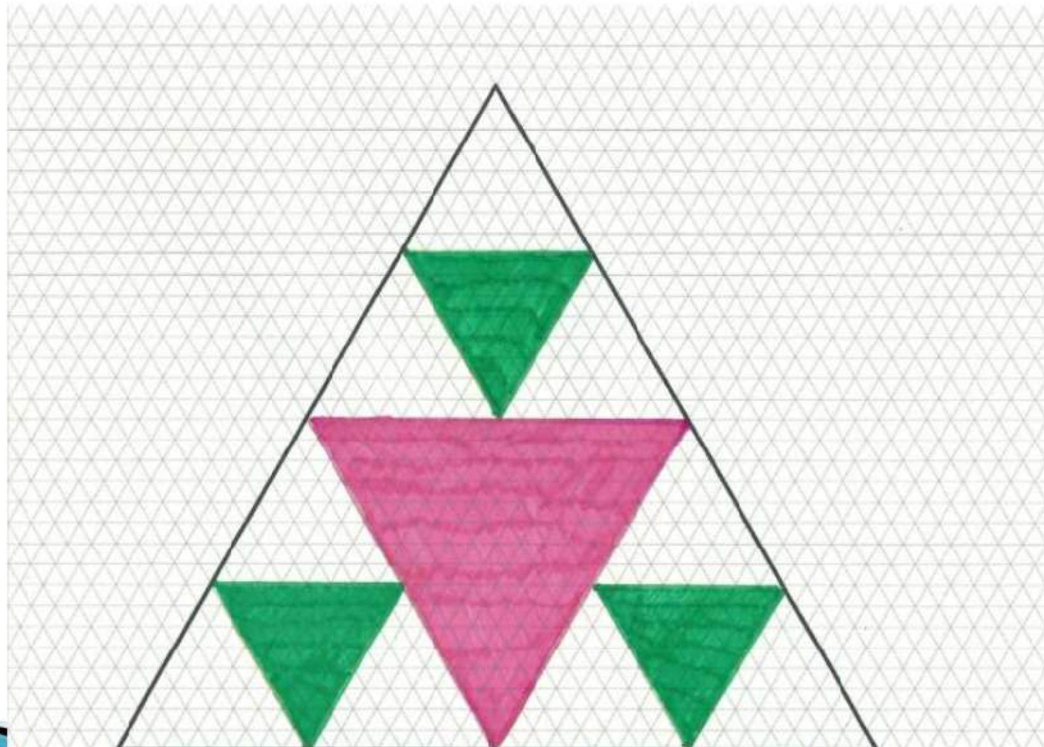
- Geometric figure is self similar
- Fractals appear identical at different scales



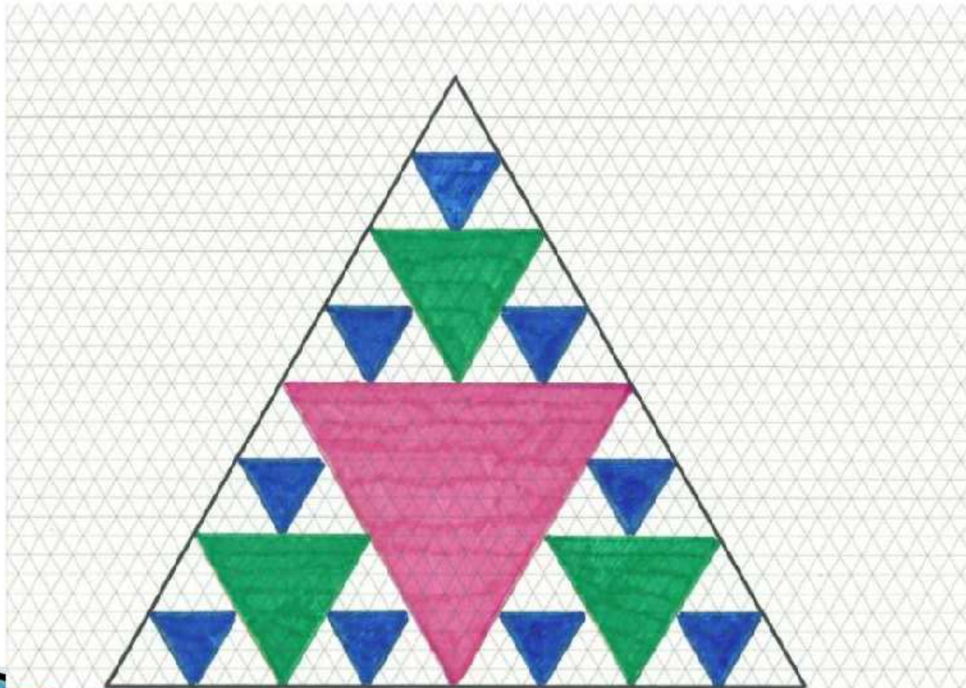
# Iteration 1



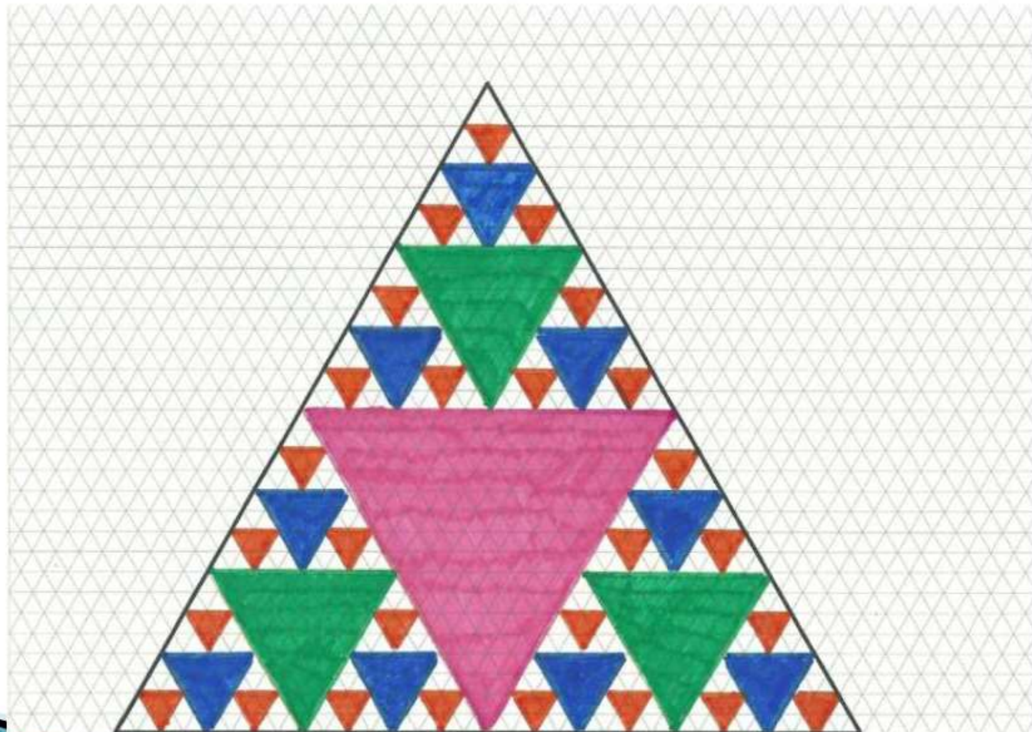
# Iteration 2



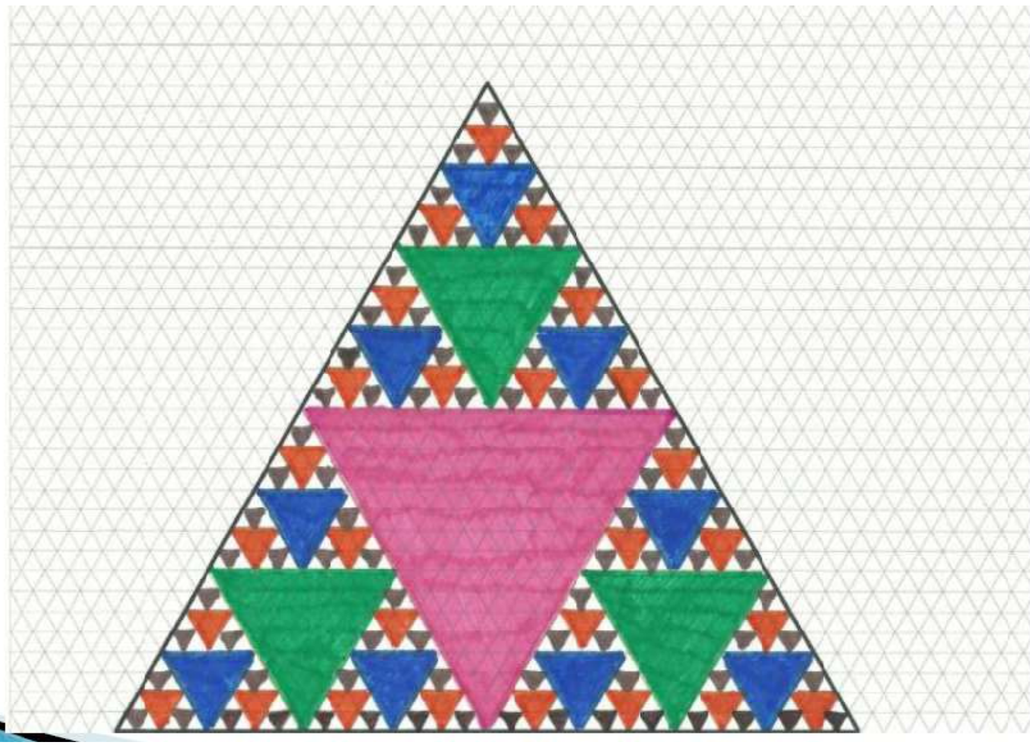
# Iteration 3



# Iteration 4

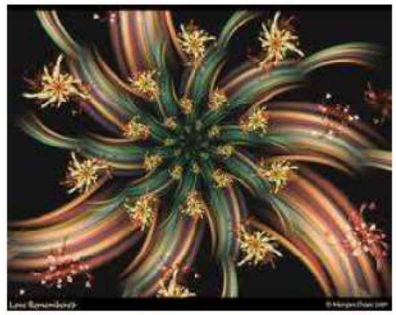


# Iteration 5



# Self similar fractal usage

- Model trees,shrubs,plants

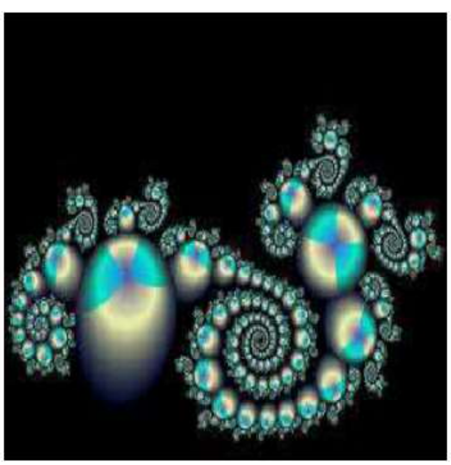


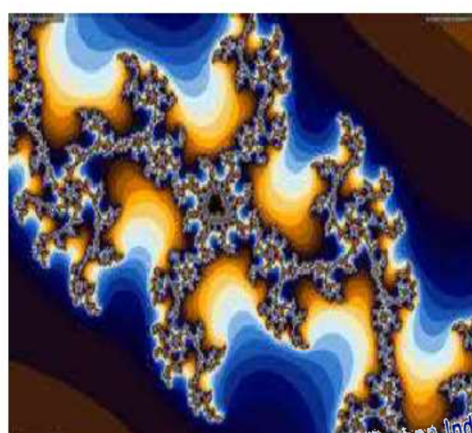
# Self Affine fractals

- Fractal appear approximately identical at different scales
- Model water,clouds,terrain

# Invariant fractals

- Non linear transformation





# Recursively defined curves

- Curves created by iterations
- Formulas repeated with slightly different values over and over again

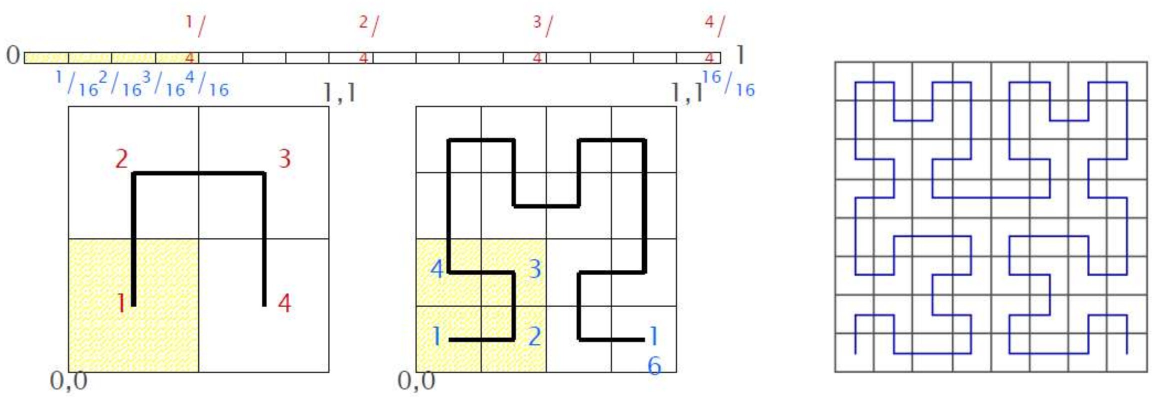
# Types

- Hilberts Curve
- Koch Curve
- Dragon Curve
- Space filling Curve/Piano Curve
- C Curve

# Hilberts Curve

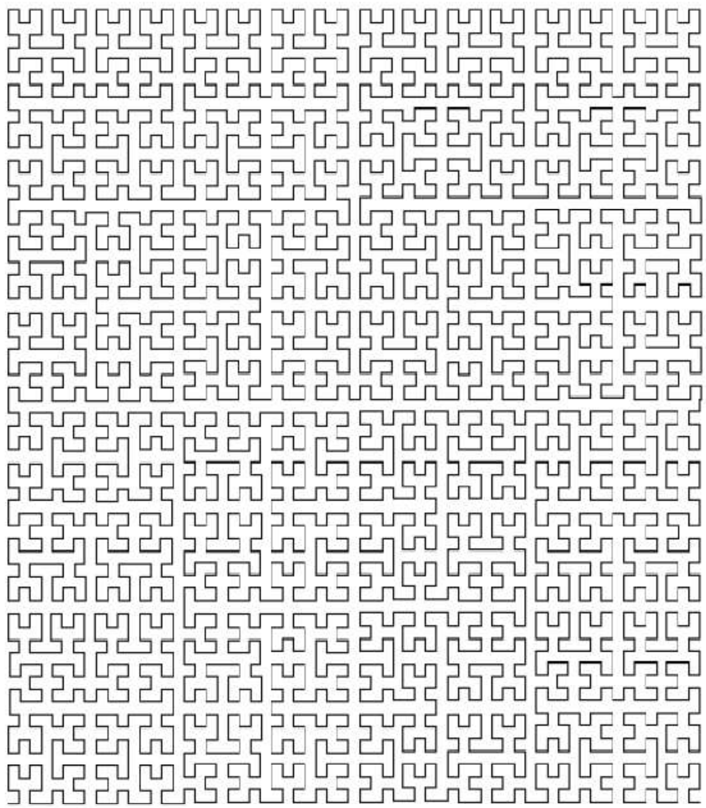
- It was described by the German mathematician David Hilbert in 1891.
- The Hilbert curve is a space filling curve.
- It visits every point in a square grid with a size of 2×2, 4×4, 8×8, 16×16, or any other power of 2.

## The Hilbert curve: geometric generation

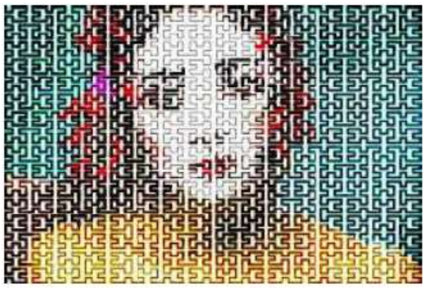



1st iteration        2nd iteration        3rd iteration

# The Hilbert curve: geometric generation

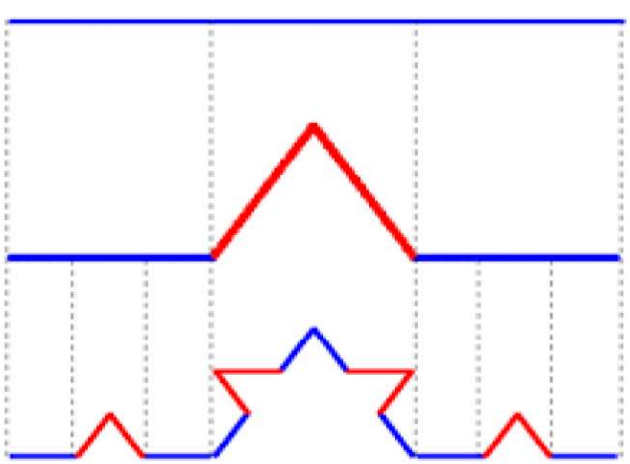


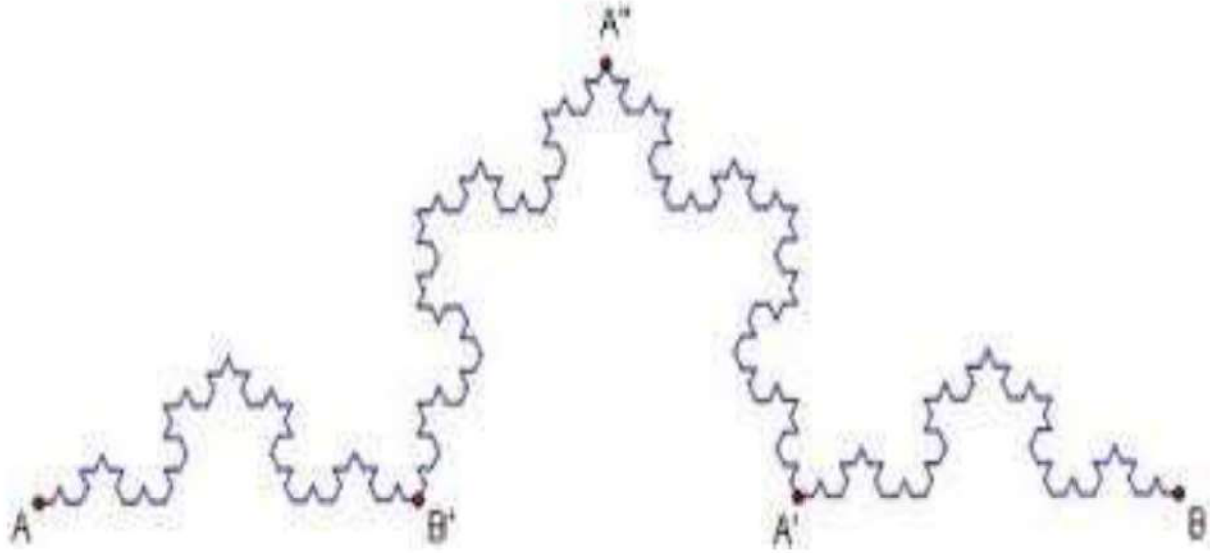
6th iteration



# Koch Curve

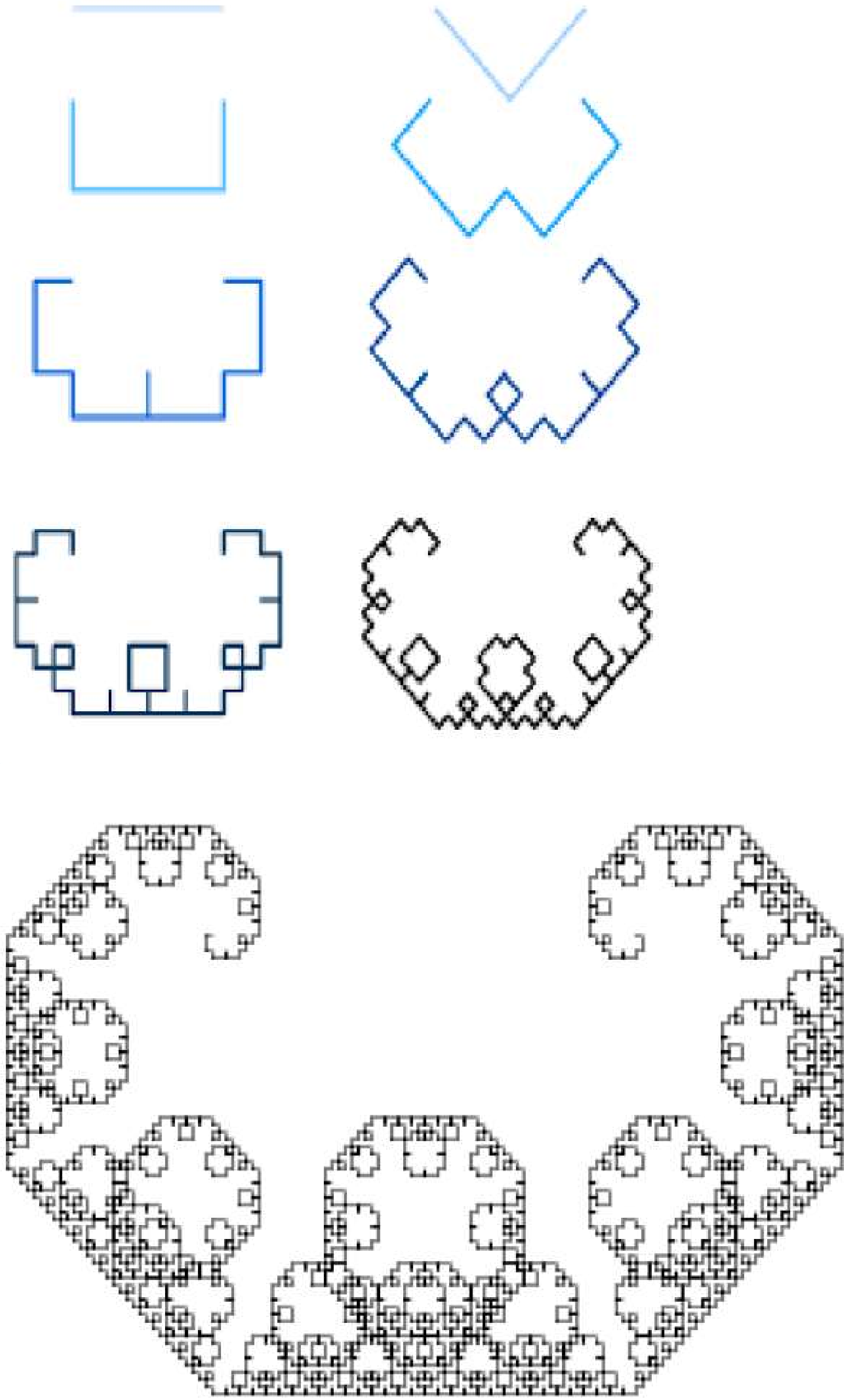
- Developed by Helga von Koch in 1904




A"
A
B'
A'
B

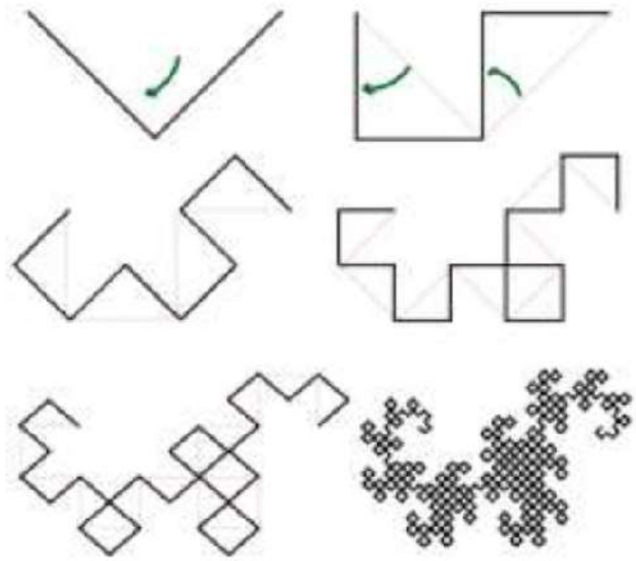
# C Curves

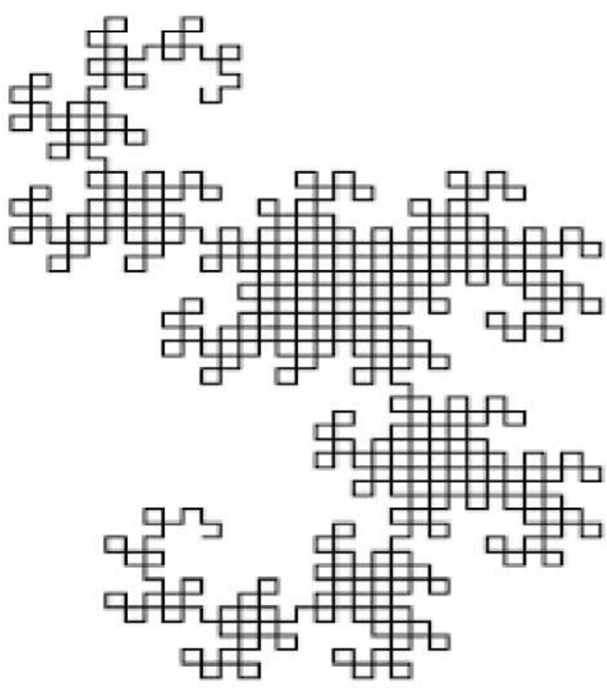
- Self similar fractals
- Described by Ernesto cesaro and Georg Faber in the year 1910



# Dragon Curves

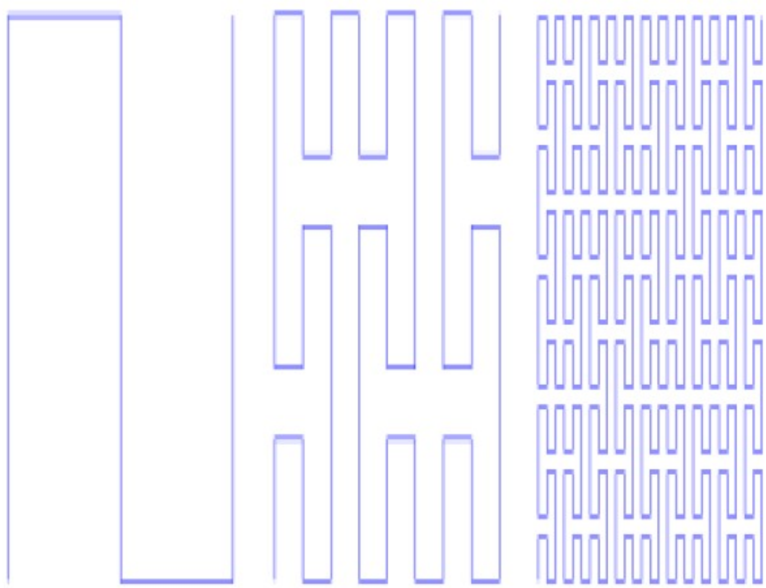
- Self similar fractal curves





# Space filling curve/Peano curve

- Developed by Italian mathematician Guiseppe peano in 1890
- Space filling curve



# Grammar based models

- Structure defined by language
- Languages described by a collection of productions
- example, A->AA creates results of A, AA, AAAA
- B->A[B] creates results of B, A[B], AA[B], etc.

**Mid Term Question Paper**

*Department of Computer Science and Engir*

**Techno India NJR Institute of Technology, U**

**B. Tech. (III Year, V Sem) (Make Up ) Examination**

***Subject:*- Computer Graphics and Multimedia (**

MM.50 Time

**Attempt any 5 questions (each question carries 10 marks)**

1. Explain the functionality of CRT Draw a neat and labeled diagram of CRT. **CO1**
2. Analyze the difference between midpoint ellipse and midpoint circle drawing algorithm. **CO2**
3. Explain 4-way and 8-way connecting method for filling primitives. Write down the algorithm of flood fill. **CO2**
4. Explain the properties of B Spline curve. **CO3**
5. Where the halftone patterns and dithering techniques are useful explain with an example. **CO4**

For Techno India NJR Institute of Technology
पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

## Previous Year Question Papers

Roll No. ____________ Total No of Pages: 3

6E6024

# 6E6024
## B. Tech. VI-Sem. (Main/Back) Exam., April/May-2016
## Computer Science
## 6CS4A Computer Graphics and Multimedia Techniques

**Time: 3 Hours** **Maximum Marks: 80**

**Min. Passing Marks (Main & Back): 26**

***Instructions to Candidates:-***

*Attempt any **five questions,** selecting **one question** from **each unit**. All Questions carry **equal** marks. Schematic diagrams must be shown wherever necessary. Any data you feel missing may suitably be assumed and stated clearly.*

*Units of quantities used/ calculated must be stated clearly.*

*Use of following supporting material is permitted during examination. (Mentioned in form No. 205)*

1. NIL 2. NIL

## UNIT-I

Q.1 (a) Explain the following terms in context of display devices:

(i) resolution [2]

(ii) flickering [2]

(iii) interlacing [2]

(iv) refreshing [2]

(b) Go through steps of Bresenham's line drawring algorithm for the line segment between end points (21, 12) to (29, 16). [8]

**OR**

Q.1 (a) Differentiate between Raster and random scan display devices. [6]

(b) Explain beam penetration method. [6]

(c) What is importance of 8 – way symmetry in scan conversion of circle? [4]

## UNIT-II

Q.2 (a) Derive composite transformation matrix of translation followed by reflection. [8]

(b) Describe Cohen – Sutherland line clipping algorithm. [8]

**OR**

Q.2 (a) Differentiate between boundary fill and flood fill techniques. [6]

(b) Provide an example of inverse transformation in homogeneous coordinate system. [6]

(c) Discuss issues related to polygon clipping. [4]

## UNIT-III

Q.3 (a) How is image space method different from object space method? [4]

(b) Discuss properties of Bezier curves. [8]

(c) What are the issues related to hidden surfaces? [4]

**OR**

Q.3 (a) Illustrate depth buffer method with diagrams. [8]

(b) Discuss properties of B-spline curves. [8]

## UNIT-IV

Q.4 (a) Discuss following color models -

(i) RGB [4]

(ii) YIQ [4]

(iii) CMY [4]

(b) Describe Phong shading. [4]

## OR

Q.4 (a) What are the various aspects of illumination of objects? [8]

(b) Describe Gourand shading. [4]

(c) What is HSV color model? [4]

## UNIT-V

Q.5 Write short notes on any two - [8×2=16]

(a) Multimedia components

(b) Steps of animation

(c) Animation techniques

(d) Multimedia techniques

----------------------------------

6E6024

Roll No. ____________________ [Total No. of Pages : 2

# 6E6024

## B.Tech. VI Semester (Main/Back) Examination, April/May - 2017
## Computer Sc. & Engg.
## 6CS4A Computer Graphics and Multimedia Techniques



**Time : 3 Hours** **Maximum Marks : 80**
**Min. Passing Marks : 26**

***Instructions to Candidates:***

*Attempt any **five questions**, selecting **one question** from **each unit**. All questions carry **equal** marks. Schematic diagrams must be shown wherever necessary. Any data you feel missing suitable be assumed and stated clearly. Units of quantities used/calculated must be stated clearly.*

### Unit-I

**1.** a) Explain various application areas of computer graphics. Differentiate beam penetration method of colored CRT with shadow mask method. **(4+4=8)**

b) What steps are required to plot a line whose slope in between 0° and 45° using Bresenham's method? Indicate the raster locations would be chosen by Bresenham's algorithm when scan converting a line from screen coordinate (20,10) to (30,18). **(4+4=8)**

**(OR)**

**1.** a) If a TV screen has 525 scan lines and an aspect ratio of 3:4 and if each pixel contains 12 bits of intensity information, how many bits are required for refresh rate 30 frames per second? **(8)**

b) Give the advantages and disadvantages of DDA line algorithm. Explain mid point circle algorithm. **(2+6=8)**

### Unit-II

**2.** a) Show rotation of a 2D Box represented by (5,5) to (10,15) with respect to (5,5) by 90° in anticlockwise direction. **(8)**

b) Explain flood fill algorithm. Differentiate it with Boundary fill algorithm. **(5+3=8)**

**(OR)**

**2.** a) Explain Cohen Sutherland line algorithm. **(8)**

b) Show that the composition of two rotations is additive by concatenating the matrix representation for $R(\theta_1)$, and $R(\theta_2)$ to obtain : **(8)**

$R(\theta_1).\ R(\theta_2) = R(\theta_1+\theta_2)$

## Unit-III

3. a) Explain the scan line method for displaying the visible surface of a given polyhedron. **(8)**

b) Differentiate B-splines with Bezier curves. Briefly describe B-spline curve. **(3+5=8)**

**(OR)**

3. a) What is hidden surface problem? Write and explain Z-buffer algorithm for visible surface detection. **(2+6=8)**

b) What is parametric representation of a curve? Explain Bezier curve in detail. **(2+6=8)**

## Unit-IV

4. a) Explain following terms : **(3×3=9)**

   i) Diffuse reflection

   ii) Specular reflection

   iii) Illumination model

b) Explain phong shading. Compare it with Gouraud shading. **(4+3=7)**

**(OR)**

4. a) What is Ray Tracing? Explain Basic ray tracing algorithm. **(2+6=8)**

b) Explain color model RGB. Compare it with HSV. **(5+3=8)**

## Unit-V

5. a) Define Animation. Explain principles of animation briefly. **(2+6=8)**

b) What is compression of data? Explain MPEG in detail. **(2+6=8)**

**(OR)**

5. a) Explain various presentation tools. **(8)**

b) Explain Authority tools with their uses. **(8)**