SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIT 1 - FUNDAMENTALS OF COMPUTER GRAPHICS - SMR1402

UNIT - I

FUNDAMENTALS OF COMPUTER GRAPHICS

1. PRE-REQUISITE DISCUSSION

1.1. PRODUCT LIFE CYCLE (PLC)

Every product goes through a cycle from birth, followed by an initial growth stage, a relatively stable matured period, and finally into a declining stage that eventually ends in the death of the product as shown schematically in *Figure*.

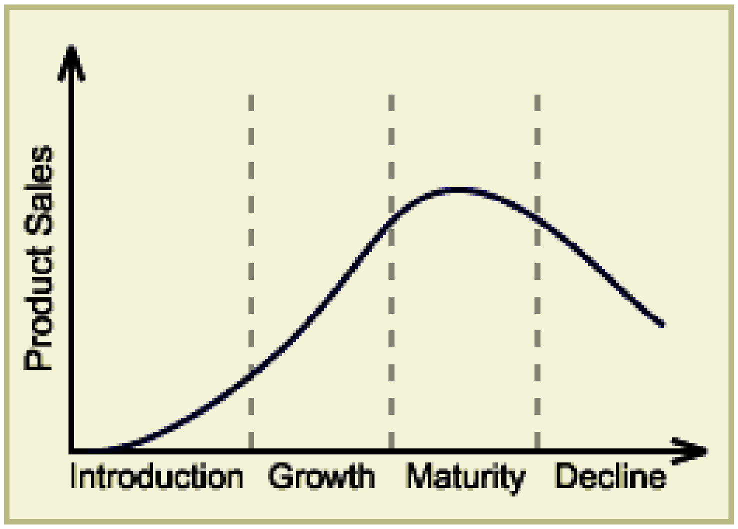


Figure.1.1. Product Life Cycle

(1) *Introduction stage*: In this stage the product is new and the customer acceptance is low and hence the sales are low.

(2) *Growth stage*: Knowledge of the product and its capabilities reaches to a growing number of customers.

(3) *Maturity stage*: The product is widely acceptable and sales are now stable, and it grows with the same rate as the economy as a whole grows.

(4) *Decline stage*: At some point of time the product enters the decline stage. Its sales start decreasing because of a new and a better product has entered the market to fulfill the same customer requirements.

1.2. PRODUCT LIFE CYCLE (PLC) FOR CONTINUOUS IMPROVEMENT

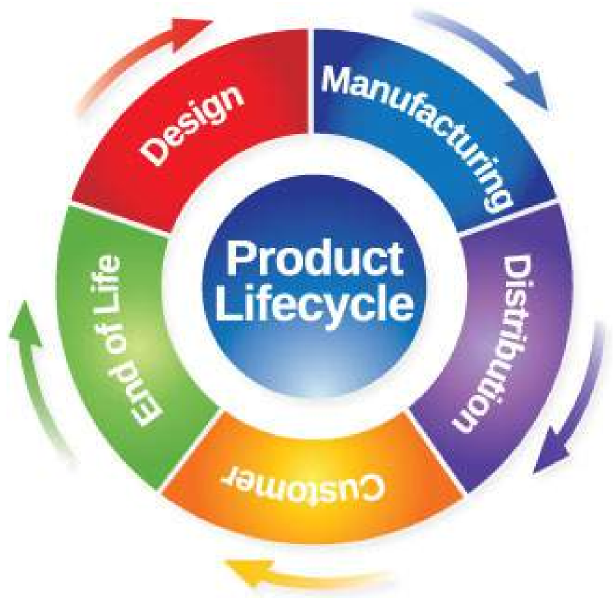


Figure.1.2. Product Life Cycle for continuous Improvement (Basic)

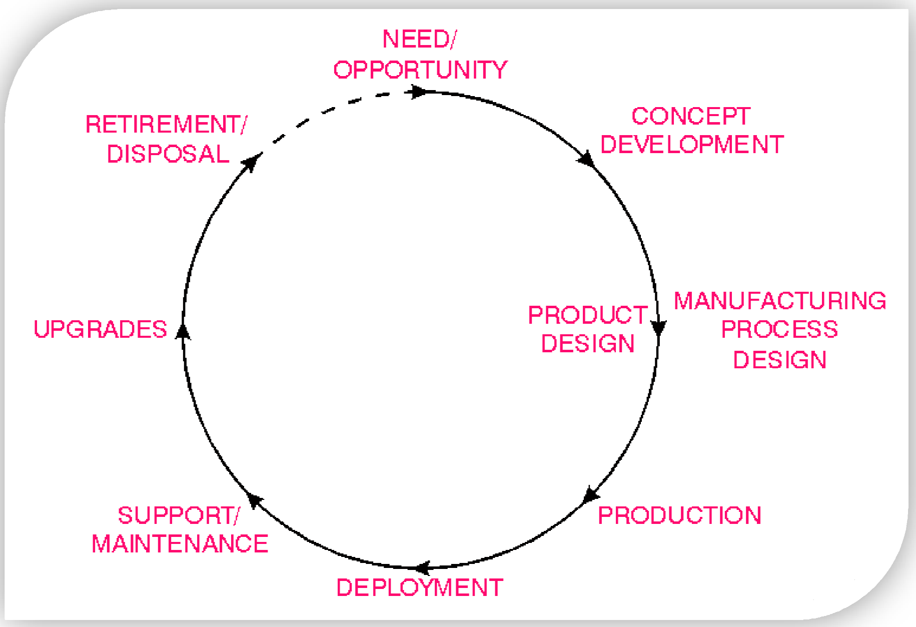


Figure.1.3. Product Life Cycle for continuous Improvement (Detailed)

1.3. TECHNOLOGY DEVELOPMENT CYCLE

- The development of a new technology follows a typical S-shaped curve. In its early stage, the progress is limited by the lack of ideas. A single good idea can make several other god ideas possible, and the rate of progress is exponential. Gradually the growth becomes linear when the fundamental ideas are in place and the progress is concerned with filling the gaps between, the key ideas.

- It is during this time when the commercial exploitation flourishes. But with time the technology begins to run dry and increased improvements come with greater difficulty. This matured technology grows slowly and approaches a limit asymptotically.

- The success of a technology based company lies in its capabilities of recognizing when the core technology on which the company's products are based begin to mature and through an active R&D program, transfer to another technology growth curve which offers greater possibilities.

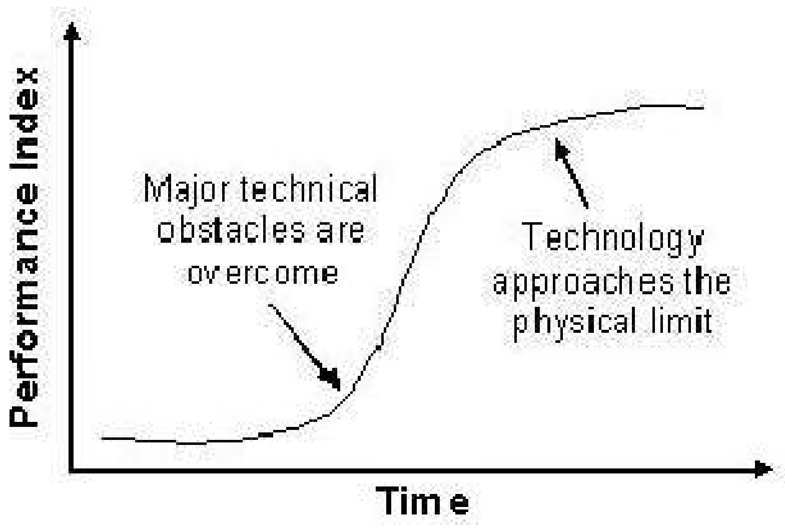

Performance Index
Major technical obstacles are overcome
Technology approaches the physical limit
Time

Figure.1.4. Schematic outline of Technology Development Curve

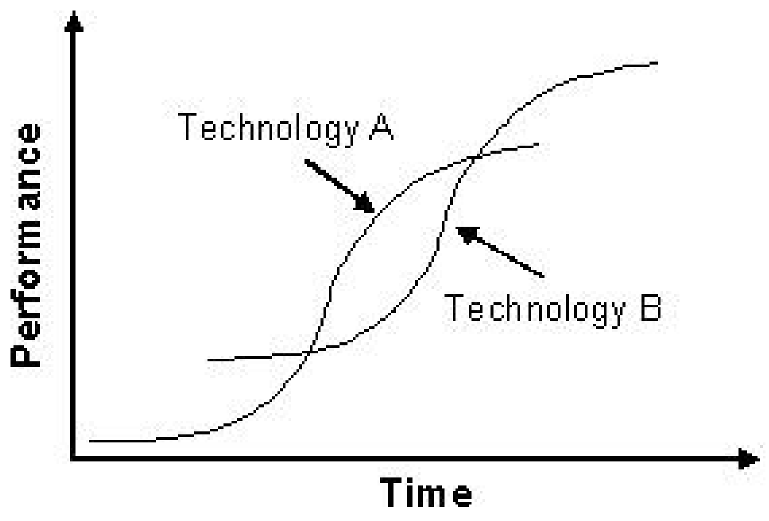


Figure.1.5. Improved program to develop new technology before the complete extinct of existing technology

1.4. THE DESIGN PROCESS - INTRODUCTION

- The **Engineering Design Process** is the formulation of a plan to help an engineer build a product with a specified performance goal. This process involves a number of steps, and parts of the process may need to be repeated many times before production of a final product can begin.

- It is a decision making process (often iterative) in which the basic sciences, mathematics, and engineering sciences are applied to convert resources optimally to meet a stated objective. Among the fundamental elements of the design process are the establishment of objectives and criteria, synthesis, analysis, construction, testing and evaluation.

- The Engineering Design process is a multi-step process including the research, conceptualization, feasibility assessment, establishing design requirements, preliminary design, detailed design, production planning and tool design, and finally production.

1.4.1. Steps involved in Engineering Design process

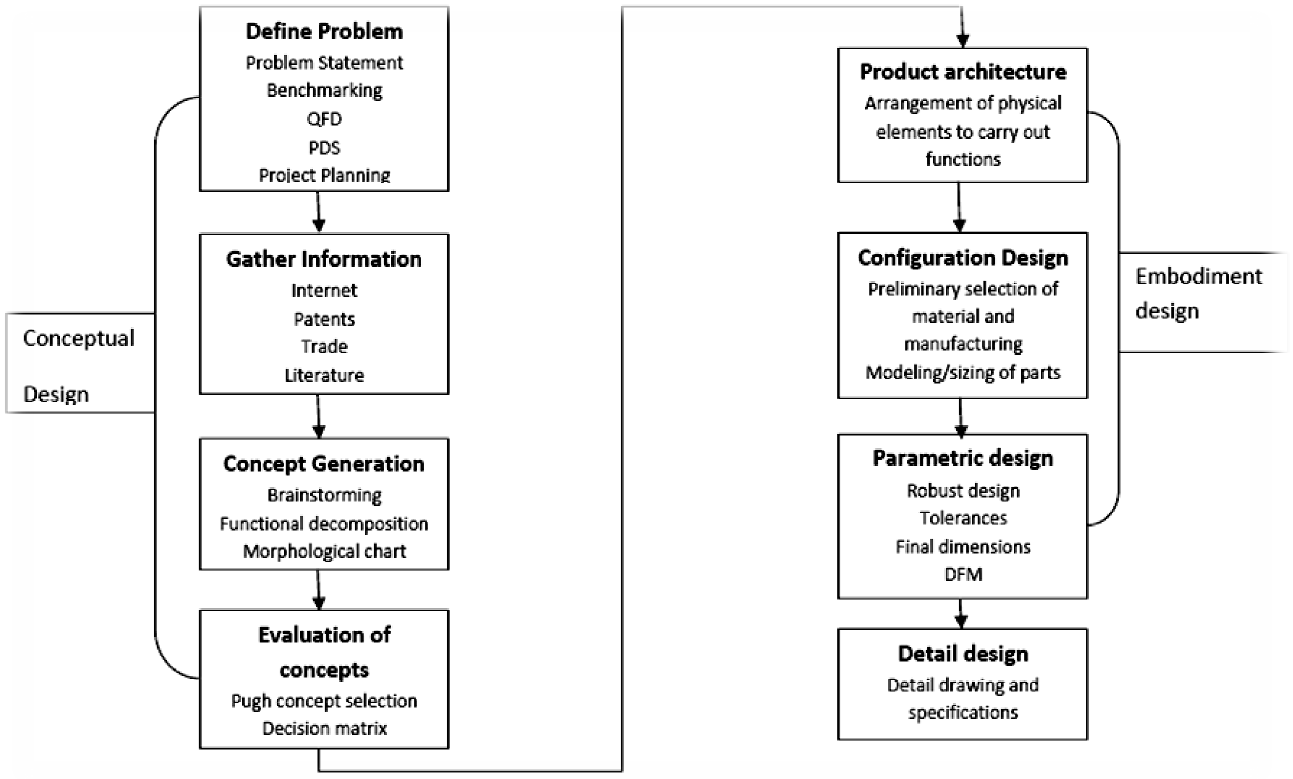


Figure.1.6. Engineering Design Process

Conceptual Design

It is a process in which we initiate the design and come up with a number of design concepts and then narrow down to the single best concept. This involved the following steps.

(1) *Identification of customer needs*: The mail objective of this is to completely understand the customers' needs and to communicate them to the design team

(2) *Problem definition*: The mail goal of this activity is to create a statement that describes what all needs to be accomplished to meet the needs of the customers' requirements.

(3) *Gathering Information*: In this step, we collect all the information that can be helpful for developing and translating the customers' needs into engineering design.

(4) *Conceptualization*: In this step, broad sets of concepts are generated that can potentially satisfy the problem statement

(5) *Concept selection*: The main objective of this step is to evaluate the various design concepts, modifying and evolving into a single preferred concept.

Embodiment Design

It is a process where the structured development of the design concepts takes place. It is in this phase that decisions are made on strength, material selection, size shape and spatial compatibility. Embodiment design is concerned with three major tasks – product architecture, configuration design, and parametric design.

(1) *Product architecture*: It is concerned with dividing the overall design system into small subsystems and modules. It is in this step we decide how the physical components of the design are to be arranged in order to combine them to carry out the functional duties of the design.

(2) *Configuration design*: In this process we determine what all features are required in the various parts / components and how these features are to be arranged in space relative to each other.

(3) *Parametric design*: It starts with information from the configuration design process and aims to establish the exact dimensions and tolerances of the product. Also, final decisions on the material and manufacturing processes are done if it has not been fixed in the previous process. One of the important aspects of parametric designs is to examine if the design is robust or not.

Detail Design

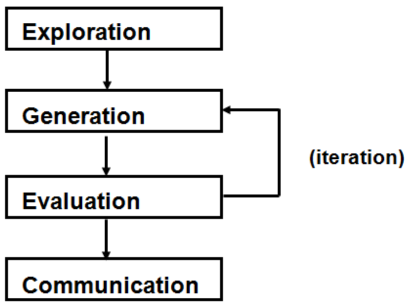
It is in this phase the design is brought to a state where it has the complete engineering description of a tested and a producible product. Any missing information about the arrangement, form, material, manufacturing process, dimensions, tolerances etc of each part is added and detailed engineering drawing suitable for manufacturing are prepared.

1.4.2. Models of the Design Process

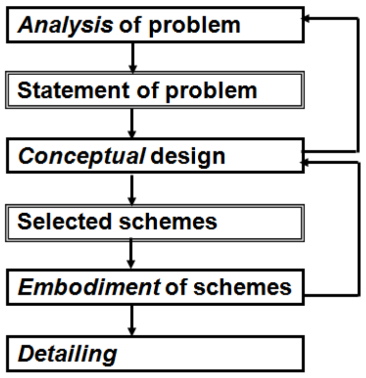
Designers have to:

Explore - the problem ‘territory’
Generate - solution concepts
Evaluate - alternative solution concepts
Communicate - a final proposal

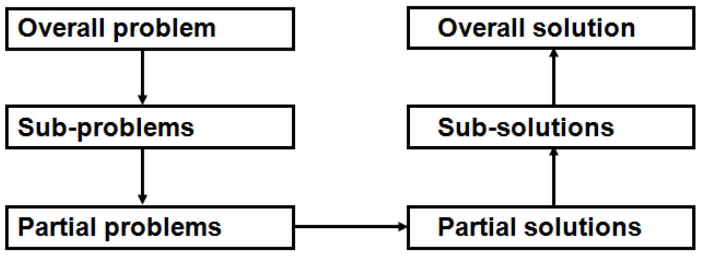
A simple model of the design process, derived from what designers have to do



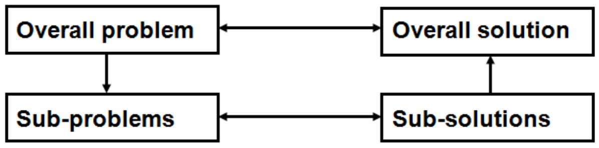
French's model



VDI model



Cross's basic model



1.4.3. New Design Procedures

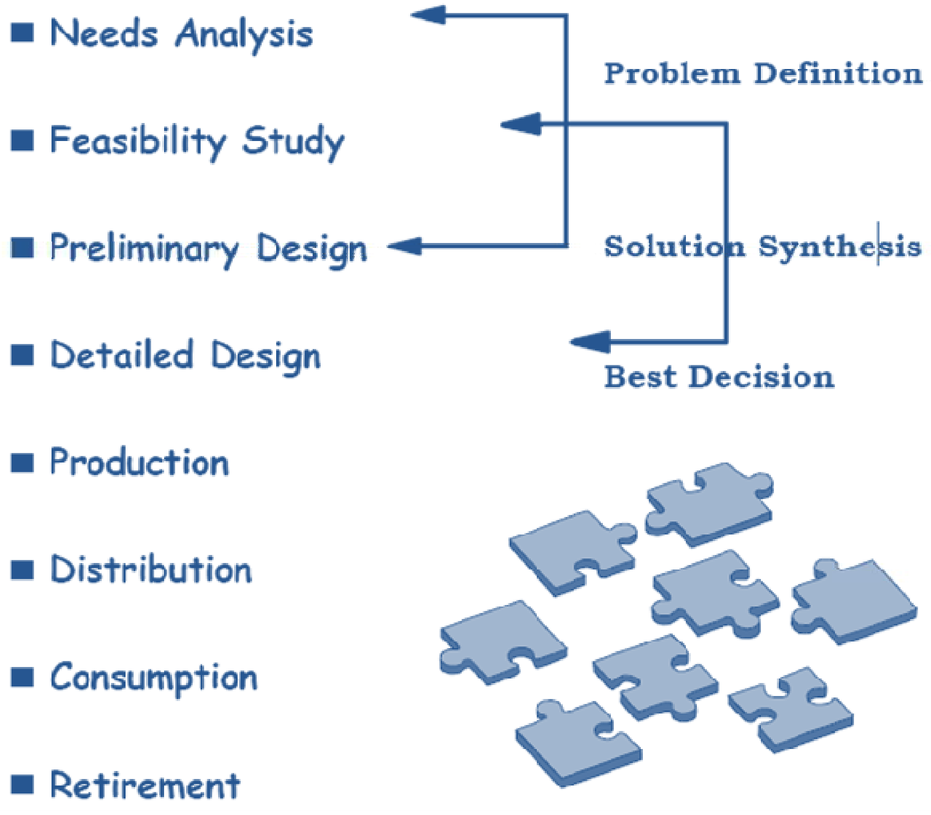
Identifying Opportunities **User Scenarios**	Determining Characteristics **Quality Function Deployment**
Clarifying Objectives **Objectives Tree**	Generating Alternatives **Morphological Chart**
Establishing Functions **Function Analysis**	Evaluating Alternatives **Weighted Objectives**
Setting Requirements **Performance Specification**	Improving Details **Value Engineering**

1.4.4. Need for Applying Technology in the Design Process

- Design is the essence of engineering
- Starts with recognition of some need
- Progresses to physical implementation
- Results may be simple or complex
- Design can be of two kind:
    - Something completely new , or
    - An improved form of something already in existence

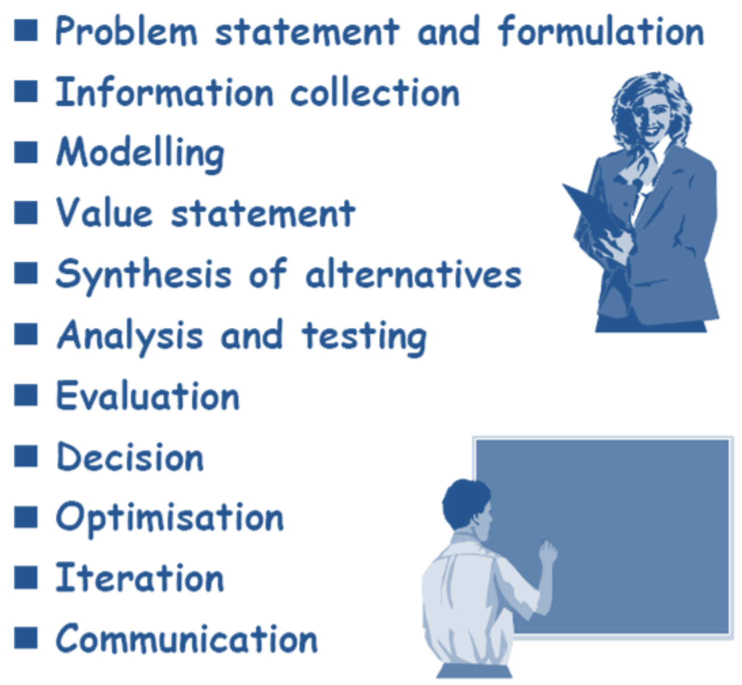
1.5. MORPHOLOGY OF DESIGN

The consideration of the product life from its conception to retirement.....



Anatomy of Design

Detailed examination of the engineer’s actions as he/she identifies and solves the problem:



- Problem statement and formulation
- Information collection
- Modelling
- Value statement
- Synthesis of alternatives
- Analysis and testing
- Evaluation
- Decision
- Optimisation
- Iteration
- Communication

1.5.1. Needs Analysis

- Creation begins by recognizing a need
    - Apparent from observation
    - Results of a detailed study
    - A specific set of circumstances

- Results in a primitive statement
    - Fact or opinion
    - Does the need exist and is it realistic?
    - Does it exist now or will it exist in the future?
    - Is it a new need? (new material or physical principle)

- Often depends on circumstances

- Needs analysis once through the Anatomy provides a good starting point for the Feasibility Study

1.5.2. Feasibility Study

- Designs can be futile unless satisfying the original need is feasible
- At this stage, the product appears in abstract forms, but is they feasible???
- Alternative solutions must be subjected to physical and economic analyses and be realizable from both
- The Feasibility Study using analysis of several alternatives establishes the design concept as something which can be realised and accepted

- Some examples.....
  (i) A building must be comfortable to live in:

  Heating, ventilation and air conditioning are required. Specify limits of temperature, humidity, velocity and fresh air constituency.

  (ii) National fossil fuel supplies are low:

  Alternative forms of energy supply are required. Specify amount and where they are needed, and any restrictions of space, time or pollution levels.

1.5.3. Preliminary Design

- Main purpose is selection of the best possible solution from a choice of alternatives
- Make comparisons against given criteria & constraints
- Must maintain an open mind; use your judgement.

1.5.4. Detailed design

- Aim is to produce a complete set of working drawings which are then transmitted to the manufacturer
- This stage of design is far less flexible than those previous
- Design should now reflect all of the planning both for manufacture and consumption stages
- Construction/testing of various components may be required
- Prototype building ....is it what was expected?

1.5.5. Production

- Here, the device or system is actually constructed, and planning for this should have been incorporated into the design
- Knowledge of the capability of the machines is required, since it must be possible to build and assemble the components as specified
- Special jigs, fixtures and even machines may be required
- Planning is vital; including quality control hold points, methods of inspection, standards for comparison etc...
- Timing of construction may be important eg. Climatics

1.5.6. Distribution

- Transportation of the manufactured article, complete or in subassembly form must be anticipated in the design
- Packaging, availability of vehicles, regulations for use of thoroughfares, shelf/component life, warehouse storage facilities, special handling, environmental control of temperature and humidity may need to be addressed

1.5.7. Consumption

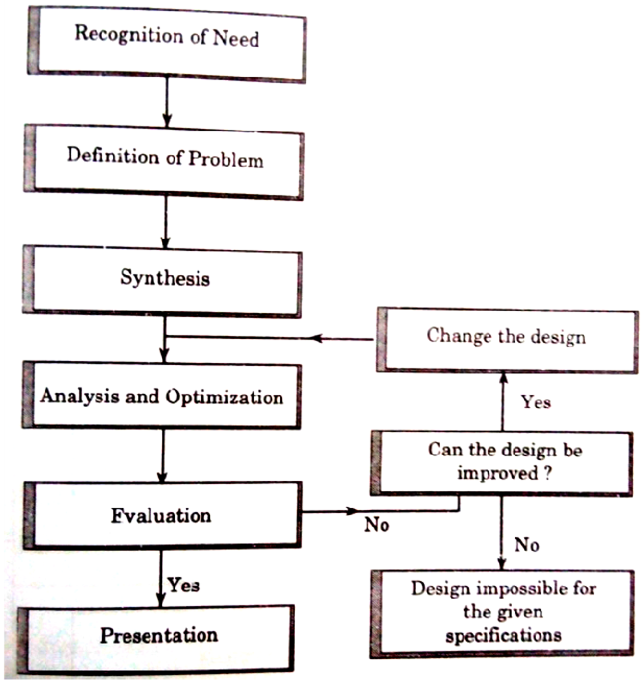
- The product is now used by the consumer
- If the design is effect, it will have met the need
- The design may yet not be complete; redesigns and modifications may be required depending on field trials or consumer feedback
- May need to consider maintenance of components and supply of spare parts or subassemblies

1.5.8. Retirement

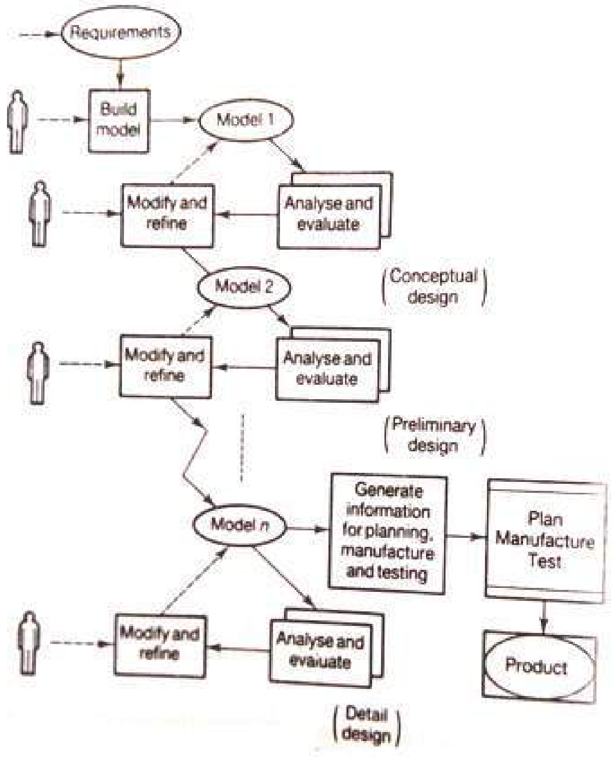
- The product will be discarded as its life cycle terminates
- It may have become obsolete whilst still serviceable and therefore the design may not have been fully economical
- Disposal and recovery of useful materials should have been included in the design
- Threats to safety should be guarded against

1.6. DESIGN PROCESS MODELS

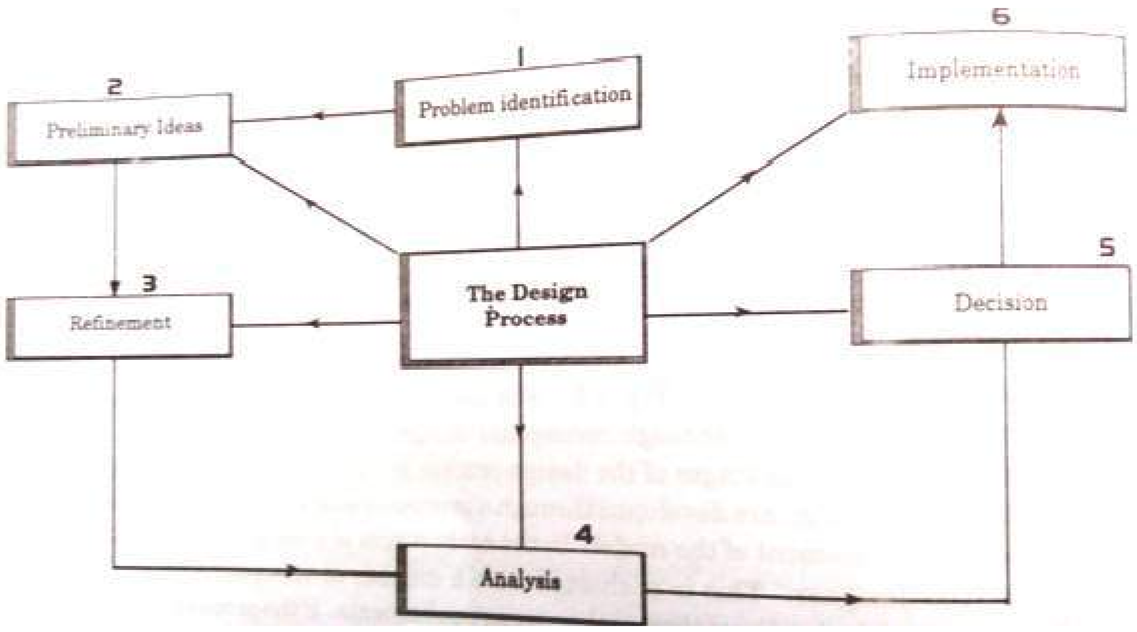
1.6.1. Shigley Model


Recognition of Need
Definition of Problem
Synthesis
Change the design
Analysis and Optimization
Yes
Can the design be improved ?
Fvaluation
No
No
Yes
Presentation
Design impossible for the given specifications

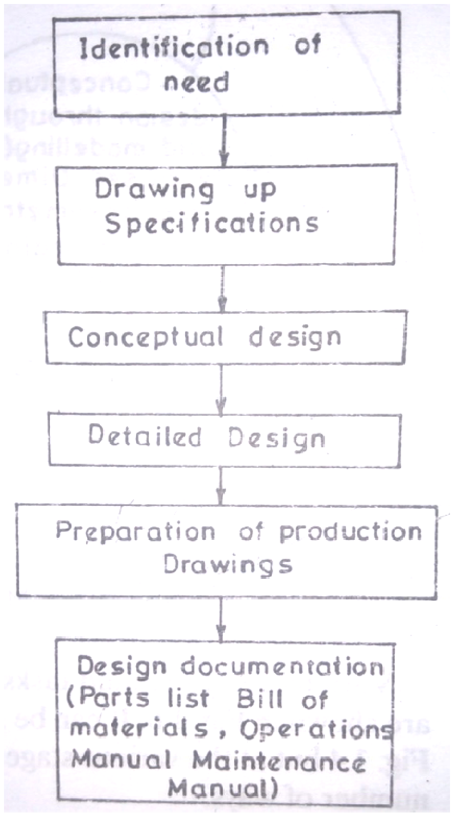
1.6.2. Ohsuga Model



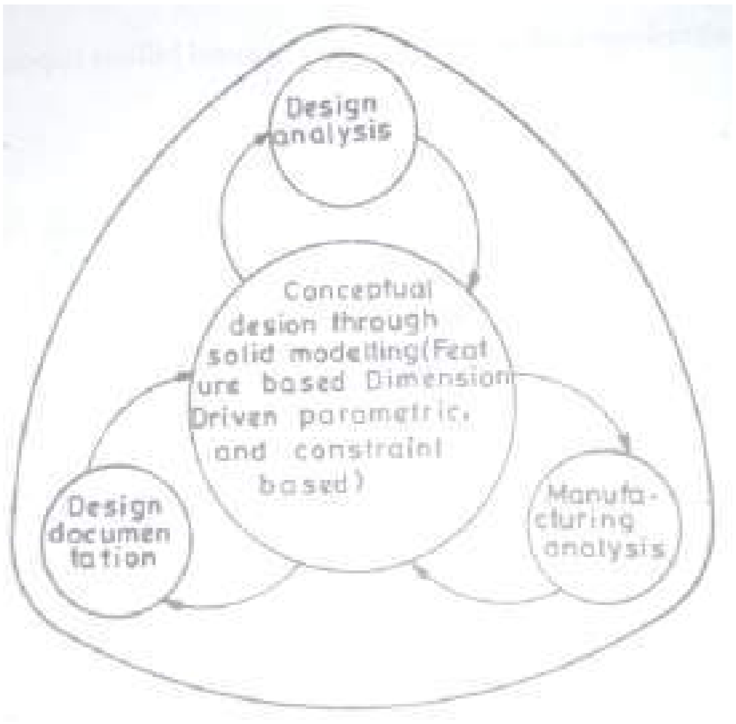
1.6.3. Earle Model



1.7. SEQUENTIAL ENGINEERING DESIGN


Identification of need
Drawing up Specifications
Conceptual design
Detailed Design
Preparation of production Drawings
Design documentation (Parts list Bill of materials, Operations Manual Maintenance Manual)

1.8. CONCURRENT ENGINEERING DESIGN


Design analysis
Conceptual design through solid modelling(Feature based Dimension Driven parametric, and constraint based)
Design documentation
Manufacturing analysis

SEQUENTIAL AND CONCURRENT ENGINEERING

With today's marketplace becoming more and more competitive, there is an ever-increasing pressure on companies to respond quickly to market needs, be cost effective, reduce lead-times to market and deliver superior quality products.

Traditionally, design has been carried out as a sequential set of activities with distinct non-overlapping phases. In such an approach, the life-cycle of a product starts with the identification of the need for that product. These needs are converted into product requirements which are passed on to the design department. The designers design the product's form, fit, and function to meet all the requirements, and pass on the design to the manufacturing department.

After the product is manufactured it goes through the phases of assembly, testing, and installation. This type of approach to life-cycle development is also known as `over the wall' approach, because the different life-cycle phases are hidden or isolated from each other. Each phase receives the output of the preceding phase as if the output had been thrown over the wall. In such an approach, the manufacturing department, for example, does not know what it will actually be manufacturing until the detailed design of the product is over.

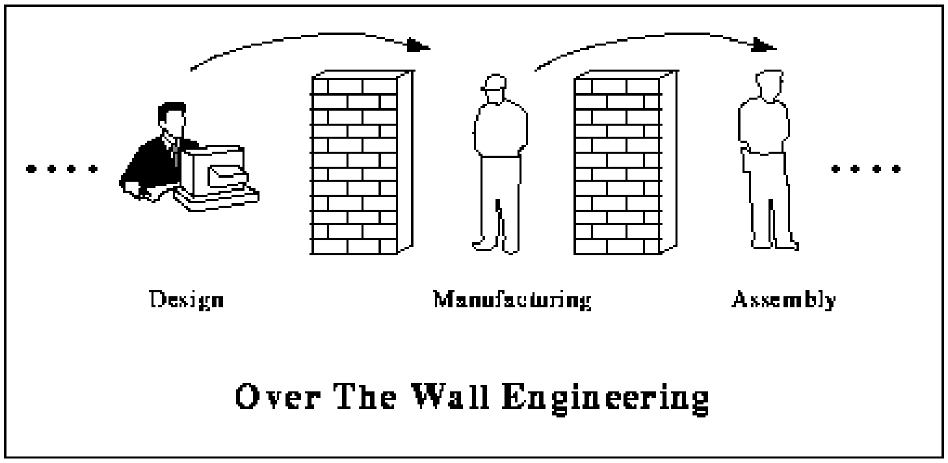


Figure.1.8.Over the Wall Engineering (Sequential Engineering)

There are a lot of disadvantages of the sequential engineering process. The designers are responsible for creating a design that meets all the specified requirements. They are usually not concerned with how the product will be manufactured or assembled. Problems and inconsistencies in the designs are therefore, detected when the product reaches into the later phases of its life-cycle. At this stage, the only possible option is to send the product back for a re-design. The whole process becomes iterative and it not until after a lot of re-designs has taken place that the product is finally manufactured. Because of the large number of changes, and hence iterations, the product's introduction to market gets delayed. In addition, each re-design, re-work, re-assembly etc. incurs cost, and therefore the resulting product is costlier than what it was originally thought to be. The market share is lost because of the delay in product's introduction to market, and customer faith is lost. All this is undesirable.

Concurrent Engineering is a dramatically different approach to product development in which various life-cycle aspects are considered simultaneously right from the early stages of design. These life-cycle aspects include product's functionality, manufacturability, testability, assimilability, maintainability, and everything else that could be affected by the design.

In addition, various life-cycle phases overlap each other, and there in no "wall" between these phases. The completion of a previous life-cycle phase is not a pre-requisite for the start of the next life-cycle phase. In addition, there is a continuous feedback between these life-cycle phases so that the conflicts are detected as soon as possible.

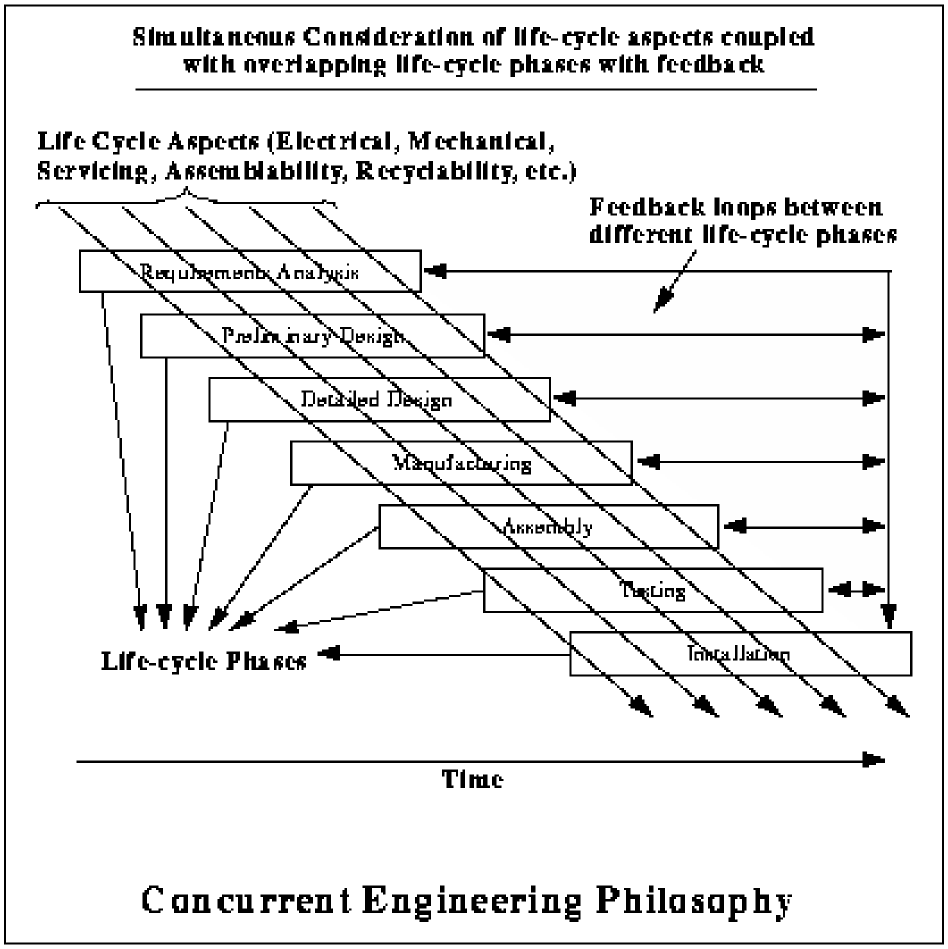

Simultaneous Consideration of life-cycle aspects coupled with overlapping life-cycle phases with feedback
Life Cycle Aspects (Electrical, Mechanical, Servicing, Assemblability, Recyclability, etc.)
Feedback loops between different life-cycle phases
Requirements Analysis
Preliminary Design
Detailed Design
Manufacturing
Assembly
Testing
Installation
Life-cycle Phases
Time
Concurrent Engineering Philosophy

Figure.1.9. Concurrent Engineering

The concurrent approach results in less number of changes during the later phases of product life-cycle, because of the fact that the life-cycle aspects are being considered all through the design. The benefits achieved are reduced lead times to market, reduced cost, higher quality, greater customer satisfaction, increased market share etc. Sequential engineering is the term used to describe the method of production in a linear format. The different steps are done one after another, with all attention and resources focused on that one task. After it is completed it is left alone and everything is concentrated on the next task.

In concurrent engineering, different tasks are tackled at the same time, and not necessarily in the usual order. This means that info found out later in the process can be added to earlier parts, improving them, and also saving a lot of time. Concurrent engineering is a method by which several teams within an organization work simultaneously to develop new products and services and allows a more stream lined approach. The concurrent engineering is a non-linear product or project design approach during which all phases of manufacturing operate at the same time - simultaneously. Both product and process design run in parallel and occur in the same time frame.

Product and process are closely coordinated to achieve optimal matching of requirements for effective cost, quality, and delivery. Decision making involves full team participation and involvement. The team often consists of product design engineers, manufacturing engineers, marketing personnel, purchasing, finance, and suppliers.

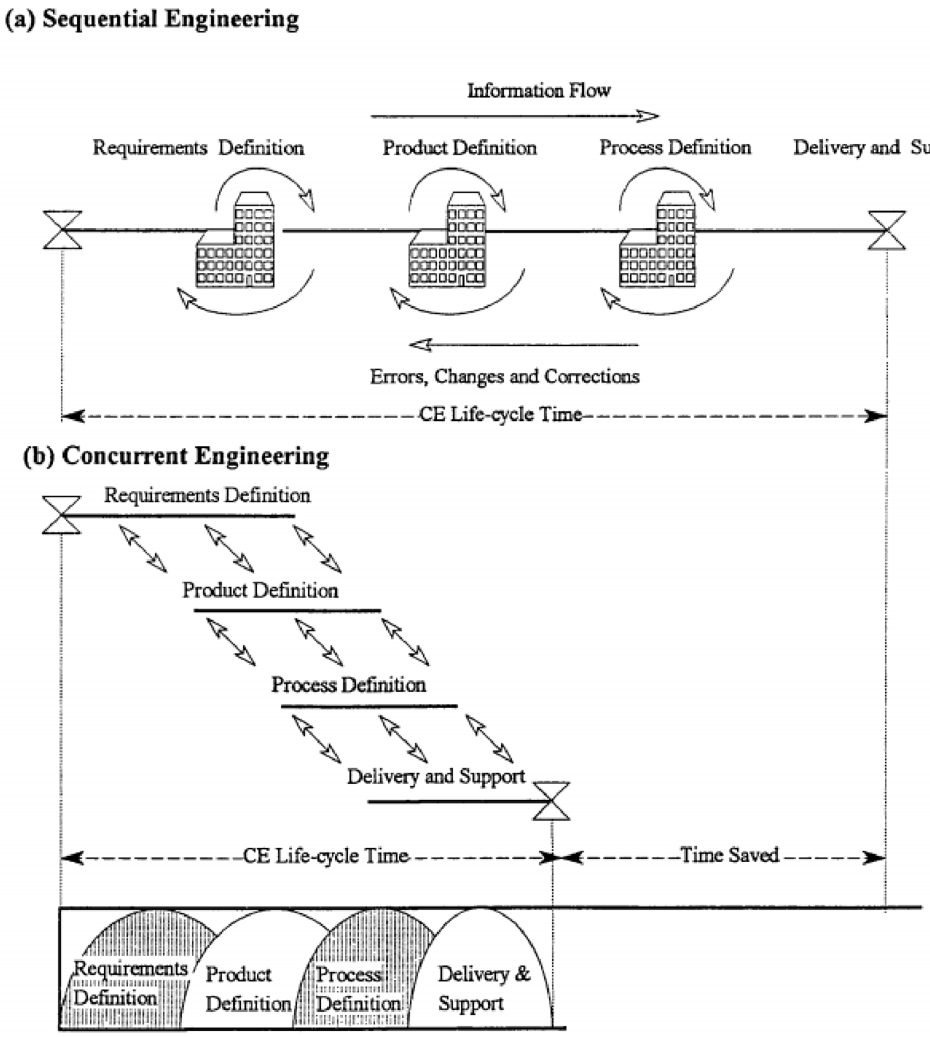
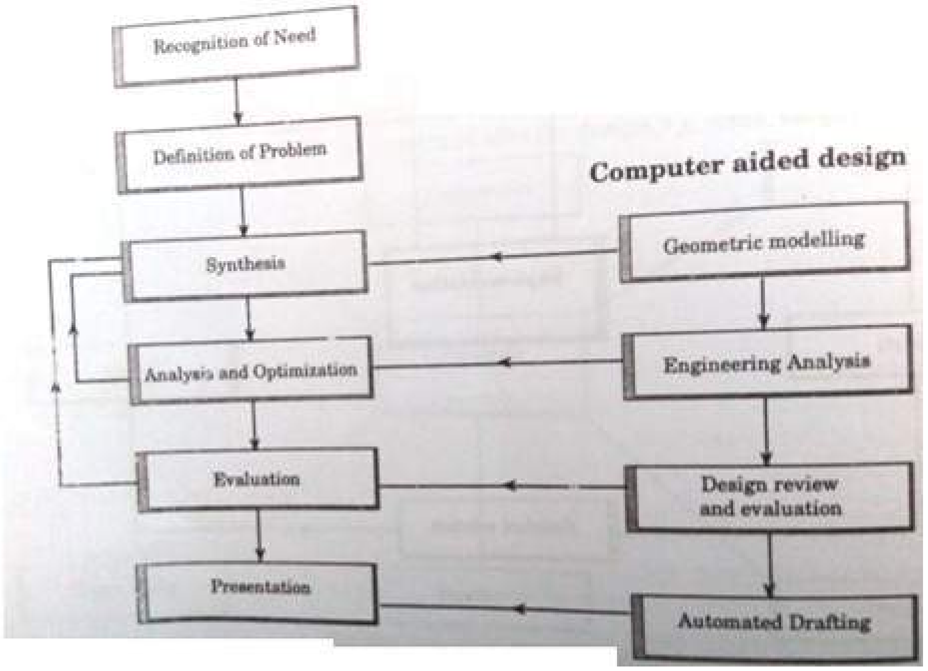
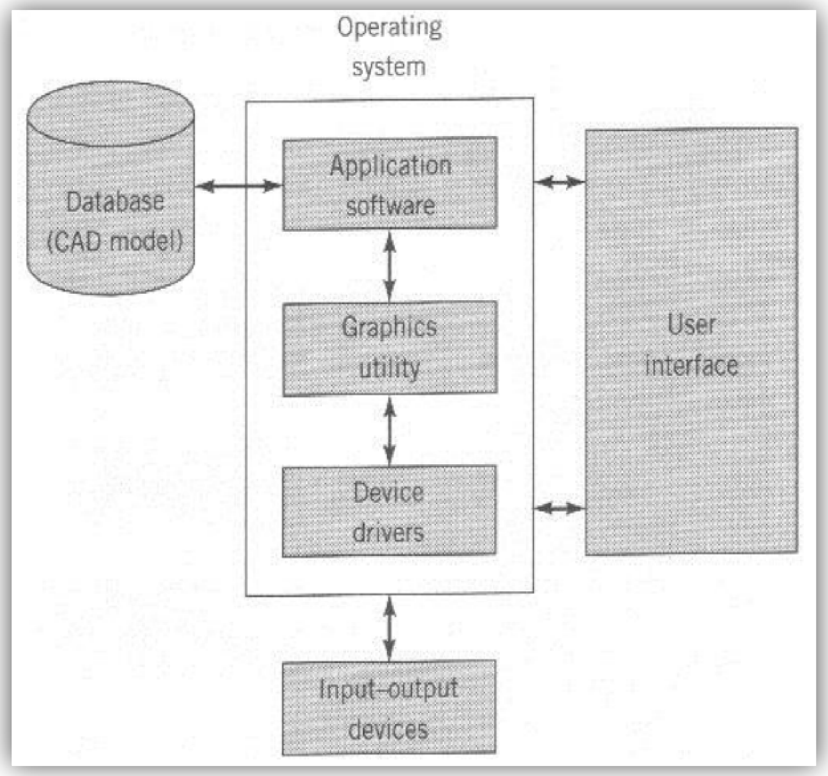


Figure.1.9. Sequential and Concurrent Engineering

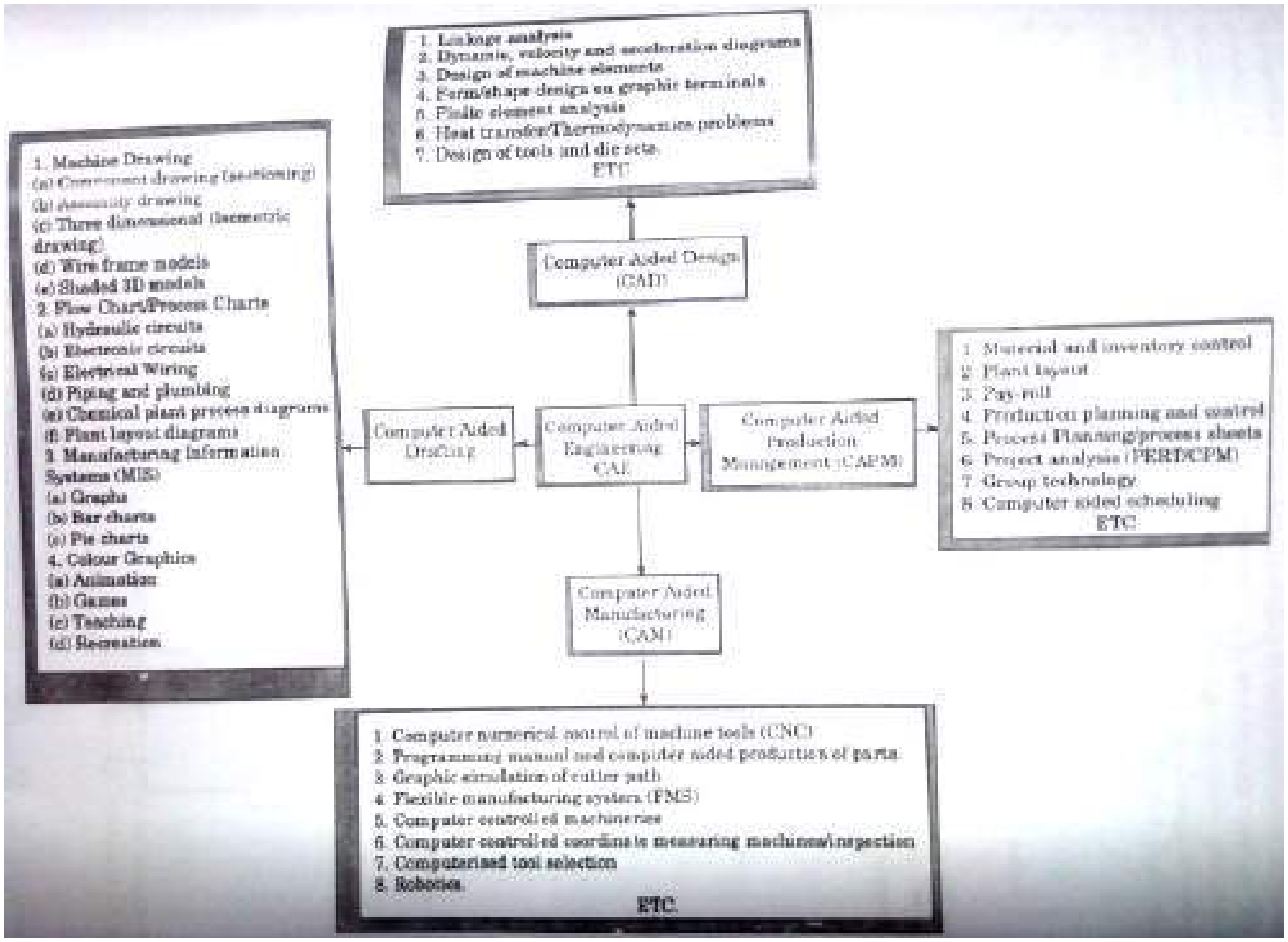
1.9. ROLE OF COMPUTERS IN DESIGN



1.10. CAD SYSTEM ARCHITECTURE



1.11. COMPUTER AIDED ENGINEERING – CAD/CAM


1. Linkage analysis
2. Dynamic, velocity and acceleration diagrams
3. Design of machine elements
4. Form/shape design on graphic terminals
5. Finite element analysis
6. Heat transfer/Thermodynamics problems
7. Design of tools and die sets
ETC
Computer Aided Design (CAD)
1. Machine Drawing
(a) Component drawing (sectioning)
(b) Assembly drawing
(c) Three dimensional (Isometric drawing)
(d) Wire frame models
(e) Shaded 3D models
2. Flow Chart/Process Charts
(a) Hydraulic circuits
(b) Electronic circuits
(c) Electrical Wiring
(d) Piping and plumbing
(e) Chemical plant process diagrams
(f) Plant layout diagrams
3. Manufacturing Information Systems (MIS)
(a) Graphs
(b) Bar charts
(c) Pie charts
4. Colour Graphics
(a) Animation
(b) Games
(c) Teaching
(d) Recreation
Computer Aided Drafting
Computer Aided Engineering CAE
Computer Aided Production Management (CAPM)
1. Material and inventory control
2. Plant layout
3. Pay-roll
4. Production planning and control
5. Process Planning/process sheets
6. Project analysis (PERT/CPM)
7. Group technology
8. Computer aided scheduling
ETC
Computer Aided Manufacturing (CAM)
1. Computer numerical control of machine tools (CNC)
2. Programming manual and computer aided production of parts
3. Graphic simulation of cutter path
4. Flexible manufacturing system (FMS)
5. Computer controlled machineries
6. Computer controlled coordinate measuring machines/inspection
7. Computerised tool selection
8. Robotics
ETC.

1.12. APPLICATION OF COMPUTERS TO DESIGN

- Modeling of the Design
- Engineering design and analysis
- Evaluation of Prototype through Simulation and Testing
- Drafting and Design Documentation

1.13. BENEFITS OF CAD

1. Productivity Improvement in Design
   Depends on Complexity of drawing,
   Degree of repetitiveness of features in the designed parts,
   Degree of symmetry in the parts,
   Extensive use of library of user defined shapes and commonly used entities
2. Shorter Lead Times
3. Flexibility in Design
4. Design Analysis
5. Fewer Design Error
6. Standardization of Des

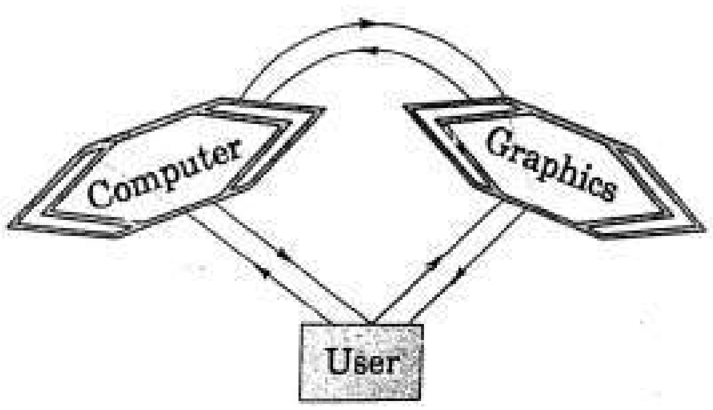
7. Drawings are more understandable
8. Improved Procedures of Engineering Changes
9. Benefits in Manufacturing :
    a. Tool and fixture design for manufacturing
    b. Computer Aided process planning
    c. Preparation of assembly lists and bill of materials
    d. Computer aided inspection
    e. Coding and classification of components
    f. Production planning and control
    g. Preparation of numerical control programs for manufacturing the parts on CNC machines
    h. Assembly sequence planning

1.14. REASONS FOR IMPLEMENTING CAD

- To increase the productivity of the designer
- To improve the Quality of Design
- To improve Documentation
- To create a Database for manufacturing

1.15. COMPUTER GRAPHICS or INTERACTIVE COMPUTER GRAPHICS

- Computer Graphics is defined as creation, storage, and manipulation of pictures and drawings by means of a digital computer
- It is an extremely effective medium for communication between people and computers
- Computer graphics studies the manipulation of visual and geometric information using computational techniques
- It focuses on the mathematical and computational foundations of image generation and processing rather than purely aesthetic issues



Concept of Interactive computer Graphics

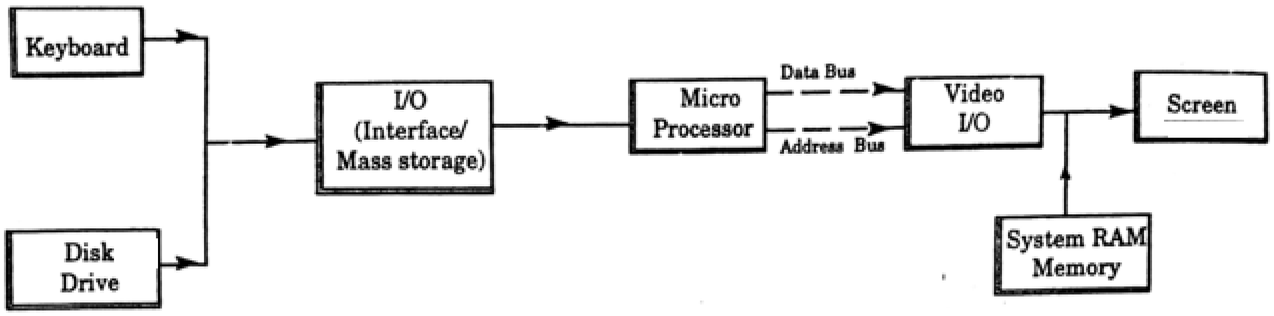
In Interactive Computer Graphics (ICG) the user interacts with the compute and comprises the following important functions:

Modeling, which is concerned with the description of an object in terms of its spatial coordinates, lines, areas, edges, surfaces, and volume

Storage, which is concerned with the storage of the model in the memory of the computer

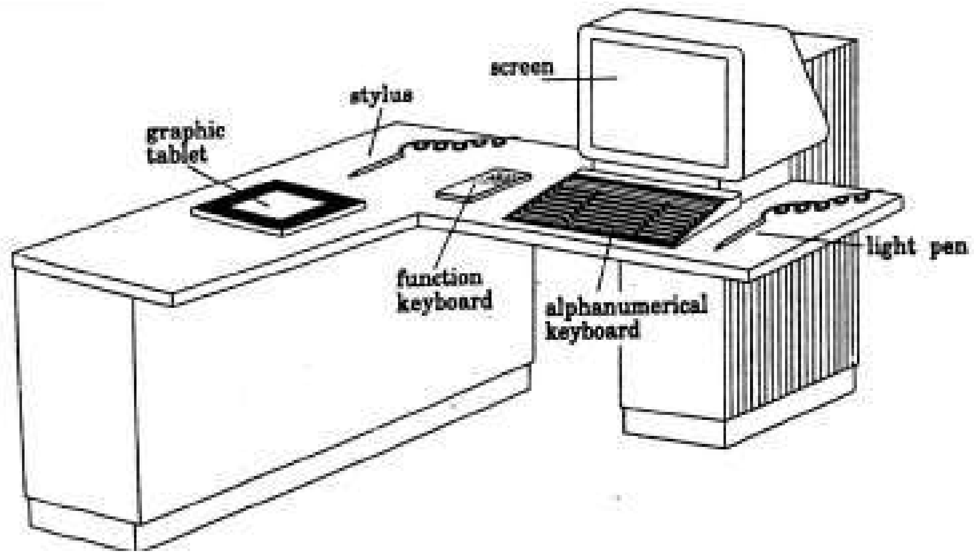
Manipulation, which is used in the construction of the model from basic primitives in combination with Boolean algebra

Viewing, in the case the computer is used to look at the model from a specific angle and presents on its screen what it sees.



Typical Hardware setup of a Graphic System

Work Station: A workstation comprises of the devices that allow the user to create and design objects, using both graphic and non-graphic instructions and data. A Stand alone workstation refers to CAD workstations that can process data and output information independent of other computer systems or workstations. It includes its own software, hardware, and peripherals.



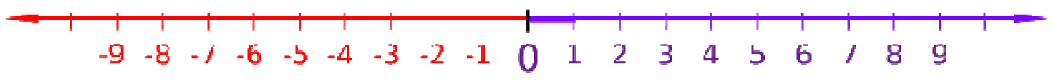
A typical work station

1.16. CO-ORDINATE SYSTEMS

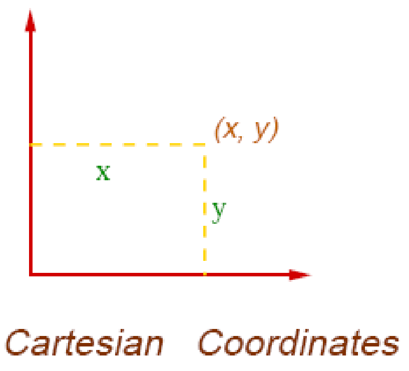
A coordinate system is one which uses one or more numbers, or coordinates, to uniquely determine the position of a point or other geometric element on a manifold such as Euclidean space.

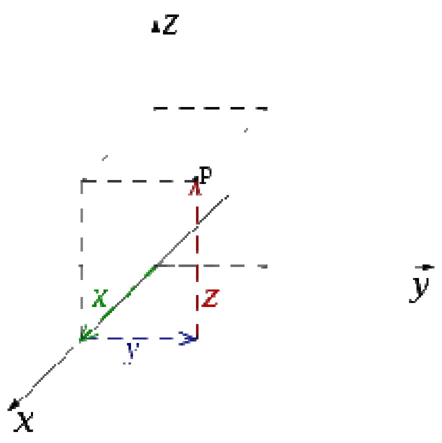
Common coordinate systems are:

- **Number line**
  The simplest example of a coordinate system is the identification of points on a line with real numbers using the number line. In this system, an arbitrary point O (the origin) is chosen on a given line. The coordinate of a point P is defined as the signed distance from O to P, where the signed distance is the distance taken as positive or negative depending on which side of the line P lies. Each point is given a unique coordinate and each real number is the coordinate of a unique point

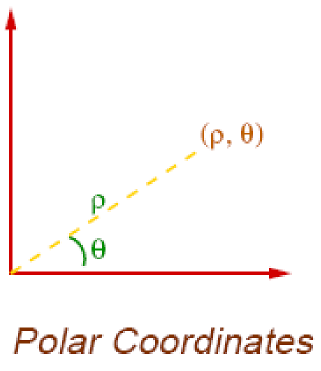


- **Cartesian coordinate system [(x,y) and (x,y,z)]**





- **Polar coordinate system (ρ,θ)**
  Another common coordinate system for the plane is the *polar coordinate system*. A point is chosen as the *pole* and a ray from this point is taken as the *polar axis*.



ρ = r

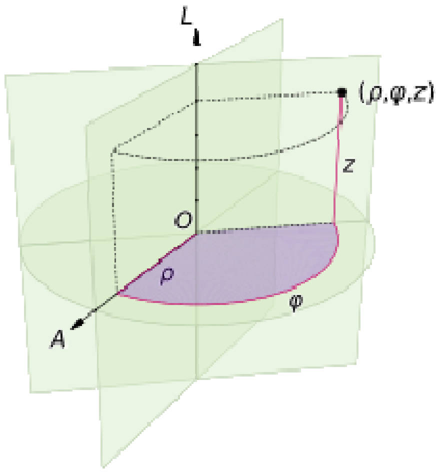
For a given angle θ, there is a single line through the pole whose angle with the polar axis is θ (measured counter clockwise from the axis to the line). Then there is a unique point on this line whose signed distance from the origin is *r* for given number *r*. For a given pair of coordinates (*r*, θ) there is a single point, but any point is represented by many pairs of coordinates. For example (*r*, θ), (*r*, θ+2π) and (−*r*, θ+π) are all polar coordinates for the same point. The pole is represented by (0, θ) for any value of θ.

Cylindrical Coordinate systems

A cylindrical coordinate system is a three-dime
system that specifies point positions by the distance from a chos
the direction from the axis relative to a chosen reference directio
from a chosen reference plane perpendicular to the axis. The latte
as a positive or negative number depending on which side of t
faces the point.

The origin of the system is the point where all three coordi
as zero. This is the intersection between the reference plane and th

The axis is variously called the cylindrical or lon
differentiate it from the polar axis, which is the ray that lies in th
starting at the origin and pointing in the reference direction.



The distance from the axis may be called the radial distan
the angular coordinate is sometimes referred to as the angu
the azimuth.

***θ* is elevation:**

$$\rho = r\cos\theta$$
$$\varphi = \varphi$$
$$z = r\sin\theta$$

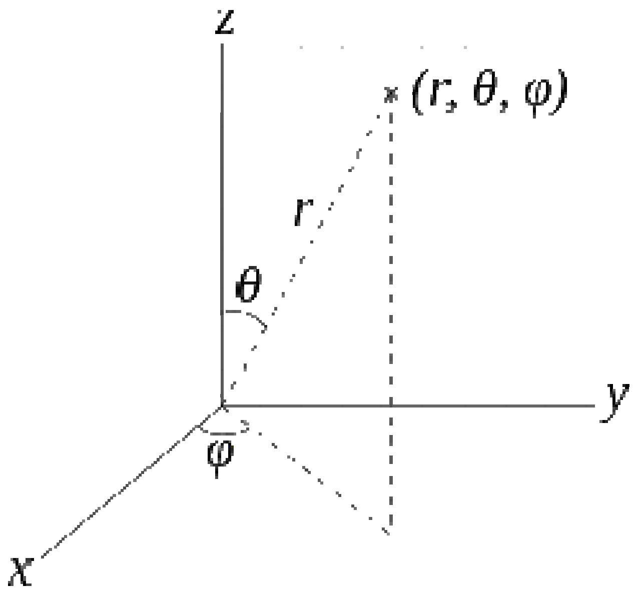
***θ* is inclination:**

$$\rho = r\sin\theta$$
$$\varphi = \varphi$$
$$z = r\cos\theta$$

- **Spherical Coordinate systems**

A spherical coordinate system is a coordinate system for three-dimensional space where the position of a point is specified by three numbers: the radial distance of that point from a fixed origin, its polar angle measured from a fixed zenith direction, and the azimuth angle of its orthogonal projection on a reference plane that passes through the origin and is orthogonal to the zenith, measured from a fixed reference direction on that plane.

The radial distance is also called the radius or radial coordinate. The polar angle may be called co-latitude, zenith angle, normal angle, or inclination angle



θ is elevation:

$$r = \sqrt{\rho^2 + z^2}$$
$$\theta = \arctan(z/\rho)$$
$$\varphi = \varphi$$

θ is inclination:

$$r = \sqrt{\rho^2 + z^2}$$
$$\theta = \arctan(\rho/z)$$
$$\varphi = \varphi$$

- **Homogeneous coordinate system**

Three dimensional representation of a two dimensional plane is called Homogeneous Co-ordinates. The respective system is called Homogeneous coordinate system.

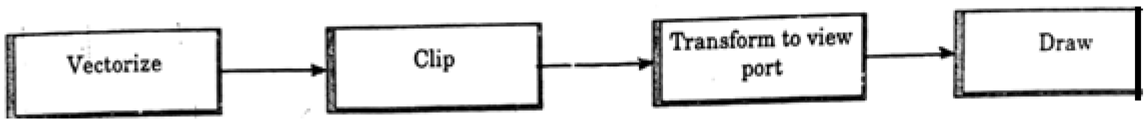
1.17. 2 – D DISPLAY CONTROL FACILITIES

The essential steps for 2D graphics are:

1. Convert the geometric representation of the model to lines (termed Vectors)
2. Transform the lines from the model coordinate system to the screen coordinate system (termed windowing)
3. Select those lines that are within the part of the model that it is wished to display known as the clipping step

4. Instruct the display device to draw the vectors

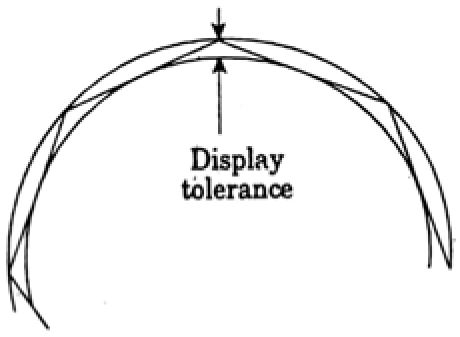
The Stages in graphics pipeline are shown.



Stages in graphics pipeline

1. **Vector Generation**

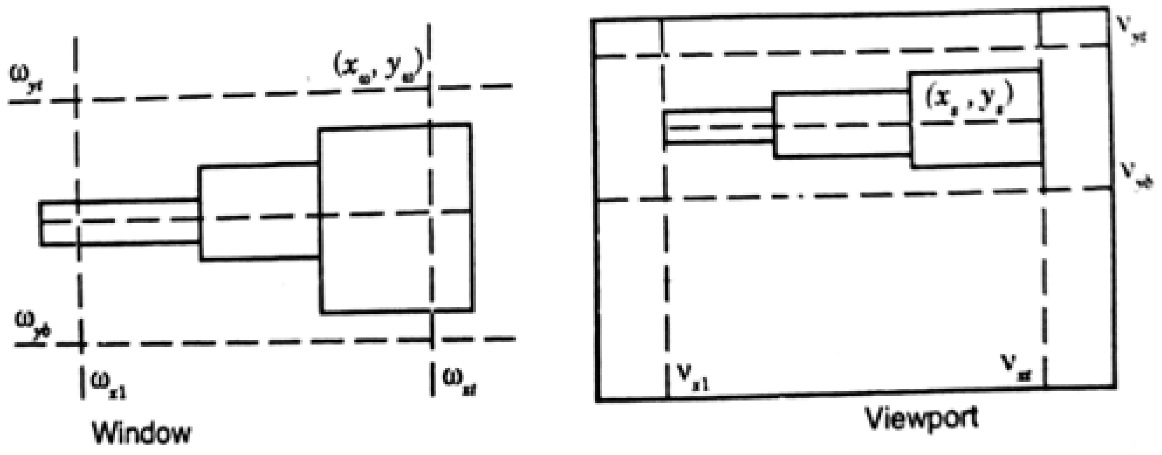
   The aim of vector display of a curve is to use sufficient vectors for the curve to appear smooth. The number needed is controlled by the display tolerance, which is maximum deviation of the vector representation from the true curve shape.



2. **Windowing Transformation**

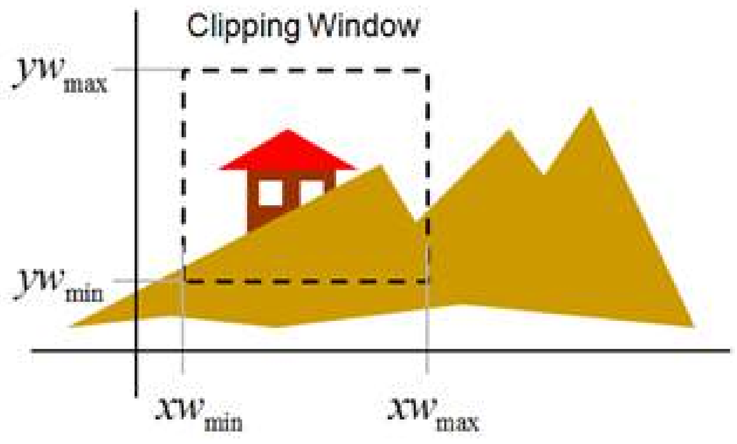
   When it is necessary to examine in detail a part of a picture being displayed, a window may be placed around the desired part and the windowed area magnified to fill the whole screen and multiple views of the model may also be shown on the same screen.

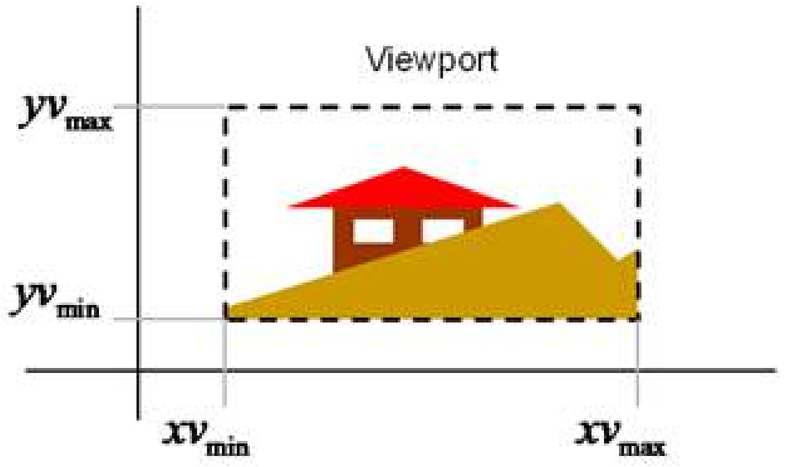
   The window is a rectangular frame or boundary through which the user looks onto the model. The viewport is the area on the screen in which the contents of the window are to be presented as an image.



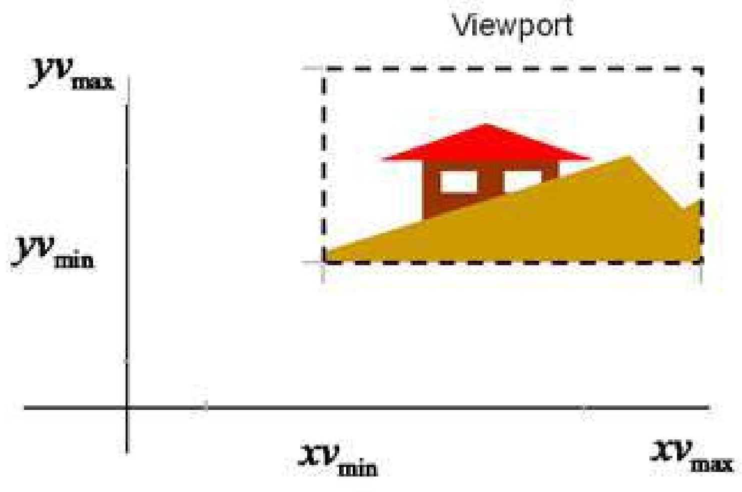
3. **Clipping Transformation**

The clipping is an operation to plot part of a picture within the given window of the plotting area and to discard the rest.





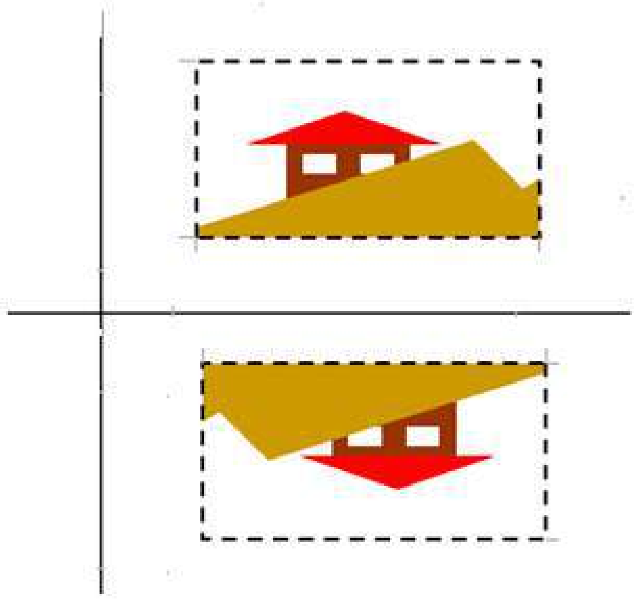
CLIPPING



CLIPPLING TRANSFORMATION

4. **Reflection Transformation**

Reflection about any axis

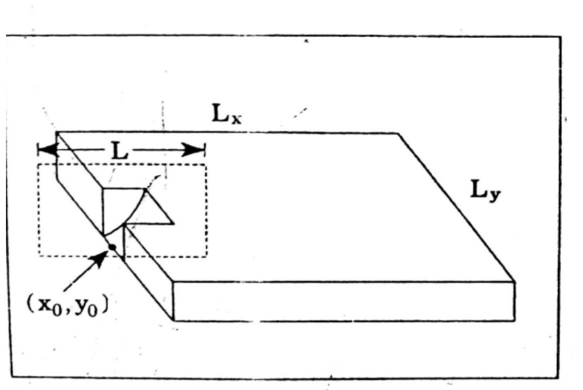


REFLECTION TRANSFORMATION

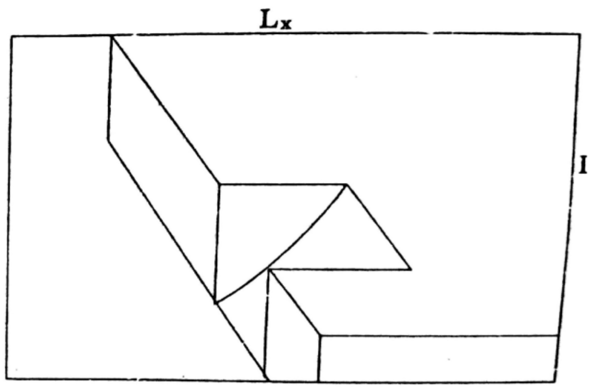
5. **Zooming**

This transformation is carried out to provide enlarged or shrunk view of a picture detail

Zooming = scaling + translation + clipping



Original picture with portion to be enlarged



After zooming of marked area.

Translation : $D_x = -x_o$, $D_y = -y_o$ (Centre of detail to origin)

Scaling: $S_x = S_y = L_x/L$

Translation : $D_x = L_x/2$, $D_y = L_y/2$,

Clipping : (to frame dimensions)

1.18. 2– D TRANSFORMATIONS

i. Translation
ii. Scaling
iii. Reflection with mirror
iv. Rotation

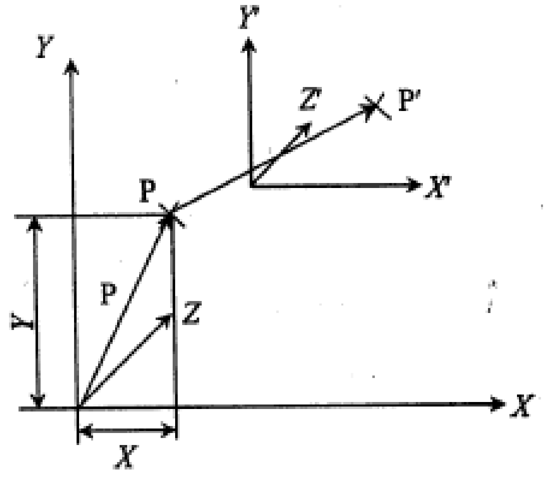
Translation

It is the most common and easily understood transformation in CAD. This moves a geometric entity in space in such a way that the new entity is parallel at all points to the old entity. A representation is shown in following figure for an object. Let us now consider a point on the object, represented by P which is translated along X and Y axes by dX and dY to a new position P'. The new coordinates after transformation are given by following equations.

$P' = [x', y']$ ------(1)

$x' = x + dX$ ------(2)

$y' = y + dY$ ------(3)



Translation of the Point

Putting equations (3) back into equations (1) we can write

$$[P'] = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + dX \\ y + dY \end{bmatrix} \qquad \text{-------(4)}$$

This can also be written in matrix form as follows.

$$[P'] = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + dX \\ y + dY \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dX \\ dY \end{bmatrix} \qquad \text{-------(5)}$$

this is normally the operation used in the CAD systems as MOVE command.

2. **Scaling**

Scaling is the transformation applied to change the scale of an entity. As shown in following figure, this alters the size of the entity by the scaling factor applied. For Example, in following figure, to achieve scaling, the original coordinates would be multiplied uniformly by the scaling factor.

$P' = [X', Y'] = [S_x \times X, S_Y \times dY]$ -------(6)

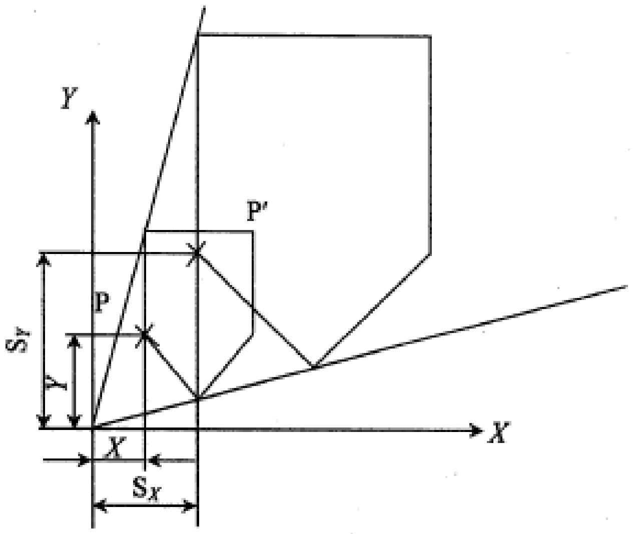
This equation can also be represented in a matrix form as follows.

$$[P'] = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \text{-------(7)}$$

$[P'] = [T_s] \cdot [P]$ -------(8)

where

$$[T_s] = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \qquad \text{-------(9)}$$

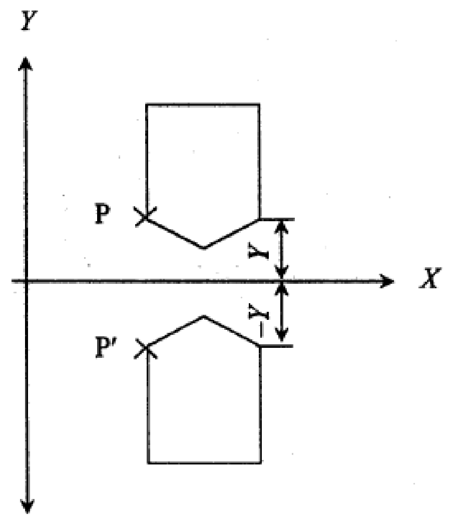


Scaling of Plane Figure

Since the scaling factors can be individually applied, there is a possibility to have differential scaling when $S_x \neq S_y$. Normally in the CAD systems uniform scaling is allowed for object manipulation. In the case of zoom facility in graphic systems, uniform scaling is applied. Zooming is only a display attribute and is applied only to the display and not to the actual geometric database.

3. Reflection or Mirror

Reflection or mirror is a transformation, which allows a copy of the object to be displayed while the object is reflected about a line or a plane.



Example for Reflection Transformation

The transformation required in this case is that the axes coordinates will get negated depending upon the reflection required. For example from following figures, the new

$P' = [X', Y'] = [X, -Y]$ -------(10)

This can be given a matrix form as

$$[P'] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{-------(11)}$$

$[P'] = [T_m] \cdot [P]$

where

$$[T_m] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{-------(12)}$$

Thus the general transformation matrix will be

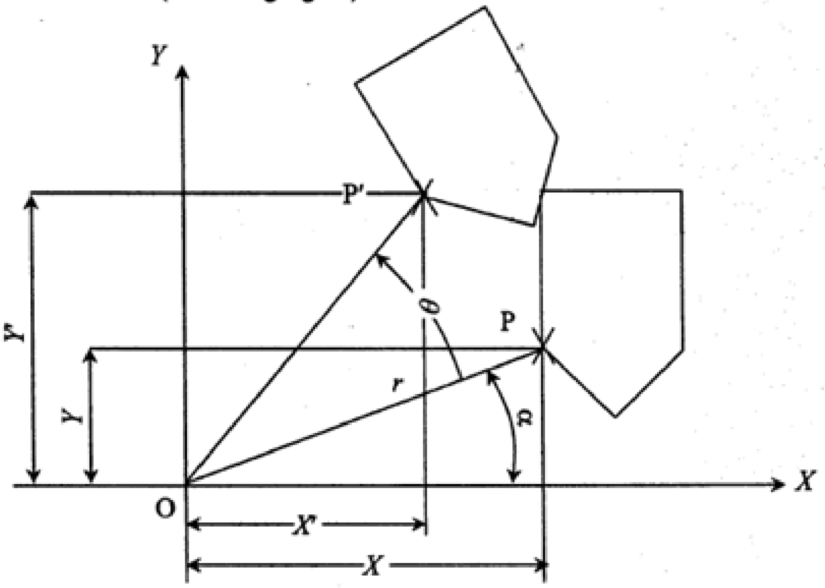
$$[M] = \begin{bmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{bmatrix} \quad \text{-------(13)}$$

Here, −1 in the first position refers to reflection about *Y*-axis where all the *X* coordinate values get negated.

When the second term becomes −1 the reflection will be about the *X*-axis with all *Y* coordinate values getting reversed. Both the values are −1 for reflection about *X* and *Y*-axes.

4. Rotation

Rotation is another important geometric transformation. The final position and orientation of a geometric entity is decided by the angle of rotation (θ) and the base point about which the rotation, is to be done (following figure)



To develop the transformation matrix for transformation, consider a point P located in XY plane, being rotated in the counter clockwise direction to the new position, P' by an angle θ as shown in following figure, The new position P' is given by
$P' = [x', y']$

From the following figure, the original position is specified by
$x = r\cos\alpha$
$y = r\sin\alpha$
The new position, P' is specified by

$$\begin{aligned} x' &= r\cos(\alpha+\theta) \\ &= r\cos\theta\cos\alpha - r\sin\theta\sin\alpha \\ &= x\cos\theta - y\sin\theta \end{aligned}$$

$$\begin{aligned} y' &= r\sin(\alpha+\theta) \\ &= r\sin\theta\cos\alpha + r\cos\theta\sin\alpha \\ &= x\sin\theta + y\cos\theta \end{aligned}$$

This can be written in a matrix form as

$$[P'] = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \text{-------(14)}$$

$$[P'] = [T_R]\cdot[P]$$

$$[T_R] = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad \text{-------(15)}$$

1.19. HOMOGENEOUS CO-ORDINATES

Homogeneous Representation

In order to concatenate the transformation as shown in equation (16), all the transformation matrices should be multiplicative type. However, as seen earlier, the translation matrix (equation (5)) is vector additive, while all others are matrix multiplications. The following form should be used to convert the translation into a multiplication form.

$$[P'] = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dX \\ 0 & 1 & dY \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \text{--------(17)}$$

Hence the translation matrix in multiplication form can be given as

$$[MT] = \begin{bmatrix} 1 & 0 & dX \\ 0 & 1 & dY \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{--------(18)}$$

This is termed as homogeneous representation. In homogeneous representation, an n-dimensional space is mapped into $(n + 1)$ dimensional space. Thus a 2 dimensions point $[x \;\; y]$ is represented by 3 dimensions as $[x \;\; y \;\; 1]$.

This greatly facilities the computer graphics operations where the concatenation of multiple transformations can be easily carried out. This can be experienced in the following situations

1.20. 3 – D TRANSFORMATIONS

The 2D transformations as explained in earlier sections can be extended to 3D by adding the

Z-axis parameter. The transformation matrix will now be 4 × 4. The following are the transformation matrices to be used for this purpose.

1. **Translation**

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dX \\ 0 & 1 & 0 & dY \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2. **Scaling**

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

3. **Reflection**

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \pm1 & 0 & 0 & 0 \\ 0 & \pm1 & 0 & 0 \\ 0 & 0 & \pm1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

4. **Rotation about *Z*-axis (*XY* plane)**

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

5. **Rotation about *Y*-axis (*ZX* plane)**

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

6. **Rotation about *X*-axis (*YZ* plane)**

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

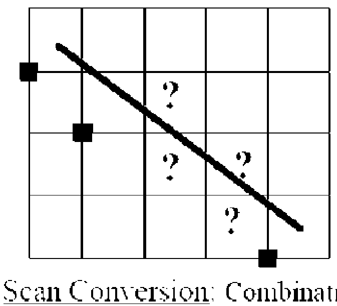
1.21. LINE DRAWING ALGORITHMS

Line Drawing Algorithms

We are going to analyze how this process is achieved.

Some useful definitions

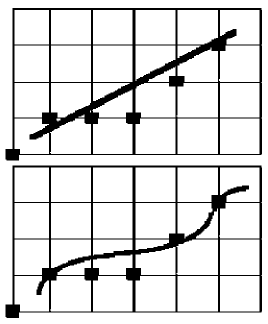
Rasterization: Process of determining which pixels provide the best approximation to a desired line on the screen.



Scan Conversion: Combination of rasterization and generating the picture in scan line order.

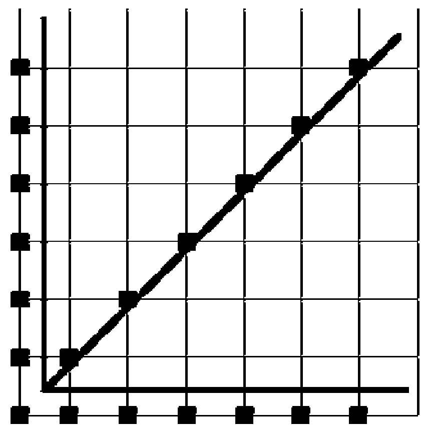
General requirements

• Straight lines must appear as straight lines.

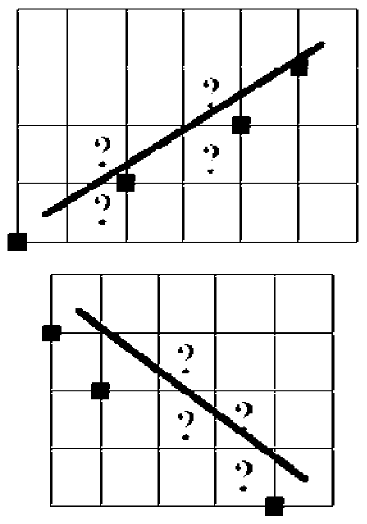


• They must start and end accurately
• Lines should have constant brightness along their length
•Lines should drawn rapidly

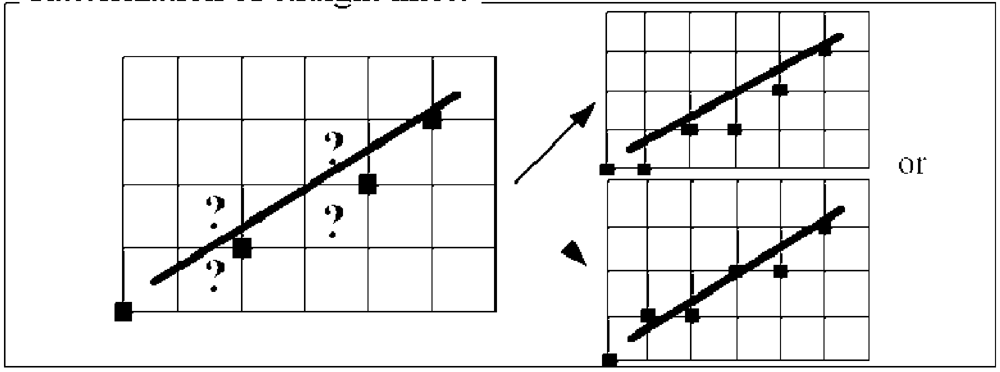
For horizontal, vertical and 45° lines, the choice of raster elements is obvious. This lines exhibit constant brightness along the length:



For any other orientation the choice is more difficult:



Rasterization of straight lines.



Rasterization yields uneven brightness: Horizontal and vertical lines appear brighter than the 45° lines.

For fixing so, we would need:
1. Calculation of square roots (increasing CPU time)
2. Multiple brigthness levels

=>

Compromise:
1. Calculate only an approximate line
2. Use integer arithmetic
3. Use incremental methods

The equation of a straight line is given by: $\boxed{y=m.x+b}$

Algorithm 1: Direct Scan Conversion

1. Start at the pixel for the left-hand endpoint x1
2. Step along the pixels horizontally until we reach the right-hand end of the line, xr
3. For each pixel compute the corresponding y value
4. round this value to the nearest integer to select the nearest pixel

```
x = xl;

while (x <= xr){

    ytrue = m*x + b;
    y = Round (ytrue);
    PlotPixel (x, y);
       * Set the pixel at (x,y) on *
    x = x − 1;

}
```

The algorithm performs a floating-point multiplication for every step in *x*. This method therefore requires an enormous number of floating-point multiplications, and is therefore expensive.

Algorithm 2: Digital Differential Analyzer (DDA)

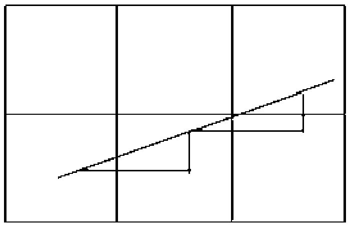
The differential equation of a straight line is given by:

$$\frac{dy}{dx} = constant \qquad \text{or} \qquad \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

The solution of the finite difference approximation is:

$$x_{i-1} = x_i + \Delta x$$

$$y_{i-1} = y_i + \frac{y_2 - y_1}{x_2 - x_1} \Delta y$$



DDA uses repeated addition

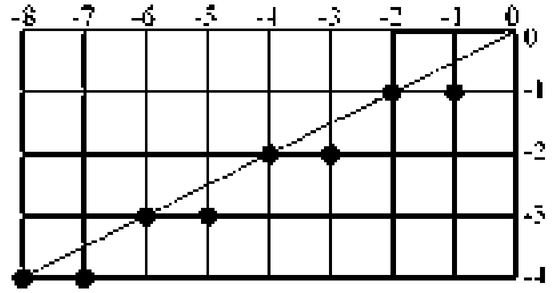
We need only compute *m* once, as the start of the scan-conversion.

The DDA algorithm runs rather slowly because it requires real arithmetic (floating-point operations).

DDA algorithm for lines with $-1 \le m \le 1$

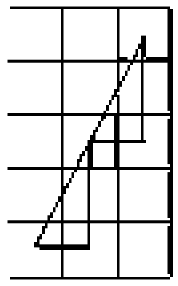
```
x = xl;
ytrue = yl;
while (x <= xr){
    ytrue = ytrue + m;
    y = Round (ytrue);
    PlotPixel (x, y);
    x = x + 1;
}
```

Example: Third quadrant

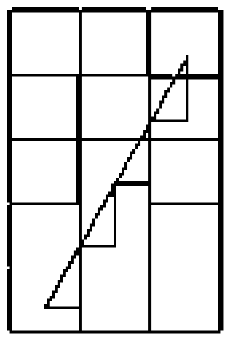


Switching the roles of *x* and *y* when $m > 1$

Gaps occur when $m > 1$



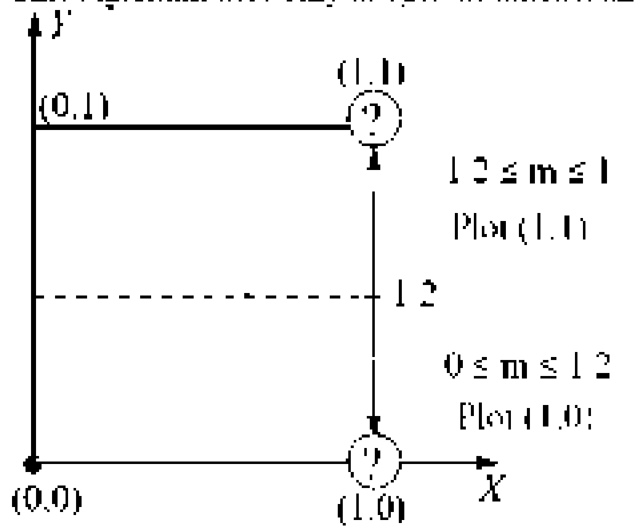
Reverse the roles of *x* and *y* using a unit step in *y*, and $1/m$ for *x*.



Algorithm 3: Bresenham's algorithm (1965)

Bresenham, J.E. ***Algorithm for computer control of a digital plotter*****, IBM Systems Journal, January 1965, pp. 25-30.**

This algorithm uses only integer arithmetic, **and runs significantly faster**



Key idea: **distance between the actual line and the nearest grid locations** (error).

Initialize error:

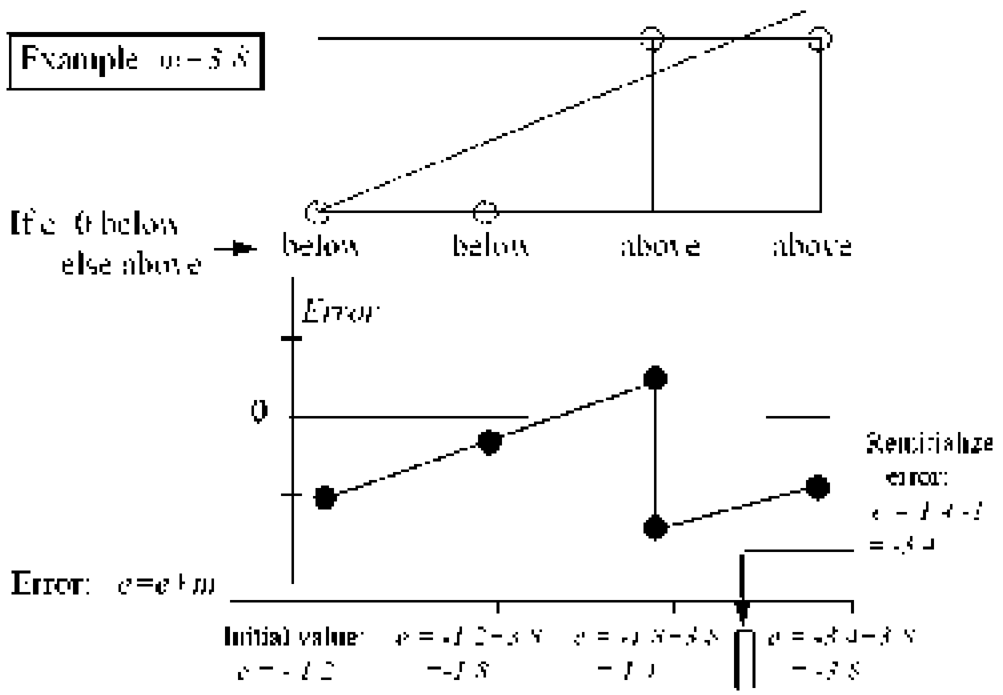
$e = -1/2$

Error is given by:

$e = e - m$

Reinitialize error

when $e > 0$



However, this algorithm does not lead to integer arithmetic. Scaling by: $2 * dx$

```
void Bresenham (int xl, int yl, int xr, int yr)
      {
        int x,y;                        * coordinates of pixel being drawn  *
        int dy, dx;
        int ne;                              * integer scaled error term     *
        x = xl; y = yl;                     * start at left endpoint          *
        ie = 2 * dy - dx;                     * initialize the error term *
          while (x <= xr){                         * pixel-drawing loop *
               PlotPixel (x,y);                         * draw the pixel   *
               if (ie > 0) {
                    y = y + 1;
                    ne = ne - 2 * dx;                   * replaces e := e - 1  *
               }
               x = x + 1;
              ne = ne + 2 * dy;                        * replaces e := e + m  *
          }
      }
```

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIT 2 - GEOMETRIC MODELING - SMR1402

UNIT - II

GEOMETRIC MODELING

PRE-REQUISITE DISCUSSION

2.1.CURVE REPRESENTATION

(1) Parametric equation x, y, z coordinates are related by a parametric variable (u or θ)

(2) Nonparametric equation x, y, z coordinates are related by a function

Example: Circle (2-D)

Parametric equation

$$x = R\cos\theta, \quad y = R\sin\theta \quad (0 \le \theta \le 2\pi)$$

Nonparametric equation

$$x^2 + y^2 - R^2 = 0$$ **(Implicit nonparametric form)**

$$y = \pm\sqrt{R^2 - x^2}$$ **(Explicit nonparametric form)**

TYPES OF CURVES USED IN GEOMETRIC MODELLING

- Hermite curves
- Bezeir curves
- B-spline curves
- NURBS curves

2.2.HERMITE CURVES

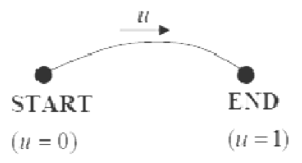
* Most of the equations for curves used in CAD software are of degree 3, because two curves of degree 3 guarantees 2nd derivative continuity at the connection point → The two curves appear to one.

* Use of a higher degree causes small oscillations in curve and requires heavy computation.

* Simplest parametric equation of degree 3

$$\mathbf{P}(u) = [x(u) \;\; y(u) \;\; z(u)]$$
$$= \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3 \qquad (0 \le u \le 1)$$

$\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$: Algebraic vector coefficients



➡ The curve's shape change cannot be intuitively anticipated from changes in these values

$$\mathbf{P}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3 \qquad (0 \le u \le 1)$$

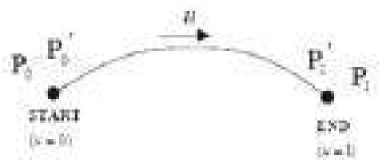
Instead of algebraic coefficients, let's use the position vectors and the tangent vectors at the two end points!

Position vector at starting point: $\mathbf{P}_0 = \mathbf{P}(0) = \mathbf{a}_0$

Position vector at end point: $\mathbf{P}_1 = \mathbf{P}(1) = \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3$

Tangent vector at starting point: $\mathbf{P}_0' = \mathbf{P}'(0) = \mathbf{a}_1$

Tangent vector at end point: $\mathbf{P}_1' = \mathbf{P}'(1) = \mathbf{a}_1 + 2\mathbf{a}_2 + 3\mathbf{a}_3$



Blending functions

$$\mathbf{P}(u) = \underbrace{[1-3u^2+2u^3 \quad 3u^2-2u^3 \quad u-2u^2+u^3 \quad -u^2+u^3]}_{\text{Blending functions}} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_0' \\ \mathbf{P}_1' \end{bmatrix}$$: Hermit curve

No algebraic coefficients

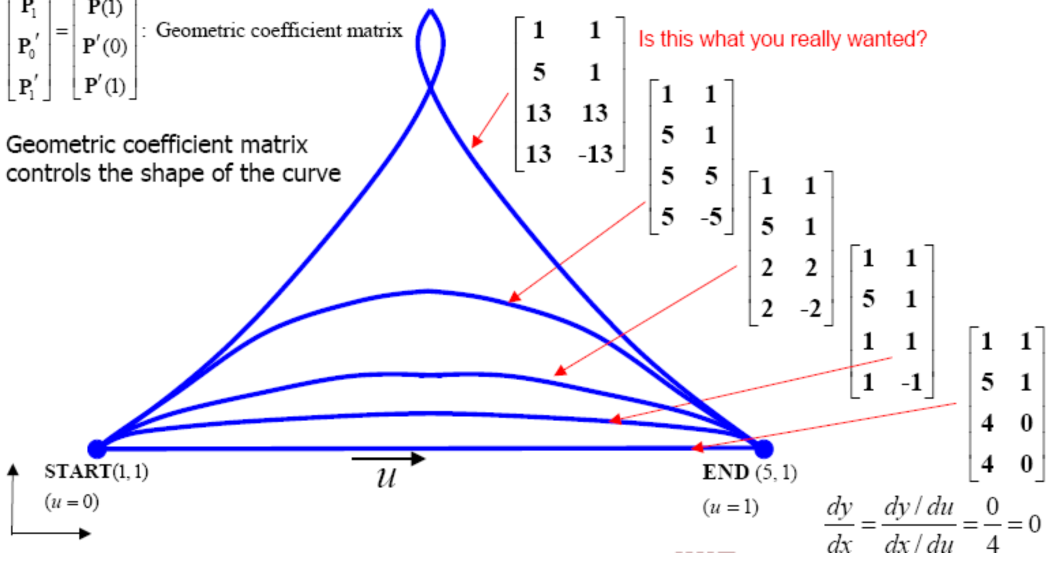
$\mathbf{P}_0, \mathbf{P}_0', \mathbf{P}_1, \mathbf{P}_1'$: Geometric coefficients

The curve's shape change can be intuitively anticipated from changes in these values

Effect of tangent vector on the curve's shape

$$\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_0' \\ \mathbf{P}_1' \end{bmatrix} = \begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}(1) \\ \mathbf{P}'(0) \\ \mathbf{P}'(1) \end{bmatrix}$$: Geometric coefficient matrix

Geometric coefficient matrix controls the shape of the curve



Is this what you really wanted?
$\begin{bmatrix} 1 & 1 \\ 5 & 1 \\ 13 & 13 \\ 13 & -13 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 \\ 5 & 1 \\ 5 & 5 \\ 5 & -5 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 \\ 5 & 1 \\ 2 & 2 \\ 2 & -2 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 \\ 5 & 1 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 \\ 5 & 1 \\ 4 & 0 \\ 4 & 0 \end{bmatrix}$
START(1, 1)
(u = 0)
u
END (5, 1)
(u = 1)
$\frac{dy}{dx} = \frac{dy/du}{dx/du} = \frac{0}{4} = 0$

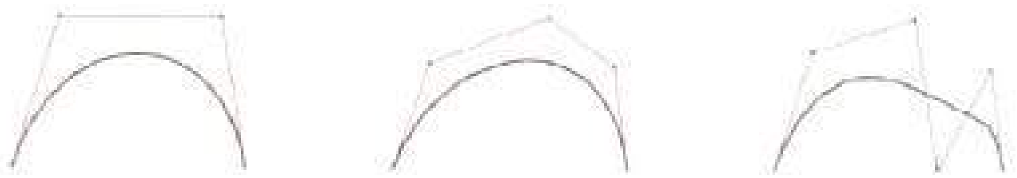
2.3.BEZIER CURVE

Properties

- The curve passes through the first and last vertex of the polygon.

-The tangent vector at the starting point of the curve has the same direction as the first segment of the polygon.

- The *n*th derivative of the curve at the starting or ending point is determined by the first or last (*n*+1) vertices.



Two Drawbacks of Bezier Curves

(1) For complicated shape representation, higher degree Bezier curves are needed.

→ Oscillation in curve occurs, and computational burden increases.

(2) Any one control point of the curve affects the shape of the entire curve.

→ Modifying the shape of a curve locally is difficult.

(Global modification property)

Desirable properties :

1. Ability to represent complicated shape with low order of the curve
2. Ability to modify a curve's shape locally

2.4.B-SPLINE CURVES

* Developed by Cox and Boor (1972)

$$\mathbf{P}(u) = \sum_{i=0}^{n} N_{i,k}(u)\mathbf{P}_i$$

where

$\mathbf{P}_i$: Position vector of the ith control point

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

$$N_{i,1}(u) = \begin{cases} 1 & t_i \le u \le t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

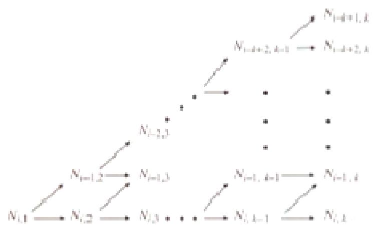
$$t_i = \begin{cases} 0 & 0 \le i < k \\ i-k+1 & k \le i \le n \\ n-k+2 & n < i \le n+k \end{cases}$$

(Nonperiodic knots)

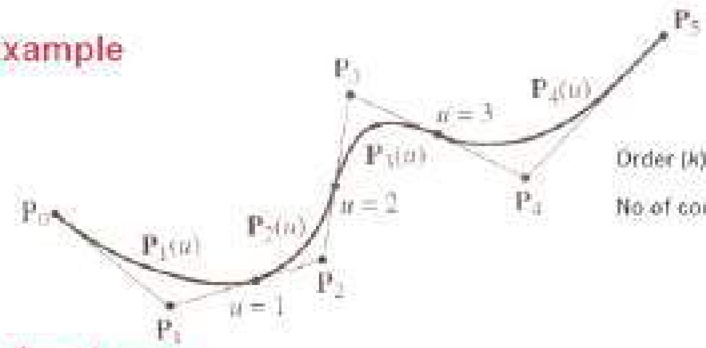
k: order of the B-spline curve

n+1: number of control points

The order of curve is independent of the number of control points!



Example



Order (k) = 3 (first derivatives are continuous)

No of control points (n+1) = 6

Advantages

(1) The order of the curve is independent of the number of control points (contrary to Bezier curves)

- User can select the curve's order and number of control points separately.
- It can represent very complicated shape with low order

(2) Modifying the shape of a curve locally is easy. (contrary to Bezier curve)

- Each curve segment is affected by k *(order)* control points. (local modification property)

2.5.NURBS curve

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{n} h_i \mathbf{P}_i N_{i,k}(u)}{\sum_{i=0}^{n} h_i N_{i,k}(u)} \qquad \left(\text{B-spline}: \ \mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(u)\right)$$

$\mathbf{P}_i$: Position vector of the ith control point

h_i : Homogeneous coordinate

* **If all the homogeneous coordinates (*h_i*) are 1, the denominator becomes 1**

If $h_i = 0\ \forall i$, then $\sum_{i=0}^{n} h_i N_{i,k}(u) = 1$.

* **B-spline curve is a special case of NURBS.**

* **Bezier curve is a special case of B-spline curve.**

Advantages of B-spline curves and NURBS curve

(1) More versatile modification capacity

- Homogeneous coordinate h_i, which B-spline does not have, can change.
- Increasing h_i of a control point → Drawing the curve toward the control point.

(2) NURBS can exactly represent the conic curves - circles, ellipses, parabolas, and hyperbolas (B-spline can only approximate these curves)

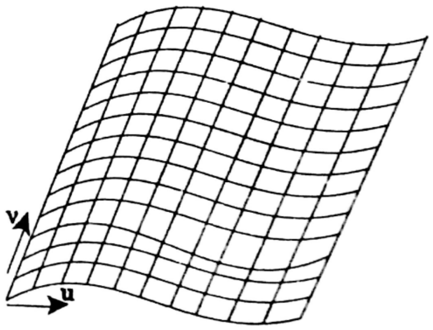
(3) Curves, such as conic curves, Bezier curves, and B-spline curves can be converted to their corresponding NURBS representations.

2.6.TECHNIQUES IN SURFACE MODELLING

i. Surface Patch
ii. Coons Patch
iii. Bicubic Patch
iv. Be'zier Surface
v. B-Spline Surface

i. Surface Patch

The patch is the fundamental building block for surfaces. The two variables *u* and *v* vary across the patch; the patch may be termed *biparametric.* The parametric variables often lie in the range 0 to 1. Fixing the value of one of the parametric variables results in a curve on the patch in terms of the other variable (*Isoperimetric curve*). Figure shows a surface with curves at intervals of *u* and *v* of 0 : 1.



Surface patch.

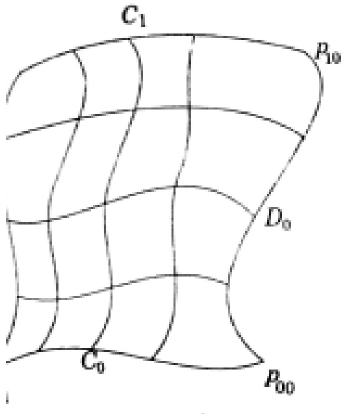
ii. Coons Patch

The sculptured surface often involve interpolation across an intersecting mesh of curves that in effect comprise a rectangular grid of patches, each bounded by four boundary curves. The linearly blended coons patch is the simplest for interpolating between such boundary curves. This patch definition technique blends for four boundary curves $C_i(u)$ and $D_j(v)$ and the corner points p_{ij} of the patch with the linear blending functions,

$$f(t) = 1 - t$$
$$g(t) = t$$

using the expression

$$\vec{p}(u, v) = \vec{C}_0(u)\, f(v) + \vec{C}_1(u)\, g(v) + \vec{D}_0(v)\, f(u) + \vec{D}_1(v)\, g(u)$$
$$- \vec{p}_{00}\, f(u)\, f(v) - \vec{p}_{01}\, f(u)\, g(u) - \vec{p}_{10}\, g(u)\, f(v) - \vec{p}_{11}\, g(u)\, g(v)$$



Coons patch

iii. Bicubic Patch

The bi-cubic patch is used for surface descriptions defined in terms of point and tangent vector information. The general form of the expressions for a bi-cubic patch is given by:

$$\vec{p}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} \vec{k_{ij}}\, u^i\, v^j$$

This is a vector equation with 16 unknown parameters k_{ij} which can be found by Lagrange interpolation through 4 x 4 grid.

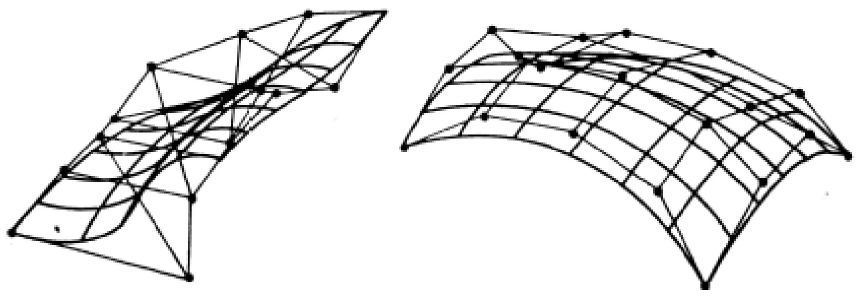
iv. Be'zier Surface

- The Be'zier surface formulation use a characteristic polygon
- Points the Bezier surface are given by

$$\vec{p}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \vec{B}_{i,m}(u)\, \vec{B}_{j,n}\, \vec{p}_{ij};\ u, v, \epsilon\ [0, 1]$$

Where,

$\vec{p}_{ij};$	-	Vertices of the characteristic polygon
$\vec{B}_{j,n}$, $\vec{B}_{i,n}$	-	Blending functions



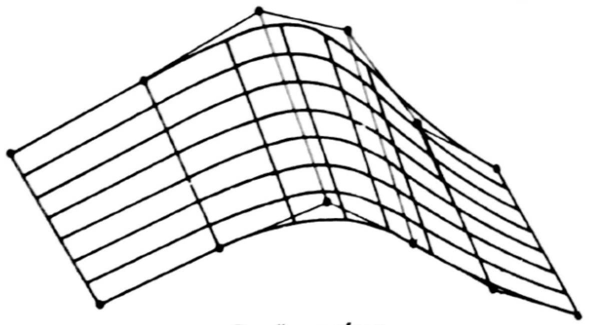
v. B-Spline Surfaces

- The B-spline surface approximates a characteristics polygon as shown and passes through the corner points of the polygon, where its edges are tangential to the edges of the polygon
- This may not happen when the control polygon is closed
- A control point of the surface influences the surface only over a limited portion of the parametric space of variables u and v.

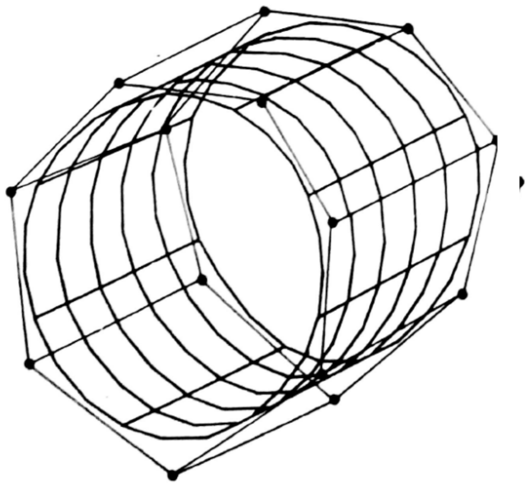
The expression for the B-spline surfaces is given by

$$\vec{p}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,k}(u)\, N_{j,l}(v)\, \vec{p_{ij}}$$

p_{ij} **are the vertices of the defining polygon and** $N_{i,k}$ **and** $N_{j,l}$ **are blending functions.**



. B-spline surface



B-spline surface for a closed control polygon

2.7.GEOMETRIC MODELLING

Geometric modeling is the starting point of the product design and manufacture process. Functions of Geometric Modeling are:

Design Analysis

- Evaluation of area, volume, mass and inertia properties
- Interference checking in assemblies
- Analysis of tolerance build-up in assemblies
- Kinematic analysis of mechanisms and robots
- Automatic mesh generation for finite element analysis

Drafting

- Automatic planar cross-sectioning
- Automatic hidden lines and surface removal
- Automatic production of shaded images
- Automatic dimensioning
- Automatic creation of exploded views of assemblies

Manufacturing

- Parts classification
- Process planning
- NC data generation and verification
- Robot program generation

Production Engineering

- Bill of materials
- Material requirement
- Manufacturing resource requirement
- Scheduling

Inspection and quality control

- Program generation for inspection machines
- Comparison of produced parts with design

2.8.PROPERTIES OF A GEOMETRIC MODELING SYSTEM

- The geometric model must stay invariant with regard to its location and orientation
- The solid must have an interior and must not have isolated parts
- The solid must be finite and occupy only a finite shape
- The application of a transformation or Boolean operation must produce another solid
- The solid must have a finite number of surfaces which can be described
- The boundary of the solid must not be ambiguous

2.9.WIRE FRAME MODELING

It uses networks of interconnected lines (wires) to represent the edges of the physical objects being modeled

Also called 'Edge-vertex' or 'stick-figure' models

Two types of wire frame modeling:

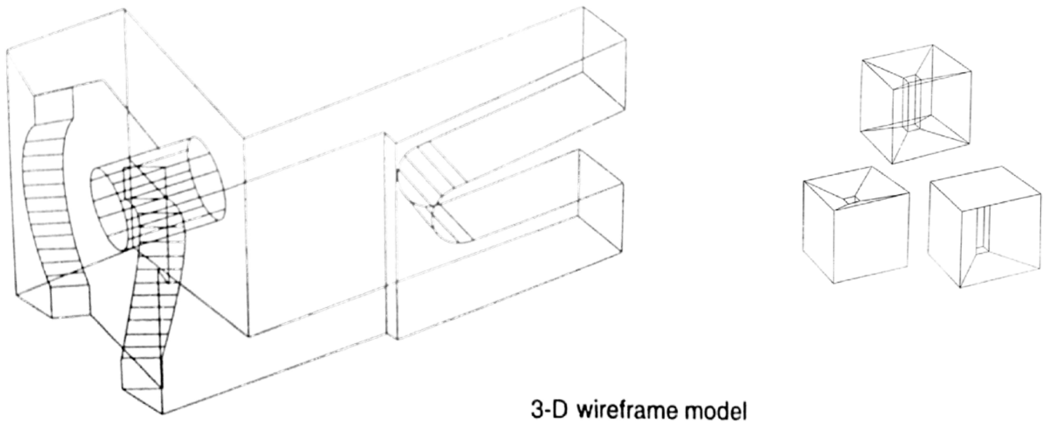
1. 2 ½ - D modeling
2. 3 – D modeling

3-D Wire frame models: These are

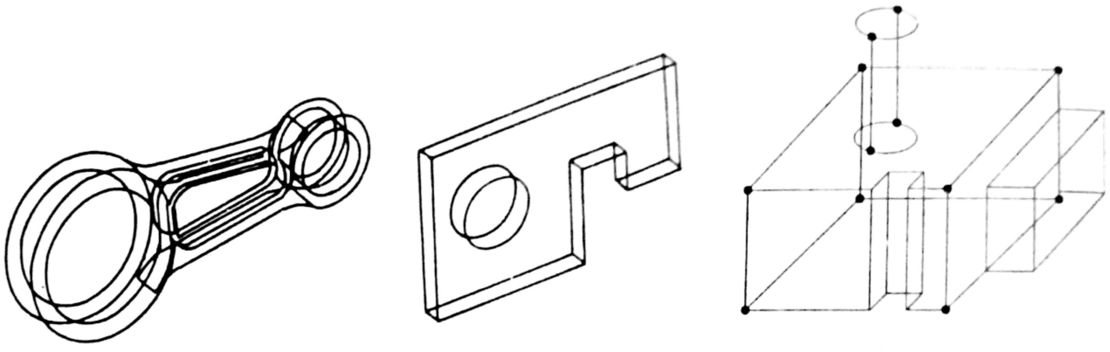
Simple and easy to create, and they require relatively little computer time and memory; however they do not give a complete description of the part. They contain little information about the surface and volume of the part and cannot distinguish the inside from the outside of part surfaces. They are visually ambiguous as the model can be interpreted in many different ways because in many wire frame models hidden lines cannot be removed. Section property and mass calculations are impossible, since the object has no faces attached to it. It has limited values a basis for manufacture and analysis

2 ½ - D Wire frame models:

Two classes of shape for which a simple wire-frame representation is often adequate are those shapes defined by projecting a plane profile along its normal or by rotating a planar profile about an axis. Such shapes are not two-dimensional, but neither do they require sophisticated three-dimensional schemes for their representation. Such representation is called 2 ½ - D.



3-D wireframe model


2½ D wire frame model

2.10. TECHNIQUES IN SURFACE MODELLING

The various methods for representing the solids are:

1. Half-space method
2. Boundary representation method (B-rep)
3. Constructive solid geometry (CSG and C-rep)
4. Sweep representation
5. Analytical solid modeling (ASM)
6. Primitive instancing
7. Spatial partitioning representation
   a. Cell decomposition
   b. Spatial occupancy enumeration
   c. Octree encoding

2.11. Boundary representation method (B-rep)

- In solid modeling and computer-aided design, boundary representation often abbreviated as B-rep or BREP—is a method for representing shapes using the limits.

- A solid is represented as a collection of connected surface elements, the boundary between solid and non-solid.

- Boundary representation models are composed of two parts:

  - Topology, and
  - Geometry (surfaces, curves and points).

- The main ***topological*** items / primitives of b-rep are:

  - Vertex (V) : It is a unique point (an ordered triplet) in space
  - Edge (E) : It is finite, non-self intersecting, directed space curve bounded by

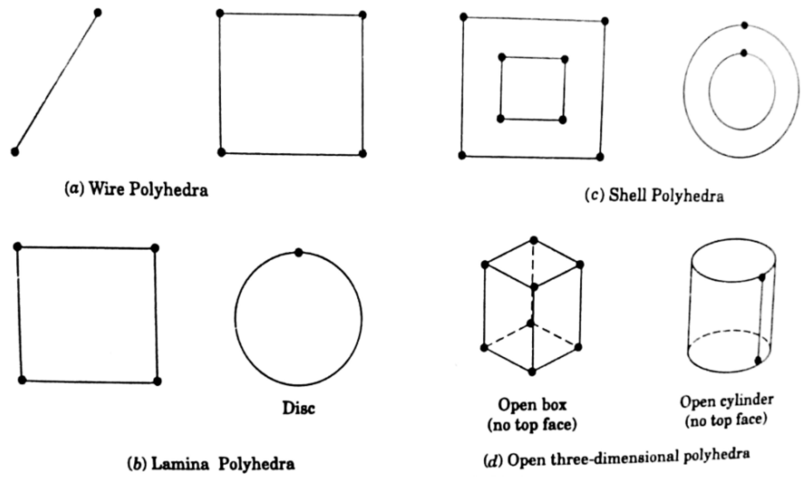
two vertices that are not necessarily distinct

- Face (F) : It is defined as a finite connected, non-self-intersecting, region of a closed oriented surface bounded by one or more loops
- Loop (L) : It is an ordered alternating sequence of vertices and edges
- Genus(G) : It is the topological name for the number of handles or through holes in an object
- Body/Shell(B) : It is a set of faces that bound a single connected closed volume. A minimum body is a point

- A minimum body is a point; topologically this body has one face, one vertex, and no edges. It is called a seminal or singular body

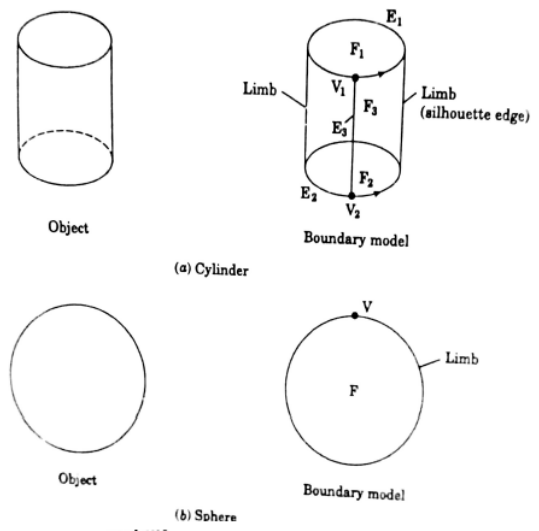
- *Geometry*

Open polyhedral objects



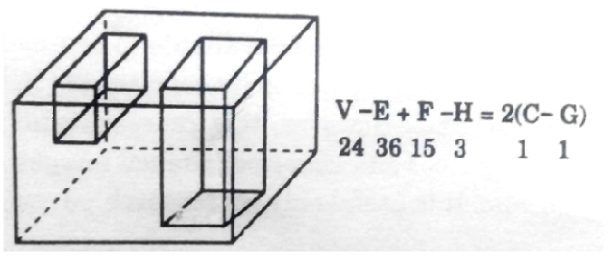
Open polyhedral objects.

Curved Objects

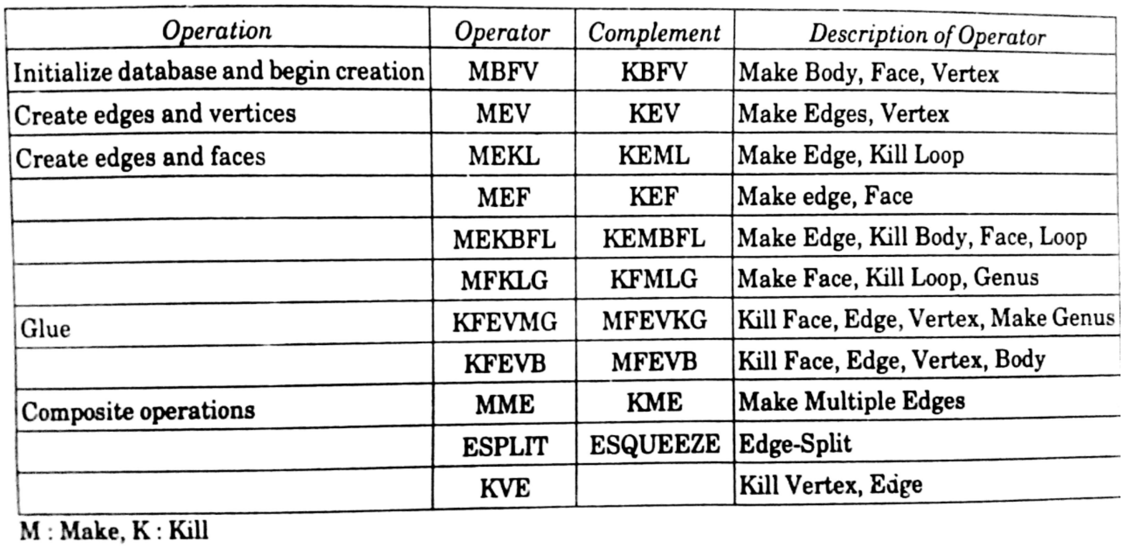


Exact B-rep of a cylinder and a sphere.

- **Euler's formula**
  - Euler – Poincare Law for closed objects : $F - E + V - L = 2 (B - G)$
  - Euler – Poincare Law for open objects : $F - E + V - L = B - G$



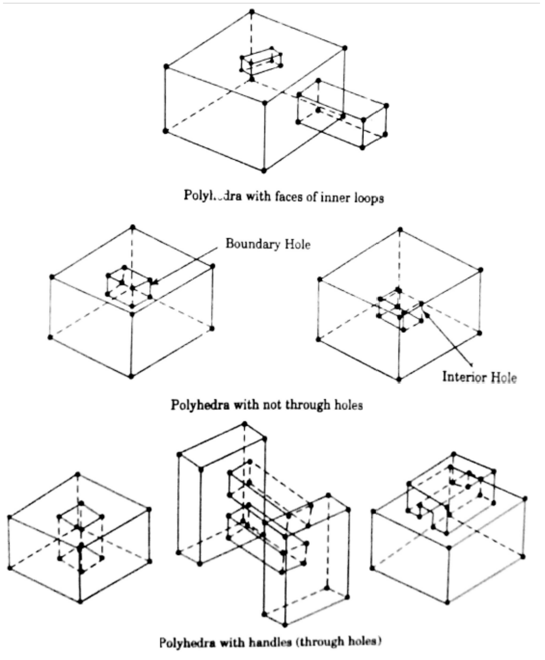
Some Euler Operations



Operation	*Operator*	*Complement*	*Description of Operator*
Initialize database and begin creation	MBFV	KBFV	Make Body, Face, Vertex
Create edges and vertices	MEV	KEV	Make Edges, Vertex
Create edges and faces	MEKL	KEML	Make Edge, Kill Loop
	MEF	KEF	Make edge, Face
	MEKBFL	KEMBFL	Make Edge, Kill Body, Face, Loop
	MFKLG	KFMLG	Make Face, Kill Loop, Genus
Glue	KFEVMG	MFEVKG	Kill Face, Edge, Vertex, Make Genus
	KFEVB	MFEVB	Kill Face, Edge, Vertex, Body
Composite operations	MME	KME	Make Multiple Edges
	ESPLIT	ESQUEEZE	Edge-Split
	KVE		Kill Vertex, Edge

M : Make, K : Kill

Solid Model Generation using B-rep



Polyhedra with faces of inner loops

Polyhedra with not through holes

Polyhedra with handles (through holes)

Advantages of b-rep

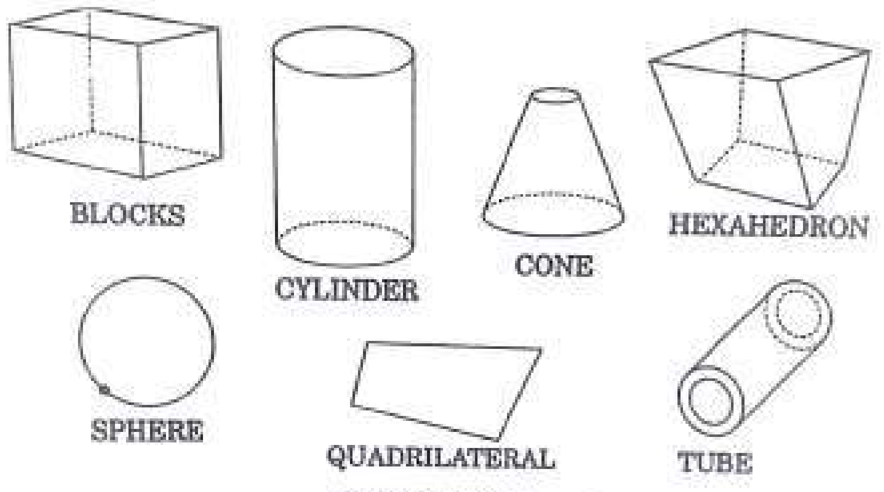
- Appropriate to construct solid models of unusual shapes
- Relatively simple to convert a b-rep model to wireframe model

Disadvantages of b-rep

- Requires more storage
- Not suitable for applications like tool path generation
- Slow manipulation

2.12. CONSTRUCTIVE SOLID GEOMETRY (CSG and C-rep)

- Constructive solid geometry (CSG) (formerly called computational binary solid geometry) is a technique used in solid modeling.
- Constructive solid geometry allows a modeler to create a complex surface or object by using Boolean operators to combine objects.
- Often CSG presents a model or surface that appears visually complex, but is actually little more than cleverly combined or de-combined objects
- The simplest solid objects used for the representation are called **primitives**. Typically they are the objects of simple shape:
  - cuboids
  - cylinders
  - prisms
  - pyramids
  - spheres
  - cones

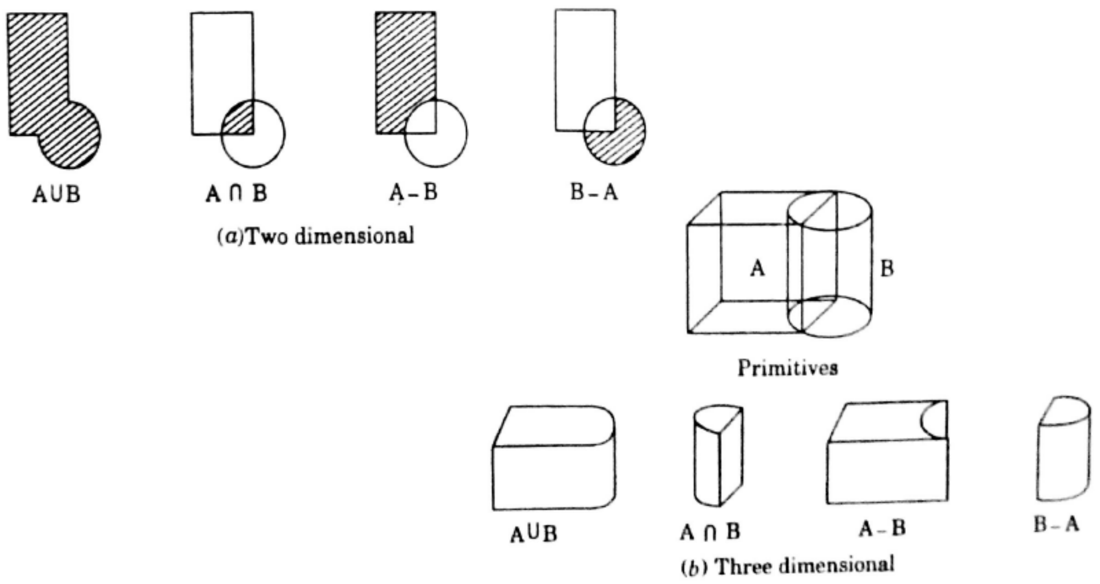


Solid primitives.

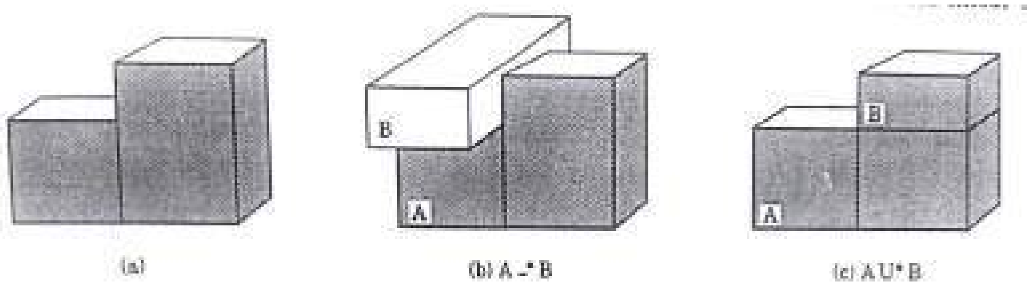
- The set of allowable primitives is limited by each software package. Some software packages allow CSG on curved objects while other packages do not

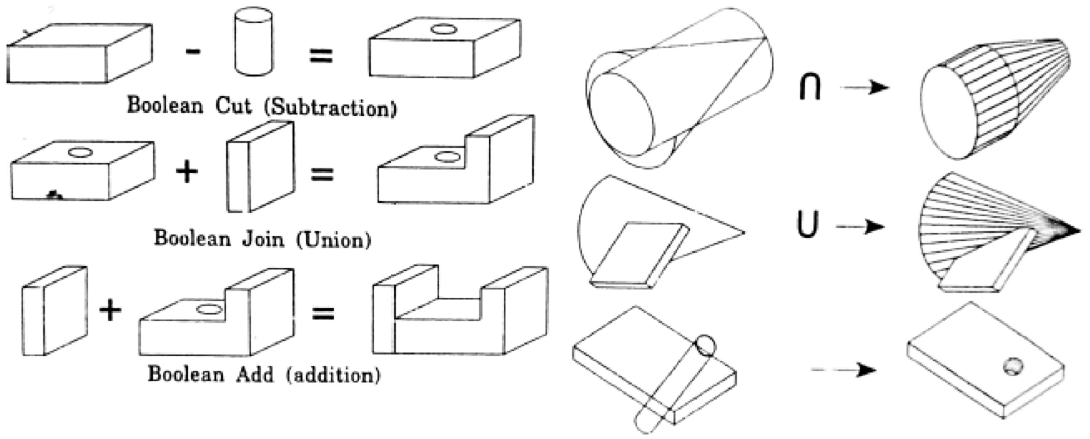
- It is said that an object is **constructed** from primitives by means of allowable **operations**, which are typically Boolean operations on sets: *union, intersection and difference, as well as geometric transformations* of those sets

Boolean Operations

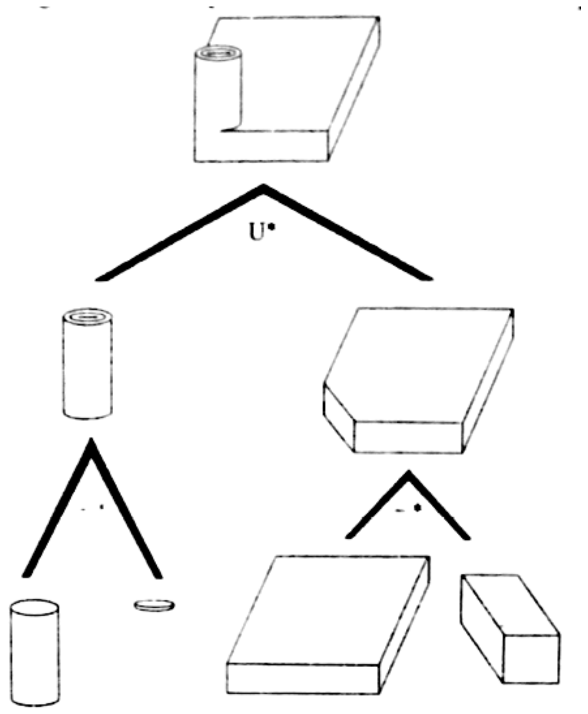


Boolean operations of a block *A* and cylinder *B*.





CSG Tree



An object defined by CSG and its tree.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIT 3 - VISUAL REALISM - SMR1402

UNIT - III

VISUAL REALISM

3.1. PRE-REQUISITE DISCUSSION

In 3D computer graphics, hidden surface determination (also known as hidden surface removal (HSR), occlusion culling (OC) or visible surface determination (VSD)) is the process used to determine which surfaces and parts of surfaces are not visible from a certain viewpoint. A hidden surface determination algorithm is a solution to the visibility problem, which was one of the first major problems in the field of 3D computer graphics. The process of hidden surface determination is sometimes called hiding, and such an algorithm is sometimes called a hider. The analogue for line rendering is hidden line removal. Hidden surface determination is necessary to render an image correctly, so that one cannot look through walls in virtual reality.

3.2. HIDDEN SURFACE REMOVAL (HSR) AND ITS ALGORITHMS

Hidden surface determination is a process by which surfaces which should not be visible to the user (for example, because they lie behind opaque objects such as walls) are prevented from being rendered. Despite advances in hardware capability there is still a need for advanced rendering algorithms. The responsibility of a rendering engine is to allow for large world spaces and as the world's size approaches infinity the engine should not slow down but remain at constant speed. Optimizing this process relies on being able to ensure the deployment of as few resources as possible towards the rendering of surfaces that will not end up being rendered to the user.

There are many techniques for hidden surface determination. They are fundamentally an exercise in sorting, and usually vary in the order in which the sort is performed and how the problem is subdivided. Sorting large quantities of graphics primitives is usually done by divide and conquer.

Hidden surface removal algorithms

Considering the rendering pipeline, the projection, the clipping, and the rasterization steps are handled differently by the following algorithms:

Z-buffering :

During rasterization the depth/Z value of each pixel (or *sample* in the case of anti-aliasing, but without loss of generality the term *pixel* is used) is checked against an existing depth value. If the current pixel is behind the pixel in the Z-buffer, the pixel is rejected, otherwise it is shaded and its depth value replaces the one in the Z-buffer. Z-buffering supports dynamic scenes easily, and is currently implemented efficiently in graphics hardware. This is the current standard. The cost of using Z-buffering is that it uses up to 4 bytes per pixel, and that the rasterization algorithm needs to check each rasterized sample against the z-buffer. The z-buffer can also suffer from artifacts due to precision errors (also known as z-fighting), although this is far less common now that commodity hardware supports 24-bit and higher precision buffers.

Coverage buffers (C-Buffer) and Surface buffer (S-Buffer):

faster than z-buffers and commonly used in games in the Quake I era. Instead of storing the Z value per pixel, they store list of already displayed segments per line of the screen. New polygons are then cut against already displayed segments that would hide them. An S-Buffer can display unsorted polygons, while a C-Buffer requires polygons to be displayed from the nearest to the furthest. Because the C-buffer technique does not require a pixel to be drawn more than once, the process is slightly faster. This was commonly used with BSP trees, which would provide sorting for the polygons.

Sorted Active Edge List

It is used in Quake 1, this was storing a list of the edges of already displayed polygons. Polygons are displayed from the nearest to the furthest. New polygons are clipped against already displayed polygons' edges, creating new polygons to display then storing the additional edges. It's much harder to implement than S/C/Z buffers, but it will scale much better with the increase in resolution.

Painter's algorithm

It sorts polygons by their bary center and draws them back to front. This produces few artifacts when applied to scenes with polygons of similar size forming smooth meshes and back face culling turned on. The cost here is the sorting step and the fact that visual artifacts can occur.

Binary space partitioning (BSP)

It divides a scene along planes corresponding to polygon boundaries. The subdivision is constructed in such a way as to provide an unambiguous depth ordering from any point in the scene when the BSP tree is traversed. The disadvantage here is that the BSP tree is created with an expensive pre-process. This means that it is less suitable for scenes consisting of dynamic geometry. The advantage is that the data is pre-sorted and error free, ready for the previously mentioned algorithms. Note that the BSP is not a solution to HSR, only an aid.

Ray tracing

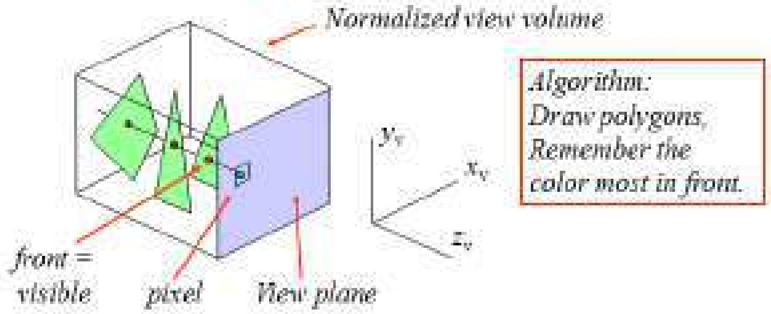
Attempt to model the path of light rays to a viewpoint by tracing rays from the viewpoint into the scene. Although not a hidden surface removal algorithm as such, it implicitly solves the hidden surface removal problem by finding the nearest surface along each view-ray. Effectively this is equivalent to sorting all the geometry on a per pixel basis.

The Warnock algorithm

It divides the screen into smaller areas and sorts triangles within these. If there is ambiguity (i.e., polygons overlap in depth extent within these areas), then further subdivision occurs. At the limit, subdivision may occur down to the pixel level.

3.3.DEPTH-BUFFER ALGORITHM

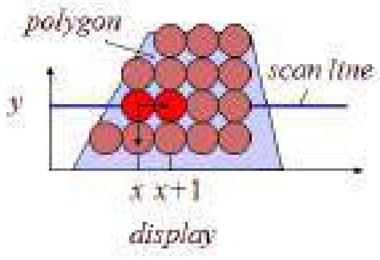
- Image-space method
- Aka *z-buffer algorithm*



```
var zbuf: array[N,N] of real;          { z-buffer: 0=near, 1=far }
    fbuf: array[N,N] of color;         { frame-buffer }

For all 1<= i, j <=N do
      zbuf[i,j] := 1.0; col[i,j] := BackgroundColour;
For all polygons do                  { scan conversion }
   For all covered pixels (i,j) do
      Calculate depth z;
      If z < zbuf[i,j] then { closer! }
         zbuf[i,j] := z;
         fbuf[i,j] := surfacecolor(i,j);
```

Fast calculation z: use coherence.



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

Hence: $z(x,y) = \frac{-Ax - By - D}{C}$

Also: $z(x+1,y) = \frac{-A(x+1) - By - D}{C}$

Thus: $z(x+1,y) = z(x,y) - \frac{A}{C}$

Also: $z(x,y) = z(x,y-1) + \frac{B}{C}$

Advantages

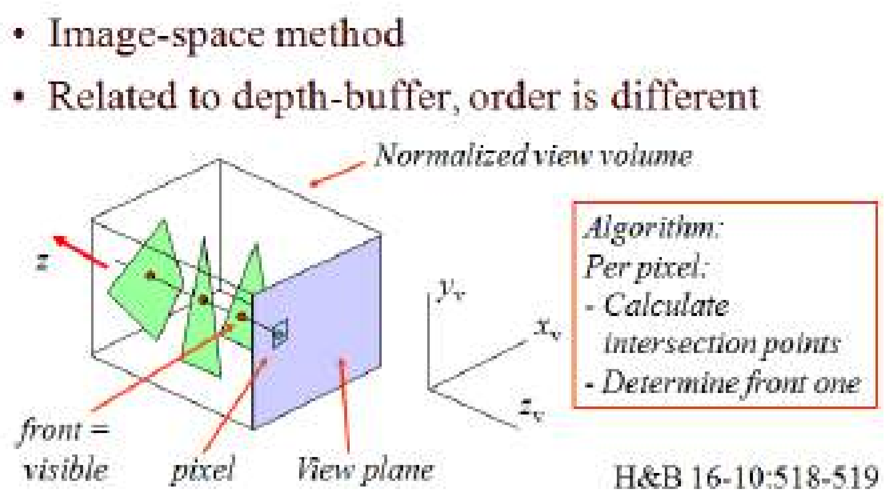
- Easy to implement
- Hardware supported
- Polygons can be processed in arbitrary order-
- Fast: ~ #polygons, #covered pixels

Disadvantages

- Costs memory
- Color calculation sometimes done multiple times
- Transparancy is tricky

3.4. RAY-CASTING ALGORITHM IN HIDDEN SURFACE REMOVAL

- Image-space method
- Related to depth-buffer, order is different

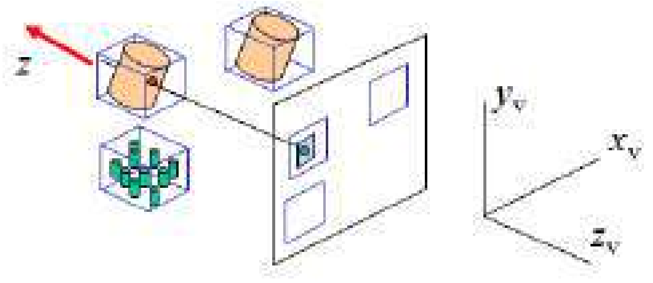


- Image-space method
- Related to depth-buffer, order is different

```
Var  fbuf: array[N,N] of colour;          { frame-buffer     }
     n : integer;                         { #intersections   }
     z : array[MaxIntsec] of real;        { intersections    }
     p : array[MaxIntsec] of object;      { corresp. objects }
For all 1<= i, j <=N do  { for alle pixels }
   For all objects do
      Calculate intersections and add these to z and p,
      keeping z and p sorted;
   if n > 1 then fbuf[i,j] := surfacecolor(p[1], z[1]); cc
```

Acceleration intersection calculations: Use (hierarchical) bounding boxes

Acceleration intersection calculations:
Use (hierarchical) bounding boxes



Advantages

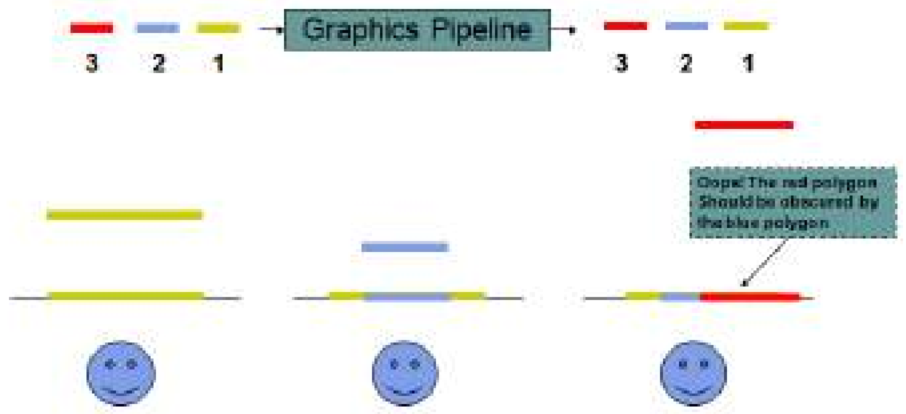
+ Relatively easy to implement
+ For some objects very suitable (for instance spheres and other quadratic surfaces)
+ Transparency can be dealt with easily

Disadvantages

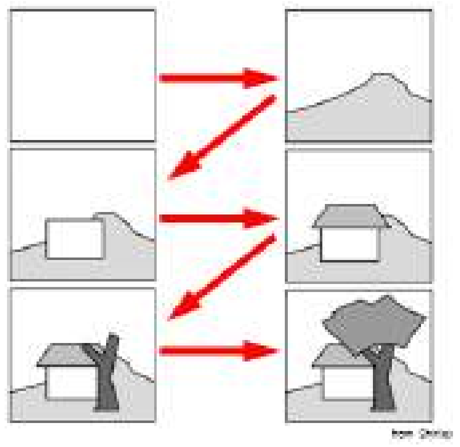
- Objects must be known in advance
- Slow: ~ #objects*pixels, little coherence

3.5. PAINTER'S ALGORITHM

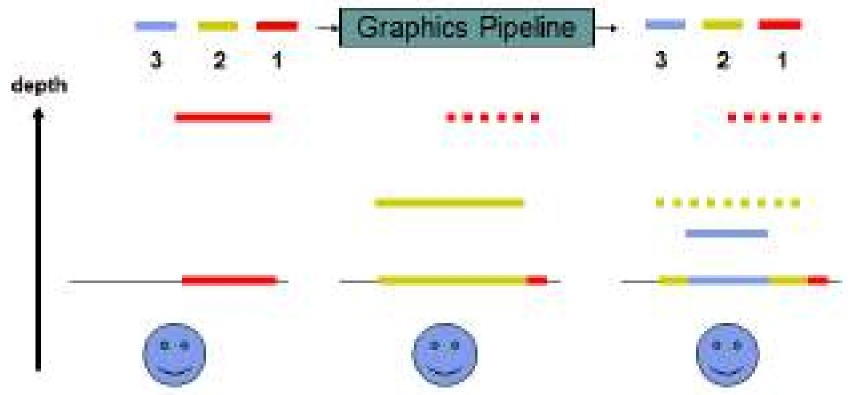
- Assumption: Later projected polygons overwrite earlier projected polygons


3
2
1
Graphics Pipeline
3
2
1
Oops! The red polygon Should be obscured by the blue polygon

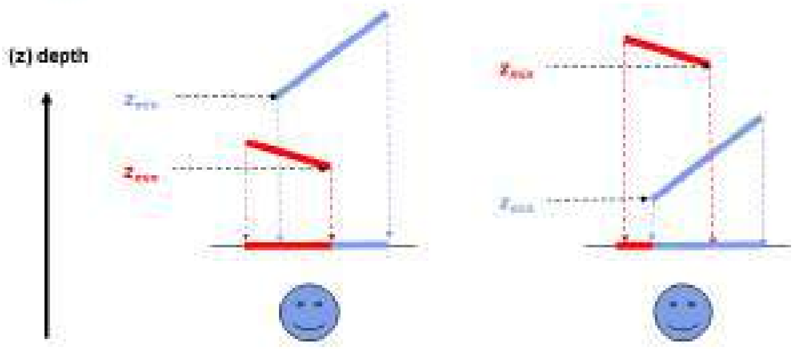
- Main Idea
  - A painter creates a picture by drawing background scene elemens before foreground ones
- Requirements
  - Draw polygons in back-to-front order
  - Need to **sort** the polygons by depth order to get a correct image



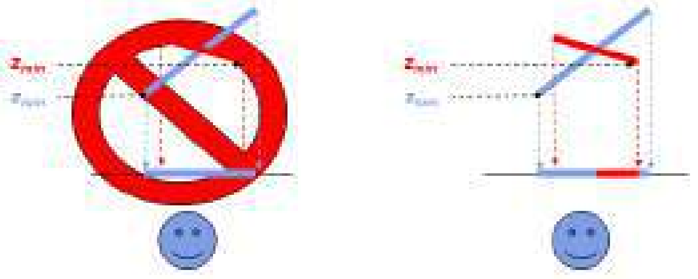
- Sort by the depth of each polygon



- Compute z_{min} ranges for each polygon
- Project polygons with furthest z_{min} first

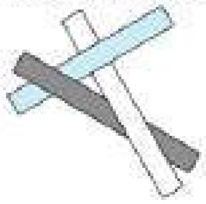


- Problem: Can you get a <u>total</u> sorting?



Correct?

- Cyclic Overlap
  - How do we sort these three polygons?



- Sorting is nontrivial
  - Split polygons in order to get a total ordering
  - Not easy to do in general

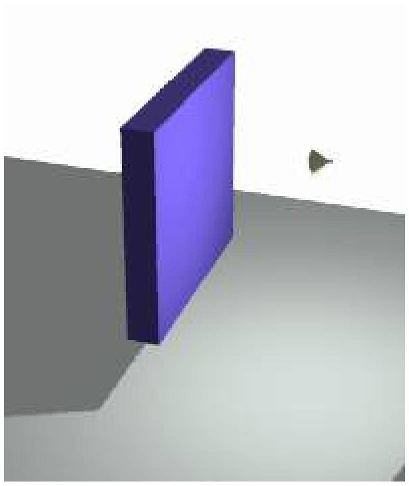
3.6. LIGHTING

Shading is also dependent on the lighting used. Usually, upon rendering a scene a number of different lighting techniques will be used to make the rendering look more realistic. Different types of light sources are used to give different effects.

Ambient lighting

An ambient light source repesents a fixed-intensity and fixed-color light source that affects all objects in the scene equally. Upon rendering, all objects in the scene are brightened with the specified intensity and color. This type of light source is mainly used to provide the scene with a basic view of the different objects in it. This is the simplest type of lighting to implement and models how light can be scattered or reflected many times producing a uniform effect.

Ambient lighting can be combined with ambient occlusion to represent how exposed each point of the scene is, affecting the amount of ambient light it can reflect. This produces diffuse, non-directional lighting throughout the scene, casting no clear shadows, but with enclosed and sheltered areas darkened. The result is usually visually similar to an overcast day.



Directional lighting

A directional light source illuminates all objects equally from a given direction, like an area light of infinite size and infinite distance from the scene; there is shading, but cannot be any distance falloff.

Point lighting

Light originates from a single point, and spreads outward in all directions.

Spotlight lighting

Models a Spotlight. Light originates from a single point, and spreads outward in a cone.

Area lighting

Light originates from a small area on a single plane. A more accurate model than a point light source.

Volumetric lighting

Light originating from a small volume, an enclosed space lighting objects within that space.

Shading is interpolated based on how the angle of these light sources reach the objects within a scene. Of course, these light sources can be and often are combined in a scene.

The renderer then interpolates how these lights must be combined, and produces a 2d image to be displayed on the screen accordingly.

3.7. SMOOTH SHADING

In contrast to flat shading with smooth shading the color changes from pixel to pixel. It assumes that the surfaces are curved and uses interpolation techniques to calculate the values of pixels between the vertices of the polygons.

Types of smooth shading include:

- Gouraud shading
- Phong shading

Gouraud shading

1. Determine the normal at each polygon vertex
2. Apply an illumination model to each vertex to calculate the vertex intensity
3. Interpolate the vertex intensities using bilinear interpolation over the surface polygon

Data structures

- *Sometimes vertex normals can be computed directly (e.g. height field with uniform mesh)*

- More generally, need data structure for mesh
- Key: which polygons meet at each vertex

Advantages

Polygons, more complex than triangles, can also have different colors specified for each vertex. In these instances, the underlying logic for shading can become more intricate.

Problems

- Even the smoothness introduced by Gouraud shading may not prevent the appearance of the shading differences between adjacent polygons.
- Gouraud shading is more CPU intensive and can become a problem when rendering real time environments with many polygons.
- T-Junctions with adjoining polygons can sometimes result in visual anomalies. In general, T-Junctions should be avoided.

Phong shading

Phong shading is similar to Gouraud shading, except that the Normal's are interpolated. Thus, the specular highlights are computed much more precisely than in the Gouraud shading model:

a. Compute a normal N for each vertex of the polygon.
b. From bilinear interpolation compute a normal, Ni for each pixel. (This must be renormalized each time)
c. From Ni compute an intensity Ii for each pixel of the polygon.
d. Paint pixel to shade corresponding to light.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIT 4 - ASSEMBLY OF PARTS - SMR1402

UNIT - IV

ASSEMBLY OF PARTS

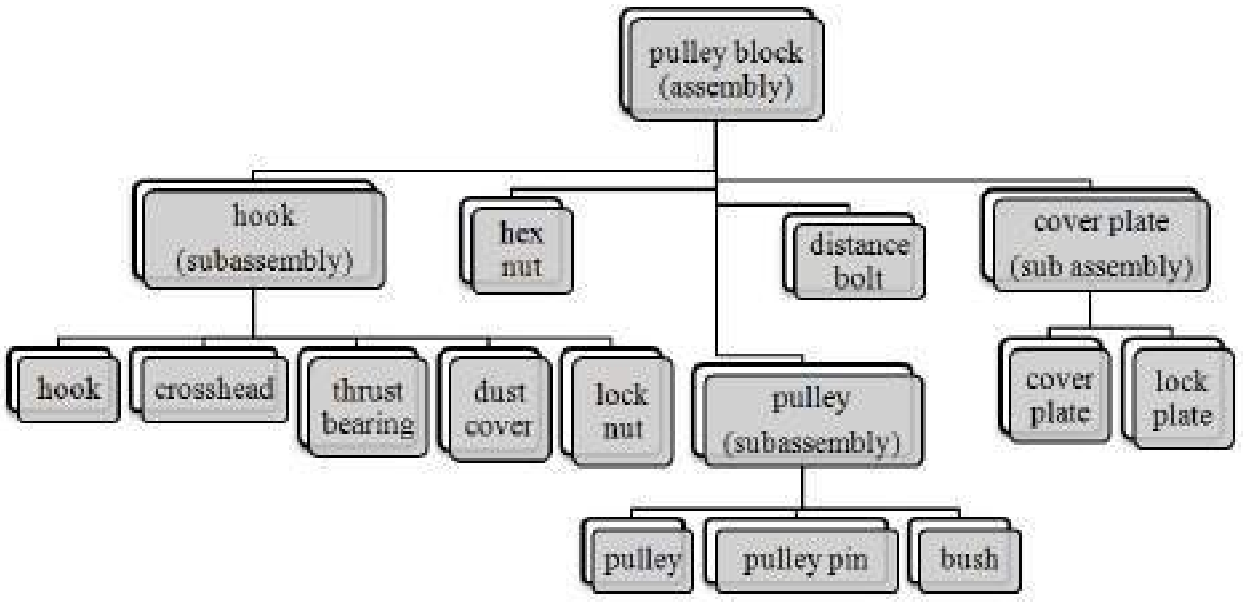
4.1. PRE-REQUISITE DISCUSSION

Assembly modeling is a technology and method used by computer-aided design and product visualization computer software systems to handle multiple files that represent components within a product. The components within an assembly are represented as solid or surface models.

4.2. ASSEMBLY MODELING OF PARTS

- Assembly modeling is a combination of two or more components using parametric relationships.
- Typically a designer would start with a base part
- Add other components to the **base part** using merge commands.

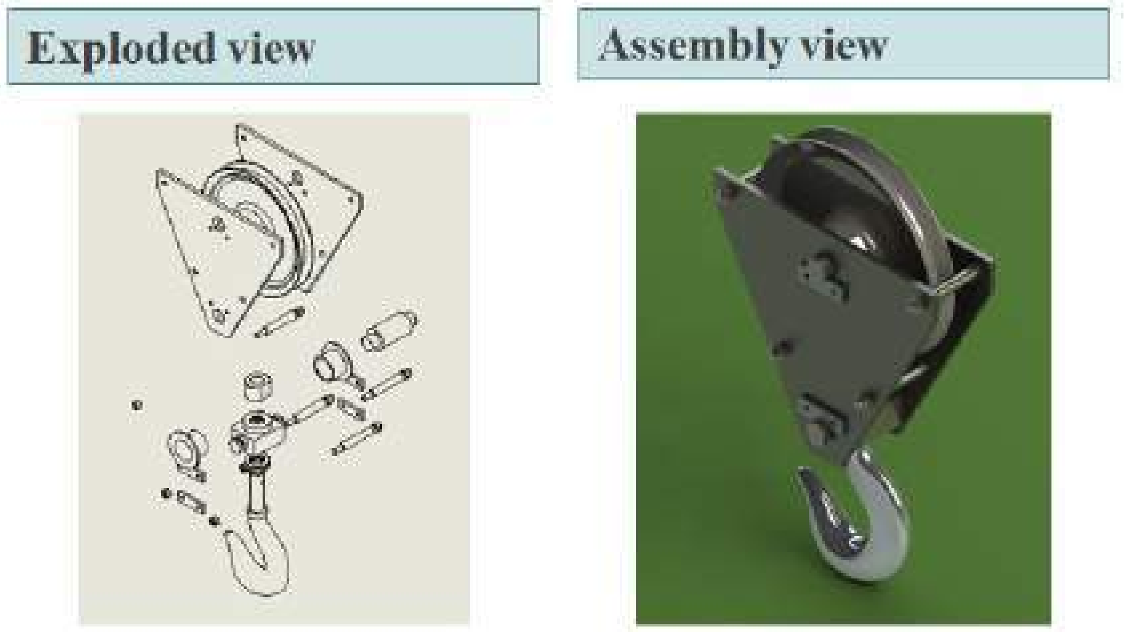
Assembly Tree



Exploded view

An exploded view consists of series of steps. One can create steps by selecting and dragging parts in graphical area.

Example - Assembly of Pulley block



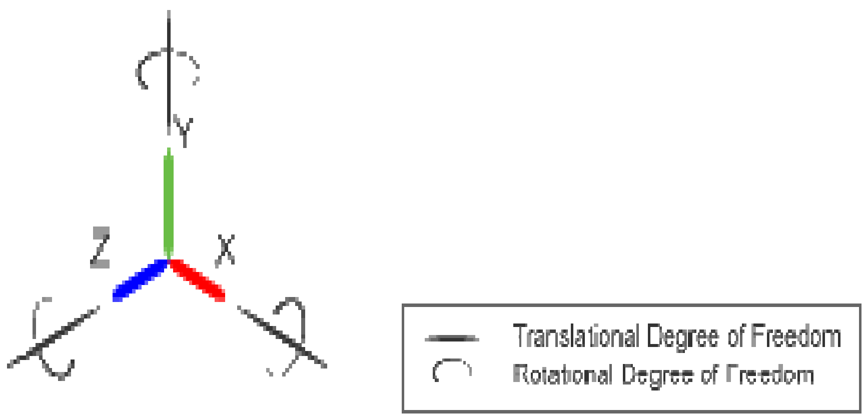
Bottom-up assembly approach -:

- Allows the designer to use part drawings that already exist (off the shelf)
- Provides the designer with more control over individual parts
- Multiple copies (instances) of parts can be inserted into the assembly

Top-down assembly approach -:

- The approach is ideal for large assemblies consisting of thousands of parts.
- The approach is used to deal with large designs including multiple design teams.
- It lends itself well to the conceptual design phase
- E.g. :
    - Piping and fittings
    - Welds
    - Lock pins

Degrees of freedom -:



- **Translation –** movement along X, Y, and Z axis

- **Rotation –** rotate about X, Y, and Z axis

Mating conditions -:

Basic mates

- Coincidence
- Parallel
- Perpendicular
- Tangent
- Concentric
- Lock
- Angle
- Distance

Advanced Mates

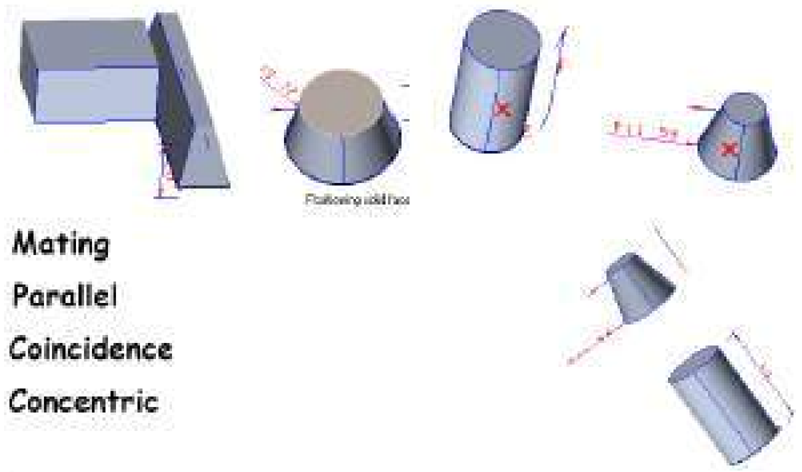
- Distance
- Linear/linear coplanar
- Path
- Width
- Symmetry
- Angle

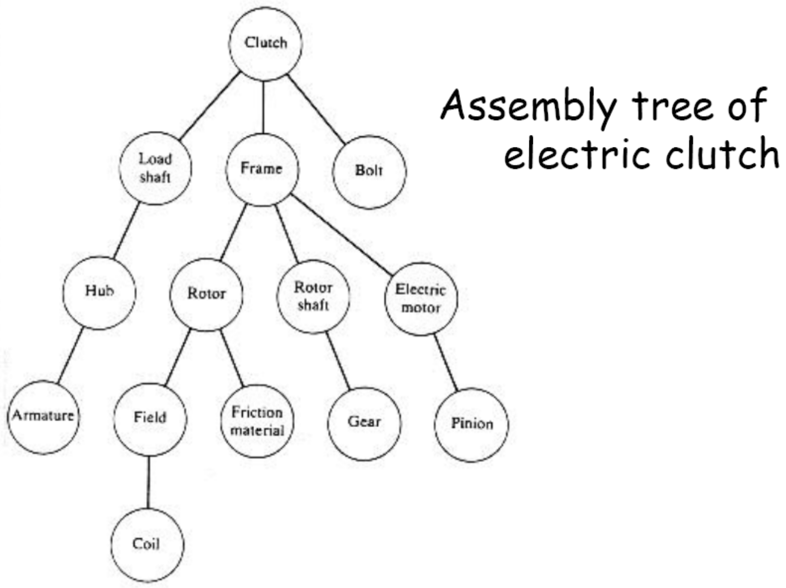
Mechanical mates:

- Cam
- Hinge
- Gear
- Rack Pinion
- Screw
- Universal Joint

Assembly Constraints

- Constraints can be used to create permanent relationships between parts
- THEY use the same commands as 2D constraints
- Typical constraints:
    - two faces meet
    - axes coincident
    - two faces parallel at fixed distance





Assembly sequence affects

- difficulty of assembly steps
- need for fixture
- potential for parts damage during assembly and part mating
- ability to do in-process testing
- occurrence of the need for reworking
- time of assembly
- assembly skill level
- unit cost of assembly

Mating condition

- Part coordinates MCS (modeling coord.)
- Base part: Datum
- Global CS
- Local CS
- Explicit position and direction vs mating conditions
- 4 x 4 homogeneous transformation matrix

Mating feature

Types: against, fits, contact, coplanar

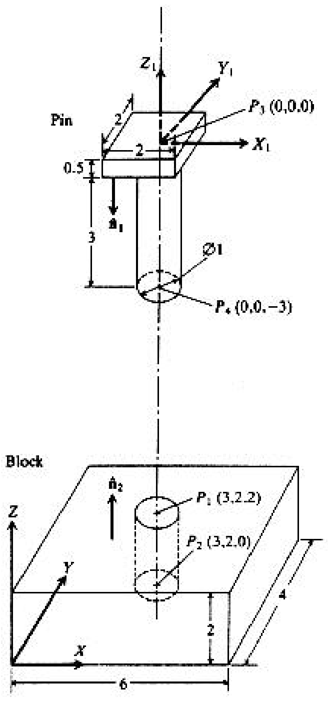
fits: center lines are concentric

- Mating condition = mating type + two faces
- Normal vector + one point on the face
- against: two normal vectors are in against directions
- fits: between two cylinders: center lines are concentric
- Against and fits allows rotation and translation between parts

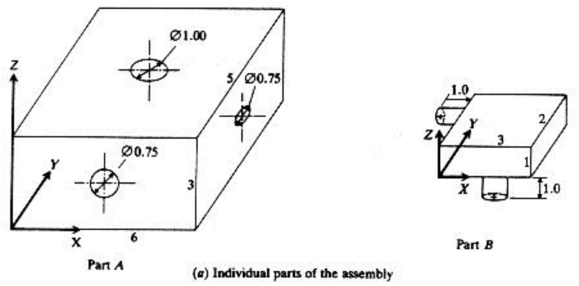
Interference fit

- Fits is clearance fit
- tight fits is interference fit
- Coplanar: two normal vectors are parallel
- 'Coplanar' complements 'against'

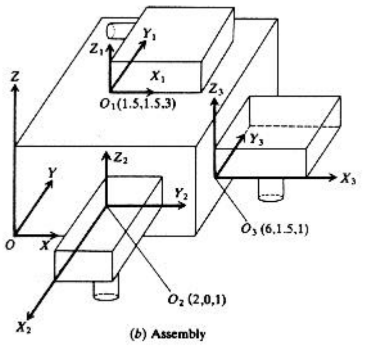
Example Pin and block



Assembly from instances

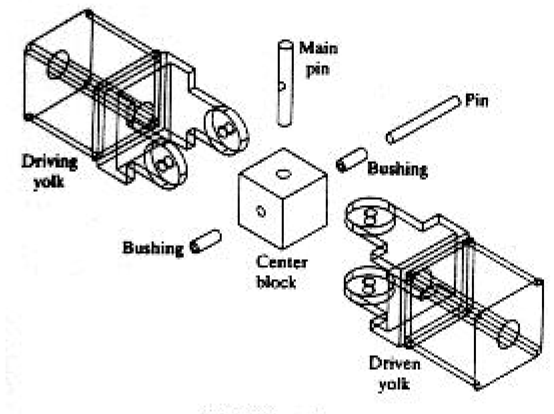


(a) Individual parts of the assembly

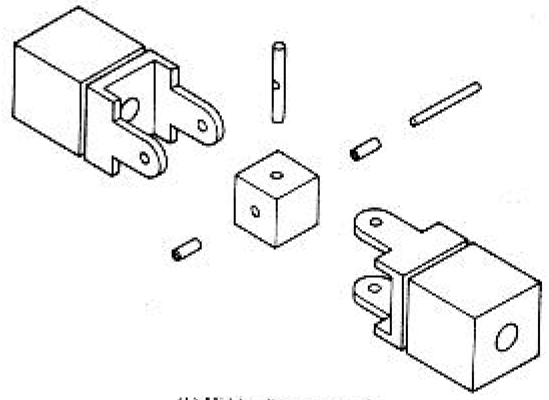


(b) Assembly

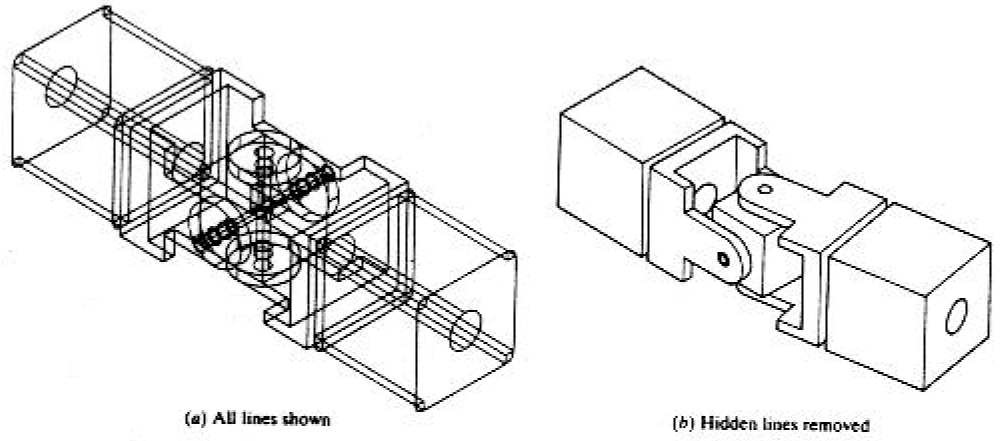
Exploded view of universal joint



(a) All lines shown


(b) Hidden lines removed

Assembly view of universal joint

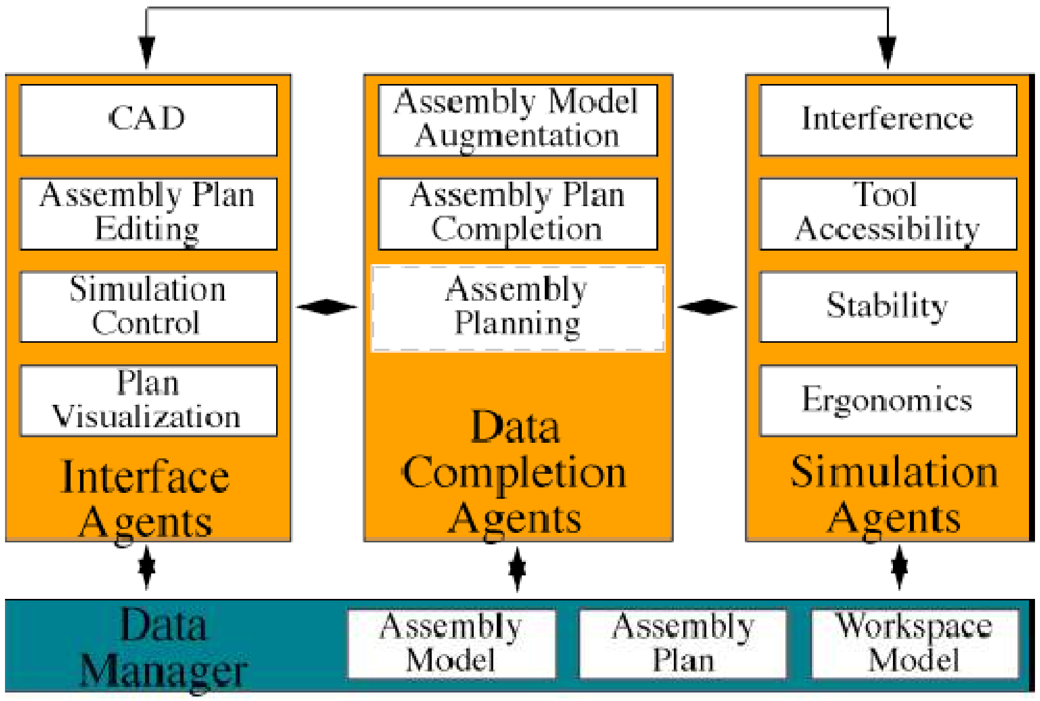


4.3. LAYOUT OF INTELLIGENT ASSEMBLY MODELING AND SIMULATION

The goal of IAMS is to avoid this expensive and time-consuming process by facilitating semblability checking in a virtual, simulated environment.

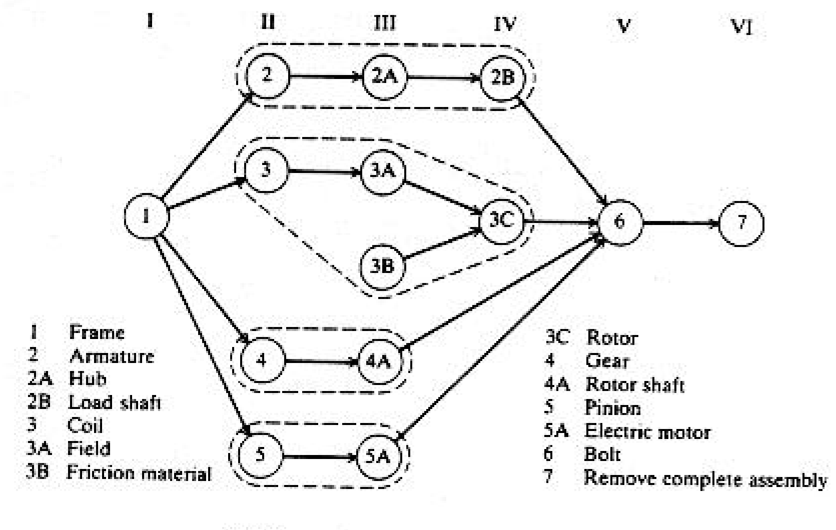
In addition to part-part interference checking, the IAMS tool will check for tool accessibility, stability, and ergonomics.

Intelligent Assembly Modeling and Simulation

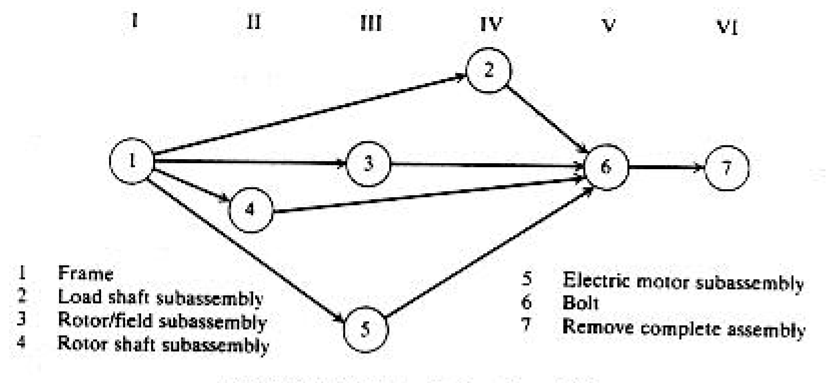


4.4. PRECEDENCE DIAGRAM

- Designed to show all the possible assembly sequences of a product.
- Each individual assembly operation is assigned a number.
- Diagram is usually organized into columns



(*a*) Diagram based on clutch individual parts



I
II
III
IV
V
VI
1
2
3
4
5
6
7
1 Frame
2 Load shaft subassembly
3 Rotor/field subassembly
4 Rotor shaft subassembly
5 Electric motor subassembly
6 Bolt
7 Remove complete assembly

(*b*) Diagram based on clutch subassemblies

4.5. PRODUCTION DRAWING LIMITS, FITS AND TOLERANCE

Limit system

There are three terms used in the limit system:

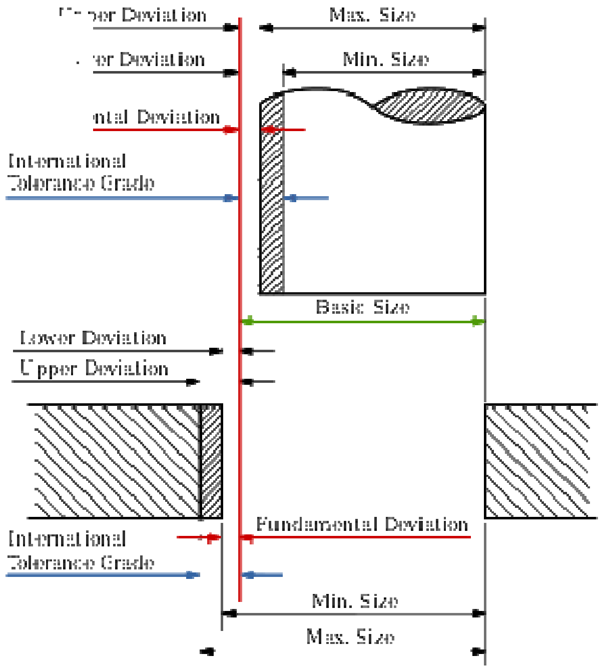
1. **Tolerance**: Deviation from a basic value is defined as Tolerance. It can be obtained by taking the difference between the maximum and minimum permissible limits.
2. **Limits**: Two extreme permissible sizes between which the actual size is contained are defined as limits.
3. **Deviation**: The algebraic difference between a size and its corresponding basic size. There are two types of deviations: 1) Upper deviation 2) Lower deviation

The fundamental deviation is either the upper or lower deviation, depending on which is closer to the basic size.

Tolerances

Due to human errors, machine settings, etc., it is nearly impossible to manufacture an absolute dimension as specified by the designer. Deviation in dimensions from the basic value always arises. This deviation of dimensions from the basic value is known as Tolerance.

The figure shows mechanical tolerances which occur during operations.



Fits

The relation between two mating parts is called fit. Depending upon the actual limits of the hole or shaft sizes, fits may be classified as clearance fit, transition fit and interference fit.

Clearance fit

Clearance fit is defined as a clearance between mating parts. In clearance fit, there is always a positive clearance between the hole and shaft.

Transition fit

Transition fit may result in either an interference or clearance, depending upon th
the tolerance of individual parts.

Interference fit

Interference fit is obtained if the difference between the hole
negative before assembly. Interference fit generally ranges from minin
interference. The two extreme cases of interference are as follows:

Minimum interference

The magnitude of the difference (negative) between the maximum size of the hole and the minimum size of the shaft in an interference fit before assembly.

Maximum interference

The magnitude of the difference between the minimum size of the hole and the maximum size of the shaft in an interference or a transition fit before assembly.

Hole Basis and shaft basis system:

In identifying limit dimensions for the three classes of fit, two systems are in use:

1. *Hole basis system*: The size of the shaft is obtained by subtracting the allowance from the basic size of the hole. Tolerances are then applied to each part separately. In this system, the lower deviation of the hole is zero. The letter symbol indication for this is 'H'.
2. *Shaft basis system*: The upper deviation of the shaft is zero, and the size of the hole is obtained by adding the allowance to the basic size of the shaft. The letter symbol indication is 'h'.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIT 5 - CAD STANDARDS - SMR1402

UNIT - V

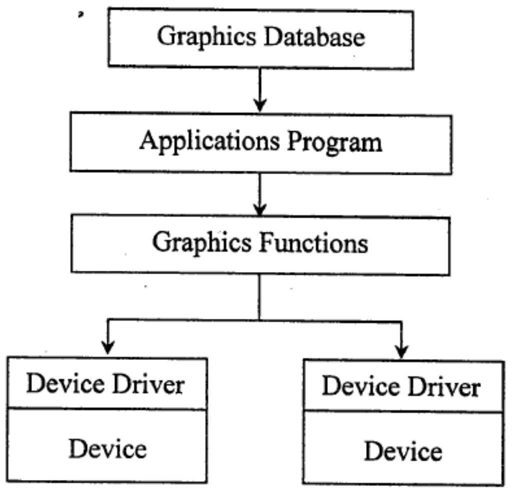
CAD STANDARDS

5.1. PRE-REQUISITE DISCUSSION

With the proliferation of computers and software in the market, it became necessary to standardize certain elements at each stage, so that investment made by companies in certain hardware or software was not totally lost and could be used without much modification on the newer and different systems. Standardization in engineering hardware is well known. Further, it is possible to obtain hardware and software from a number of vendors and then be integrated into a single system. This means that there should be compatibility between various software elements as also between the hardware and software. This is achieved by maintaining proper interface standards at various levels. Following are some of them.

- **GKS** (Graphical Kernel Systems)
- **PHIGS** (Programmer's Hierarchical Interface for Graphics)
- **CORE** (ACM-SIGGRAPH)
- **GKS-3D**
- **IGES** (Initial Graphics Exchange Specification)
- **DXF** (Drawing Exchange Format)
- **STEP** (Standard for the Exchange of Product Model Data)
- **DMIS** (Dimensional Measurement Interface Specification)
- **VDI** (Virtual Device Interface)
- **VDM** (Virtual Device Metafile)
- **GKSM** (GKS Metafile)
- **NAPLPS** North American Presentation Level Protocol Syntax)

Schematically, the operation of these standards with application programs is depicted in Figure. Details of some of these standards are discussed in the following sections.



Various Standards in Graphics Programming

5.2. GRAPHICAL KERNEL SYSTEM (GKS)

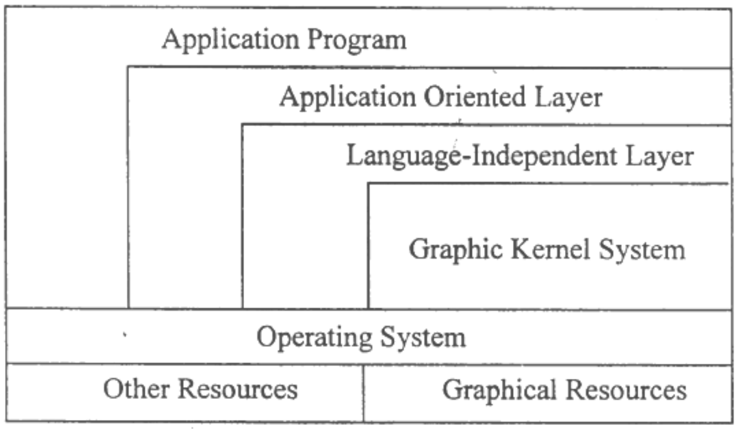
People with experience in graphical programming for various hardware systems know how difficult it is if a program developed for a particular system, were to run on a different system. The program would have to be completely recoded. If a comparison is made of a number of graphical programs, one may find that a substantial portion of it is similar. As a result, the same code is rewritten a number of times by different people. Some of the tasks such as drawing a line from one location to another , drawing a circle, setting a point, drawing a marker, form parts of such common features. Therefore, it is desirable to have programs interchangeable with a number of systems and also to make programmers learn the system once and then repeatedly use it on different systems. With this in mind, a number of attempts have been made in the past to provide a set of useful procedures (routines) for graphical manipulation, but with specific objective or hardware in mind. Examples are:

- **Tektronics Plot 10**
- **GINO-F from CAD Centre, Cambridge**

However, the real effort as regards standardisation came from ACM-SIGGRAPH in the form of CORE standard in 1977. In 1979, DIN released the version of GKS. Taking all the existing graphic packages, ISO has standardised the GKS as a 2D standard in 1982. The main objectives those were put forward for GKS are the following.

1. To provide the complete range of graphical facilities in 2D, including the interactive capabilities.
2. To control all types of graphic devices such as plotters and display devices in a consistent manner.
3. To be small enough for a variety of programs.

The major contribution of GKS for the graphics programming is in terms of the layer model, as shown in figure. An environment for user to work is termed as Work Station in GKS. This could be VDU, plotter or printer. For a programmer, all workstations are identical. The characteristics of these workstations are built into GKS. It is also possible to work simultaneously on more than one workstation.



Layer Model of Graphics Kernel System

The coordinate frames available to the users of the following three types.

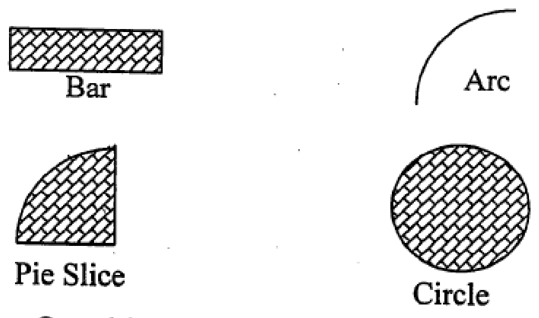
(i) World coordinates (WC) which are the user-oriented drawing coordinates.

(ii) Normalized device coordinates (NDC) which is a uniform system for all work stations.

(iii) Device coordinates (DC) are the actual coordinate system for the particular work station.

Input methods into GKS environment are organised in the following way:

LOCATOR	means of entering the location in world coordinates.
VALUATOR	real value in terms of distances.
CHOICE	integer options such as 0, 1, 2, 3, etc.
PICK	to select an object or segment in a drawing already created.
STRING	character values.
STROKE	to provide continuously the location values in world coordinates.



Graphics Primitives in IBM GKS

For drawing lines, the concept of PEN is used. PEN has the attributes of colour, thickness and line type. Lines can be drawn with any PEN that can defined. The basic graphic primitives that were made available are.

- **POLYLINE** for lines after specifying in the line type, line width and line colour.
- **POLYMARKER** for specific marker types after specifying the type, size and colour.
- **GENERALISED DRAWING PRIMITIVES (GDP)** for specific graphic primitives such as arc, circle, ellipse, spline, etc.
- **TEXT** after specifying font type, precision, colour, height of the box, expansion factor , spacing up vector and the path (left, right, up or down).
- **FILL AREA** for hatching and filling of areas.

In essence, the GKS is essentially a set of procedures that can be called by user programs for carrying out certain generalised functions. In the interest of interchangeability, ISO has identified certain calling conventions for all these functions in various languages in order to take care of the variability of the programming languages.

GKS is defined in terms of a number of levels describing the level of support in terms of facilities. The highest level is 2c, though level 2b is the most commonly available facility with marginal difference in terms of the length of input queue (5 in case of 2c and 0 in case of 2b). A number of implementations are available for GKS on all types of computer starting from the micros to the main frame computers.

DATA EXCHANGE STANDARDS

5.3. IGES STANDARD

However, the IGES is the most comprehensive standard and ***is designed to transmit the entire product definition including that of manufacturing and any other associated information.*** A brief description of the IGES version 3.0 is given below highlighting the philosophy of the conversion methodology.

In IGES the records are present with 80 column fields, with columns 1 to 72 providing the data and columns 73 to 80 providing a sequence number for the record with identification as to the location of the sub-section. This sequence number is utilized as a pointer for the data. The IGES file consists of the following 6 sub-sections.

(a) Flag Section
This is optional and is used to indicate the form in which the data is specified. Originally, the initial versions contained the data in ASCII format with a very detailed structure. This has been criticized by a number of people in view of the very large file sizes. From version 3.0 onwards, the format has been standardized in the following three modes.

- ASCII mode — default option
- Binary form
- Compressed ASCII form

The other two options provided will help in reducing the bulk of the drawing exchange file size. The sequence number has a starting character signifying the sub-section. They are:

S for Start section	G for Global section
D for Directory entry section	P for Parameter entry section
T for terminate section	

(b) Start Section
This section contains a man-readable prologue to the file. The information contained in this section is essentially for the person who would be post processing this for any other application. Any number of lines can be contained in this section. A sample listing an IGES file for the drawing shown in Figure.

(c) Global Section
This contains information about details of the product, the person originating the product, name of the company originating it, date, the details of the system which generated it, drafting standard used and some information required for its post processing on the host computer.

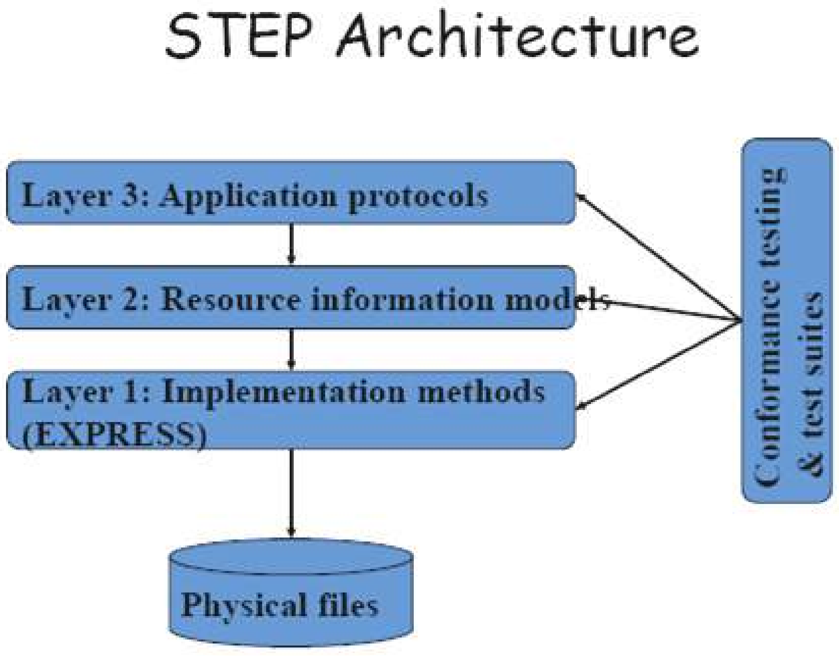
(d) Directory Entry Section
For each entity present in the drawing is fixed in size and contains 20 fields of 8 characters each. The purpose of this section is to provide an index for the file and to contain attribute information. Some of the attribute information such as colour, line type, transformation matrix, etc. may be present directly or through a pointer (to a record in the same file) where the necessary information is stored. It also contains the pointer to the parameter data section entry which actually contains the requisite parameter data.

(e) Parameter Data Section
This contains the data associated with the entities. A free format is allowed for maximum convenience. It may contain any number of records. The total number of entities that are present in IGES version 5.1.

(f) Terminate Section
This contains the sub-totals of the records present in each of the earlier sections. This would always contain a single record.

5.4. STEP (Standard for the Exchange of Product model Data)

- Standard for Exchange of Product Model Data
- Uses a formal model for data exchange
- Information is modeled using the EXPRESS language
- EXPRESS has elements of Pascal, C, and other languages
- It contains constructs for defining data types and structures, but not for processing data
- EXPRESS describes geometry and other information in a standard, unambiguous way

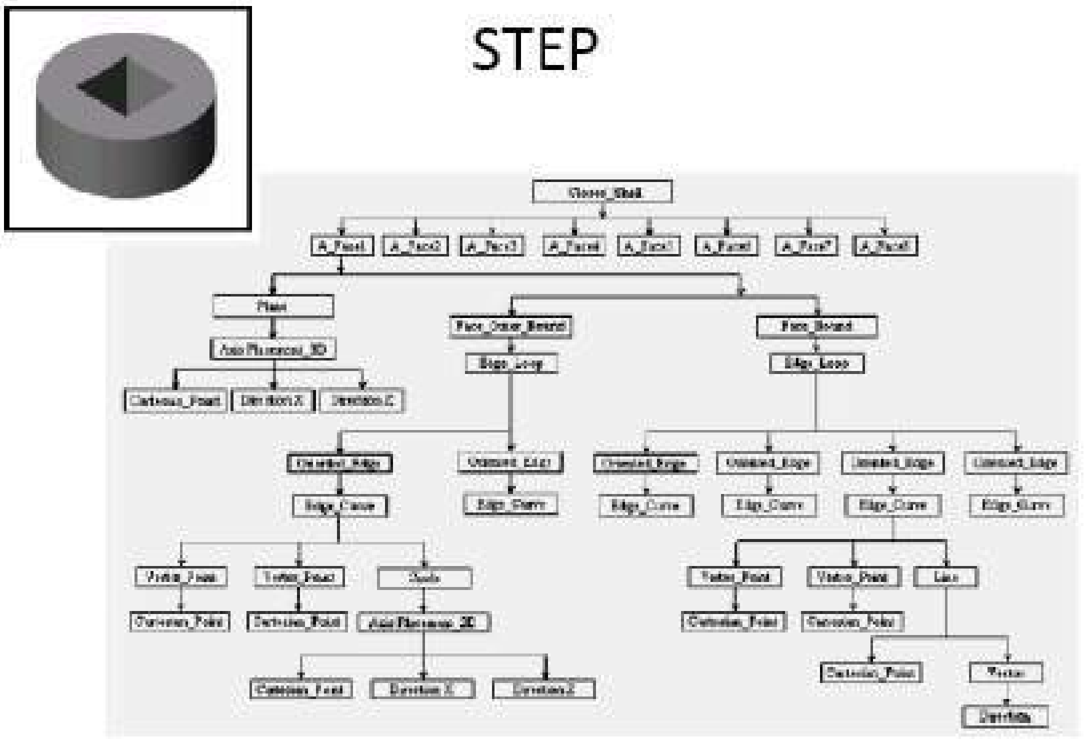

STEP Architecture
Layer 3: Application protocols
Layer 2: Resource information models
Layer 1: Implementation methods (EXPRESS)
Physical files
Conformance testing & test suites

Classes of STEP Parts

•Introductory
•Description methods
•Implementation methods
•Conformance testing methodology and framework
•Integrated resources
•Application protocols
•Abstract test suites
•Application interpreted constructs

Status of STEP

•STEP has been under development for many years, and will continue for many more
•Over a dozen STEP parts have been approved as international standards
•Many others are under development



5.5. CONTINUOUS ACQUISITION AND LIFE-CYCLE SUPPORT (CALS)

•Developed by US Department of Defense
•Prescribes formats for storage and exchange of technical data
•Technical publications an important focus

Important CALS Standards

• Standard Generalized Markup Language (SGML)
-developed in 1960s IBM

i. document description language
ii. separates content from structure (formatting)
iii. uses "tags" to define headings, sections, chapters, etc.
iv. HTML is based on SGML

• Computer Graphics Metafile (CGM)

i. Developed in 1986
ii. vector file format for illustrations and drawings
iii. All graphical elements can be specified in a textual source file that can be compiled into a binary file or one of two text representations

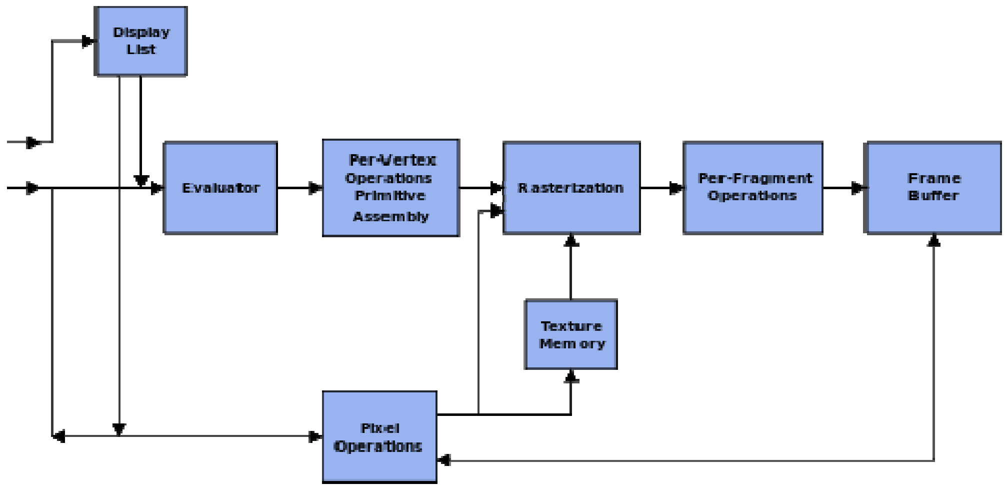
5.6.OpenGL (Open Graphics Library)

OpenGL is a cross-language, multi-platform application programming interface (API) forrendering 2D and 3D vector graphics. The API is typically used to interact with a processing unit (GPU), to achieve hardware-accelerated rendering.

The OpenGL specification describes an abstract API for drawing 2D Although it is possible for the API to be implemented entirely in software, i implemented mostly or entirely in hardware.

The API is defined as a number of functions which may be called by t alongside a number of named integer constants (for example, the constant GI which corresponds to the decimal number 3553). Although the functio superficially similar to those of the C programming language, they are lang As such, OpenGL has many language bindings, some of the most the JavaScriptbinding WebGL (API, based on OpenGL ES 2.0, for 3D rend aweb browser); the C bindings WGL, GLX and CGL; the C binding prov the Java and C bindings provided by Android.

In addition to being language-independent, OpenGL is also platform specification says nothing on the subject of obtaining, and managing, an leaving this as a detail of the underlying windowing system. For the same r purely concerned with rendering, providing no APIs related to input, audio, or



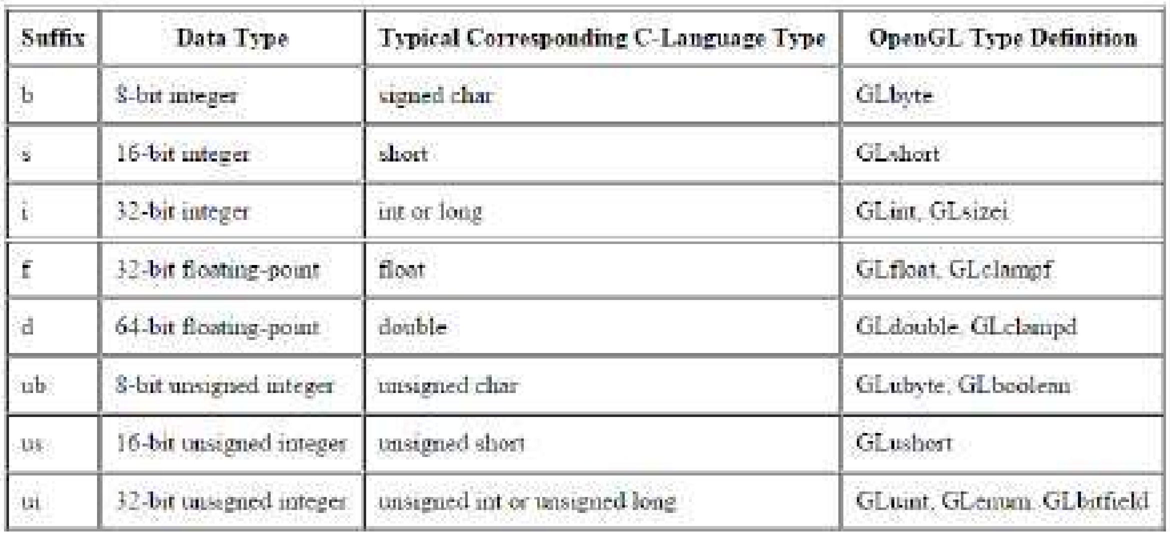
OpenGL Command Syntax

As you might have observed from the simple program in the previous section, OpenGL commands use the prefix **gl** and initial capital letters for each word making up the command name (recall **glClearColor()**, for example). Similarly, OpenGL defined constants begin with GL_, use all capital letters, and use underscores to separate words (like GL_COLOR_BUFFER_BIT).

You might also have noticed some seemingly extraneous letters appended to some command names (for example, the **3f** in **glColor3f() and glVertex3f()**). It's true that the **Color** part of the command name **glColor3f()** is enough to define the command as one that sets the current color. However, more than one such command has been defined so that you can use different types of arguments. In particular, the **3** part of the suffix indicates that three arguments are given; another version of the **Color** command takes four arguments. The **f** part of the suffix indicates that the arguments are floating-point numbers. Having different formats allows OpenGL to accept the user's data in his or her own data format.

Some OpenGL commands accept as many as 8 different data types for their arguments. The letters used as suffixes to specify these data types for ISO C implementations of OpenGL are shown in Table 1-1, along with the corresponding OpenGL type definitions. The particular implementation of OpenGL that you're using might not follow this scheme exactly; an implementation in C++ or Ada, for example, wouldn't need to.

Table: Command Suffixes and Argument Data Types



Suffix	Data Type	Typical Corresponding C-Language Type	OpenGL Type Definition
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	int or long	GLint, GLsizei
f	32-bit floating-point	float	GLfloat, GLclampf
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned int or unsigned long	GLuint, GLenum, GLbitfield

OpenGL-Related Libraries

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to allow you to simplify your programming tasks, including the following:

- The OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. This library is provided as part of every OpenGL implementation. Portions of the GLU are described in the *OpenGL Reference Manual*. The more useful GLU routines are described in this guide, where they're relevant to the topic being discussed, such as in all of Chapter 11 and in the section "The GLU NURBS Interface". GLU routines use the prefix **glu**.
- For every window system, there is a library that extends the functionality of that window system to support OpenGL rendering. For machines that use the X Window System, the OpenGL Extension to the X Window System (GLX) is provided as an adjunct to OpenGL. GLX routines use the prefix **glX**. For Microsoft Windows, the WGL routines provide the Windows to OpenGL interface. All WGL routines use the prefix **wgl**. For IBM OS/2, the PGL is the Presentation Manager to OpenGL interface, and its routines use the prefix **pgl**.

- The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit, written by Mark Kilgard, to hide the complexities of differing window system APIs. GLUT is the subject of the next section, and it's described in more detail in Mark Kilgard's book *OpenGL Programming for the X Window System* (ISBN 0-201-48359-9). GLUT routines use the prefix **glut.** "How to Obtain the Sample Code" **in the Preface** describes how to obtain the source code for GLUT, using ftp.
- Open Inventor is an object-oriented toolkit based on OpenGL which provides objects and methods for creating interactive three-dimensional graphics applications. Open Inventor, which is written in C++, provides prebuilt objects and a built-in event model for user interaction, high-level application components for creating and editing three-dimensional scenes, and the ability to print objects and exchange data in other graphics formats. Open Inventor is separate from OpenGL.