MOHAMED SATHAK A.J. COLLEGE OF ENGINEERING
(Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai)

ISO 9001:2015 Certified by IRQS
NABCB QM 006

Department of Mechanical Engineering

Lecture Notes

ME8691 - COMPUTER AIDED DESIGN & MANUFACTURING

UNIT I FUNDAMENTALS OF COMPUTER GRAPHICS

Product cycle- Design process- sequential and concurrent engineering- Computer aided design – CAD system architecture- Computer graphics – co-ordinate systems- 2D and 3D transformations- homogeneous coordinates - Line drawing -Clipping- viewing transformation

1.1. Introduction of CAD

In the mid of 1970s, as computer aided design starts to offer more potential than just a skill to replicate manual drafting with electronic drafting, the cost gain for companies to switch to CAD became obvious. The benefit of CAD methods over manual drafting are the capabilities one often takes for established from computer systems; automated creation of Bill of Material, interference checking, auto layout in integrated circuits.

1.2. Product cycle

Product cycle integrate processes, people, data, and business and gives a product information for industries and their extended activity. Product cycle is the process of managing the entire lifecycle of a product from starting, through design and manufacture, to repair and removal of manufactured products.

Product cycle methods assist association in managing with the rising difficulty and engineering challenges of developing new products for the worldwide competitive markets.

Product lifecycle management (PLM) can be part of one of the following four fundamentals of a manufacturing information technology structure.

(i) Customer Relationship Management (CRM)

(ii) Supply Chain Management (SCM)

(iii) Enterprise resource planning (ERP)

(iv) Product Planning and Development (PPD).

The core of PLM is in the formation and management of all product information and the technology used to access this data and knowledge. PLM as a discipline appeared from tools such as CAD, CAM and PDM, but can be viewed as the combination of these tools with processes, methods and people through all stages of a product's life cycle. PLM is not just about software technology but is also a business approach.

1.2.1. Product Cycle Model

There are several Product cycle models in industry to be considered, one of the possible product cycle is given below (Fig.1.1.):

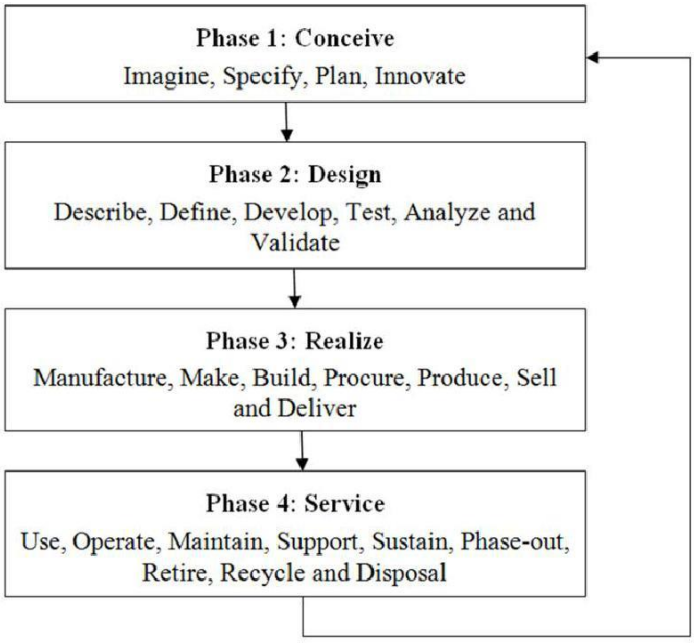


Fig.1.1. Product Cycle Model

Step 1: Conceive

Imagine, Specify, Plan, Innovate

The first step is the definition of the product requirements based on company, market and customer. From this requirement, the product's technical data can be defined. In parallel, the early concept design work is performed defining the product with its main functional features. Various media are utilized for these processes, from paper and pencil to clay mock-up to 3D Computer Aided Industrial Design.

Step 2: Design

Describe, Define, Develop, Test, Analyze and Validate

This is where the completed design and development of the product begins, succeeding to prototype testing, through pilot release to final product. It can also involve redesign and ramp for

improvement to existing products as well as planned obsolescence. The main tool used for design and development is CAD. This can be simple 2D drawing / drafting or 3D parametric feature based solid/surface modeling.

This step covers many engineering disciplines including: electronic, electrical, mechanical, and civil. Besides the actual making of geometry there is the analysis of the components and assemblies.

Optimization, Validation and Simulation activities are carried out using Computer Aided Engineering (CAE) software. These are used to perform various tasks such as: Computational Fluid Dynamics (CFD); Finite Element Analysis (FEA); and Mechanical Event Simulation (MES). Computer Aided Quality (CAQ) is used for activities such as Dimensional tolerance analysis. One more task carried out at this step is the sourcing of bought out components with the aid of procurement process.

Step 3: Realize

Manufacture, Make, Build, Procure, Produce, Sell and Deliver

Once the design of the components is complete the method of manufacturing is finalized. This includes CAD operations such as generation of CNC Machining instructions for the product's component as well as tools to manufacture those components, using integrated Computer Aided Manufacturing (CAM) software.

It includes Production Planning tools for carrying out plant and factory layout and production simulation. Once details components are manufactured their geometrical form and dimensions can be verified against the original data with the use of Computer Aided Inspection Equipment (CAIE). Parallel to the engineering tasks, sales and marketing work take place. This could consist of transferring engineering data to a web based sales configuration.

Step 4: Service

Use, Operate, Maintain, Support, Sustain, Phase-out, Retire, Recycle and Disposal

The final step of the lifecycle includes managing of information related to service for repair and maintenance, as well as recycling and waste management information. This involves using tools like Maintenance, Repair and Operations Management software.

1.3. Design Process

The design process includes series of steps that engineers apply in making functional products and processes. The parts of the process often need to be repeated many times before production of a product can start. The parts that get iterated and the number of such design cycles in any given project can be highly changeable.

One method of the engineering design process focuses on the following common aspects:

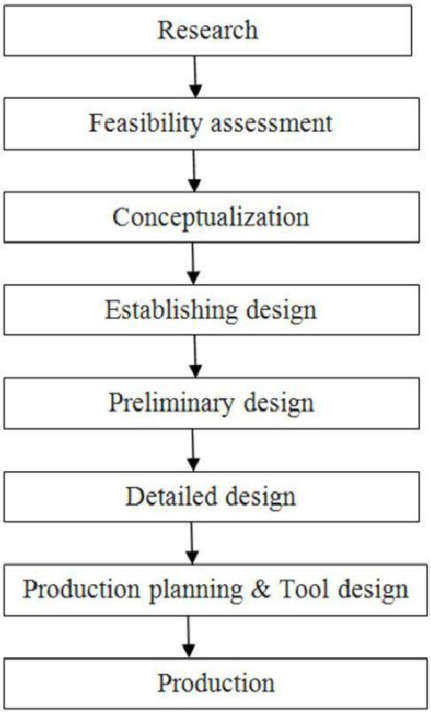


Fig.1.2. Design Process

1. Research

A considerable amount of time is used on research, or finding information. Consideration should be given to the available applicable literature, issues and successes linked with avaialbe solutions, and need of marketplaces.

The basis of information should be significant, including existing results. Reverse engineering can be a successful technique if other solutions are available in the market. Added sources of information include the trade journals, available government documents, local libraries, vendor catalogs and personal organizations.

2. Feasibility assessment

The feasibility study is an analysis and assessment of the possible of a proposed design which is based on detail investigation and research to maintain the process of decision creation. The feasibility assessment helps to focus the scope of the project to spot the best situation. The purpose of a feasibility assessment is to verify whether the project can continue into the design phase.

3. Conceptualization

A Concept Study is the stage of project planning that includes developing ideas and taking into account the all features of executing those ideas. This stage of a project is done to reduce the likelihood of assess risks, error and evaluate the potential success of the planned project.

4. Establishing the design requirements

Establishing design requirements is one of the most essential elements in the design practice, and this task is usually performed at the same time as the feasibility analysis. The design requirements control the design of the project all over the engineering design process. A few design requirements comprise maintainability, hardware and software parameters, availability, and testability.

5. Preliminary design

The preliminary design fills the gap between the design concept and the detailed design phase. During this task, the system configuration is defined, and schematics, diagrams, and layouts of the project will offer early project configuration. In detailed design and optimization, the parameters of the part being produced will change, but the preliminary design focuses on creating the common framework to construct the project.

6. Detailed design

The next phase of preliminary design is the Detailed Design which may includes of procurement also. This phase builds on the already developed preliminary design, aiming to further develop each phase of the project by total description through drawings, modeling as well as specifications.

The advancement CAD programs have made the detailed design phase more competent. This is because a CAD program can offer optimization, where it can shrink volume without compromising the part's quality. It can also calculate displacement and stress using the FEM to find stresses throughout the part. It is the responsibility of designer to find whether these stresses and displacements are acceptable, so the part is safe.

7. Production planning and tool design

The production planning and tool design is more than planning how to mass-produce the project and which tools should be used in the manufacturing of the component. Tasks to complete in this stage include material selection, identification of the production processes, finalization of the sequence of

operations, and selection of jigs, fixtures, and tooling. This stage also includes testing a working prototype to confirm the created part meets qualification standards.

With the finishing of qualification testing and prototype testing, the design process is completed.

1.4. Sequential and Concurrent Engineering

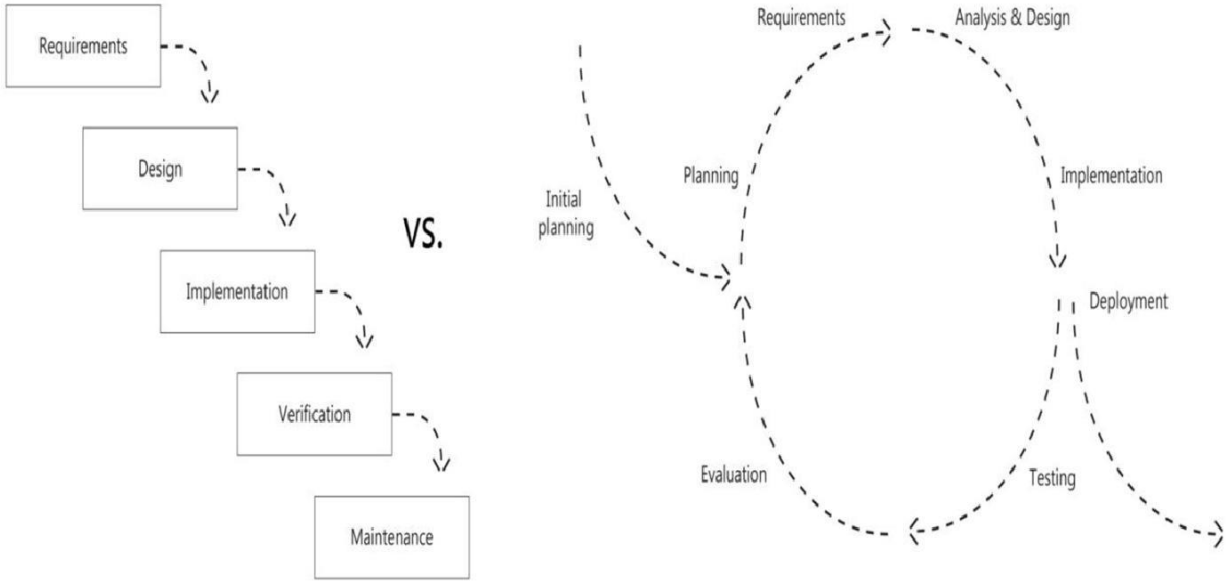


Fig. 1.3. Sequential Vs Concurrent Engineering

Table 1.1. Sequential Vs Concurrent Engineering

Sequential Engineering	Concurrent Engineering
Sequential engineering is the term used to explain the method of production in a linear system. The various steps are done one after another, with all attention and resources focused on that single task.	In concurrent engineering, various tasks are handled at the same time, and not essentially in the standard order. This means that info found out later in the course can be added to earlier parts, improving them, and also saving time.

Sequential engineering is a system by which a group within an organization works sequentially to create new products and services.	Concurrent engineering is a method by which several groups within an organization work simultaneously to create new products and services.
The sequential engineering is a linear product design process during which all stages of manufacturing operate in serial.	The concurrent engineering is a non-linear product design process during which all stages of manufacturing operate at the same time.
Both process and product design run in serial and take place in the different time.	Both product and process design run in parallel and take place in the same time.
Process and Product are not matched to attain optimal matching.	Process and Product are coordinated to attain optimal matching of requirements for effective quality and delivery.
Decision making done by only group of experts.	Decision making involves full team involvement.

1.5. Computer Aided Design

CAD is the intersection of Computer Graphics, Geometric modeling and Design tools (Fig.1.4.). The concepts of computer graphics and geometric modeling and must be used innovatively to serve the design process.

CAD is the function of computer systems to support in the creation, modification, analysis, or optimization of a design. CAD software is used to raise the productivity of the designer, progress the

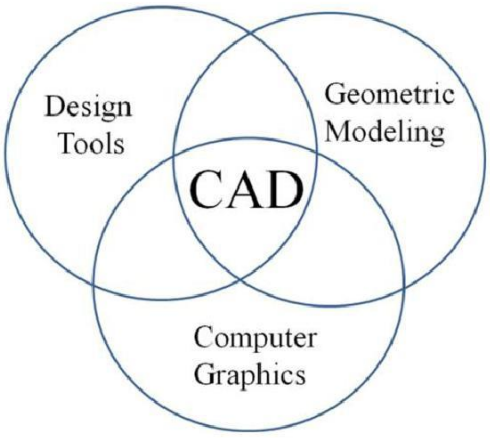


Fig. 1.4. CAD

CAD software for design uses either vector-based graphics to explain the objects of traditional drafting, or may also develop raster graphics showing the overall look of designed objects. During the manual drafting of engineering drawings, the output of CAD must convey information, like dimensions, materials, processes, and tolerances.

CAD is a significant industrial art used in many purposes, including industrial and architectural design, shipbuilding, automotive, and aerospace industries, and many more. CAD is also extensively used to create computer animation for special effects in movies, and technical manuals, frequently called as Digital Content Creation.

CAD software packages provide the designer with a multi window environment with animation which is regularly used in Digital Content Creation. The animations using wire frame modeling helps the designer to see into the interior of object and to observe the behaviors of the inner components of the assembly during the motion.

1.5.1. CAD Technology

Initially software for CAD systems was developed with computer languages such as FORTRAN but with the development of object-oriented programming methods this has completely changed. Classic modern parametric attribute based modeler and freeform surface systems are developing around a number of key ‘C’ modules.

A CAD system can be seen as develop from the interaction of a Graphical User Interface (GUI) with NURBS geometry and Boundary representation data through a kernel for geometric modeling. A geometry constraint engine may also be employed to organize the associative relationships between components in an assembly.

Unexpected facilities of these relationships have led to a new form of prototyping called digital prototyping. In difference to physical prototypes, which involve manufacturing time in the design. CAD models can be created by a computer after the physical prototype has been scanned using an CT scanning device. Based on the nature of the business, digital or physical prototypes can be primarily selected according to specific requirements.

Currently, no special hardware is required for CAD software. However, some special CAD systems can do graphically and computationally intensive tasks, so a higher end graphics card, high speed CPUs may be suggested. CAD systems exist for all the major platforms and some packages even perform multiple platforms.

The human-machine interface is generally through a mouse but can also be using a digitizing graphics tablet. Handling of the view of the part on the screen is also sometimes done with the help of a Space mouse or Space Ball. Special CAD systems also support stereoscopic glasses for viewing the 3D objects.

1.5.2. CAD Tools

The CAD tools are mainly using for graphics applications and modeling. Aids such a color, grids, geometric modifiers and group facilitate structural geometric models. Visualization is achieved through shaded components and animation which focus design conceptualization, communication and interference detection. FEM packages provide optimization in shape and structure. Adding tolerances, tolerance analysis and investigating the effect of manufacturing on the design can perform by utilizing CAD tools (Table 1.2).

Table 1.2. CAD Tools Vs Design Process

CAD Tools	Design Process
Geometric modeling, Graphics aids, visualization and manipulation	Conceptualization

Geometric modeling, Graphics aids, visualization and manipulation, animation, assemblies	Modeling and Simulation
Analysis packages, customized programs	Design Analysis
Structural optimization	Design Optimization
Dimensioning, tolerance, bill of materials	Design evaluation
Drafting and detailing, Shaded images	Communication and Documentation

1.5.3. Uses of CAD

CAD is one of the tools used by designers and engineers and is used in different ways depending on the profession of the customer and the type of software.

CAD is one of the Digital Product Development activities within the Product Lifecycle Management practices with other tools, which are either integrated modules or individual, such as:

- Computer Aided engineering (CAE) and Finite Element Analysis (FEA)
- Computer Aided Manufacturing (CAM)
- Realistic Rendering and Simulation.
- Product Data Management (PDM).

CAD is also used for the development of photo simulations that are frequently necessary in the preparation of Environmental Impact Reports, in which proposed CAD buildings are superimposed into photographs of existing situation to represent what that conditions will be like, where the proposed services are allowed to be built.

Parameters and constraints can be used to get the size, shape, and other properties of the modeling elements. The features of the CAD system can be used for the several tools for measurement such as yield strength, tensile strength and electrical or electro-magnetic properties.

1.6. CAD System Architecture

Computer architecture is a pattern describing how a group of software and hardware technology standards relate to form a computer system. In general, computer architecture refers to how a computer is designed and what technologies it is compatible with. Computer architecture is likened to the art of shaping the needs of the technology, and developing a logical design and standards based on needs.

In CAD, Computer architecture is a set of disciplines that explains the functionality, the organization and the introduction of computer systems; that is, it describes the capabilities of a computer and its programming method in a summary way, and how the internal organization of the system is

designed and executed to meet the specified facilities. Computer architecture engages different aspects, including instruction set architecture design, logic design, and implementation. The implementation includes Integrated Circuit Design, Power, and Cooling. Optimization of the design needs expertise with Compilers, Operating Systems and Packaging.

1. Instruction set architecture

An instruction set architecture is the interface between the software and hardware and also can be observed as the programmer's view of the machine. Computers do not understand high level languages, if any, language elements that translate directly into a machine's native op codes. A processor

UNIT II GEOMETRIC MODELING

Representation of curves- Hermite curve- Bezier curve- B-spline curves-rational curves- Techniques for surface modeling – surface patch- Coons and bicubic patches- Bezier and B-spline surfaces. Solid modeling techniques- CSG and B-rep

Geometric Modeling

2.1. Introduction

Geometric modeling is a part of computational geometry and applied mathematics that studies algorithms and techniques for the mathematical description of shapes.

The shapes defined in geometric modeling are generally 2D or 3D, even though several of its principles and tools can be used to sets of any finite dimension. Geometric modeling is created with computer based applications. 2D models are significant in computer technical drawing and typography. 3D models are fundamental to CAD and CAM and extensively used in many applied technical branches such as civil engineering and mechanical engineering and medical image processing.

Geometric models are commonly differentiated from object oriented models and procedural, which describe the shape perfectly by an opaque algorithm that creates its appearance. They are also compared with volumetric models and digital images which shows the shape as a subset of a regular partition of space; and with fractal models that provide an infinitely recursive description of the shape. Though, these differences are often fuzzy: for example, a image can be interpreted as a collection of colored squares; and geometric shape of circles are defined by implicit mathematical equations. Also, a fractal model gives a parametric model when its recursive description is truncated to a finite depth.

2.2. Representation of curves

A curve is an entity related to a line but which is not required to be straight. A curve is a topological space which is internally homeomorphism to a line; this shows that a curve is a set of points which close to each of its points looks like a line, up to a deformation.

A conic section is a curve created as the intersection of a cone with a plane. In analytic geometry, a conic may be described as a plane algebraic curve of degree two, and as a quadric of dimension two.

There are several of added geometric definitions possible. One of the most practical, in that it involves only the plane, is that a non circular conic has those points whose distances to various point, called a ‘focus’, and several line, called a ‘directrix’, are in a fixed ratio, called the ‘eccentricity’.

2.2.1.Conic Section

Conventionally, the three kinds of conic section are the hyperbola, the ellipse and the parabola. The circle is a unique case of the ellipse, and is of adequate interest in its own right that it is sometimes described the fourth kind of conic section. The method of a conic relates to its 'eccentricity', those with eccentricity less than one is ellipses, those with eccentricity equal to one is parabolas, and those with eccentricity greater than one is hyperbolas. In the focus, directrix describes a conic the circle is a limiting with eccentricity zero. In modern geometry some degenerate methods, such as the combination of two lines, are integrated as conics as well.

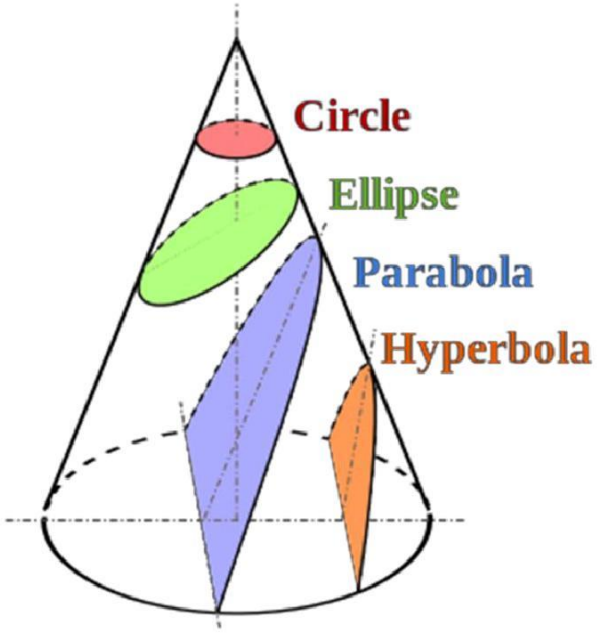


Fig.2.1. Conic sections

The three kinds of conic sections are the ellipse, parabola, and hyperbola. The circle can be taken as a fourth kind of ellipse. The circle and the ellipse occur when the intersection of plane and cone is a closed curve. The circle is generated when the cutting plane is parallel to the generating of the cone. If the cutting plane is parallel to accurately one generating line of the cone, then the conic is unbounded and is mentioned a parabola. In the other case, the figure is a hyperbola.

Different factors are connected with a conic section, as shown in the Table 2.1. For the ellipse, the table shows the case of 'a' > 'b', for which the major axis is horizontal; for the other case, interchange the symbols 'a' and 'b'. For the hyperbola the east-west opening case is specified. In all cases, 'a' and 'b' are positive.

Table 2.1. Conic Sections

conic section	equation	eccentricity (e)	linear eccentricity (c)	semi-latus rectum (ℓ)	focal parameter (p)
circle	$x^2 + y^2 = a^2$	0	0	a	∞
ellipse	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$	$\sqrt{1 - \frac{b^2}{a^2}}$	$\sqrt{a^2 - b^2}$	$\frac{b^2}{a}$	$\frac{b^2}{\sqrt{a^2 - b^2}}$
parabola	$y^2 = 4ax$	1	–	$2a$	$2a$
hyperbola	$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$	$\sqrt{1 + \frac{b^2}{a^2}}$	$\sqrt{a^2 + b^2}$	$\frac{b^2}{a}$	$\frac{b^2}{\sqrt{a^2 + b^2}}$

The non-circular conic sections are accurately those curves that, for a point 'F', a line 'L' not having 'F' and a number 'e' which is non-negative, are the locus of points whose distance to 'F' equals 'e' multiplies their distance to 'L'. 'F' is called the focus, 'L' the directrix, and 'e' the eccentricity.

i. Linear eccentricity (c) is the space between the center and the focus.

ii. Latus rectum (2l) is parallel to the directrix and passing via the focus.

iii. Semi-latus rectum (l) is half the latus rectum.

iv. Focal parameter (p) is the distance from the focus to the

directrix. The relationship for the above : p*e = l and a*e=c.

2.3. Hermite curve

A Hermite curve is a spline where every piece is a third degree polynomial defined in Hermite form: that is, by its values and initial derivatives at the end points of the equivalent domain interval. Cubic Hermite splines are normally used for interpolation of numeric values defined at certain dispute values x1,x2,x3, ….., xn, to achieve a smooth continuous function. The data should have the preferred function value and derivative at each X_k. The Hermite formula is used to every interval (X_k, X_{k+1}) individually. The resulting spline become continuous and will have first derivative.

Cubic polynomial splines are specially used in computer geometric modeling to attain curves that pass via defined points of the plane in 3D space. In these purposes, each coordinate of the plane is individually interpolated by a cubic spline function of a divided parameter't'.

Cubic splines can be completed to functions of different parameters, in several ways. Bicubic splines are frequently used to interpolate data on a common rectangular grid, such as pixel values in a digital picture. Bicubic surface patches, described by three bicubic splines, are an necessary tool in

computer graphics. Hermite curves are simple to calculate but also more powerful. They are used to well interpolate between key points.

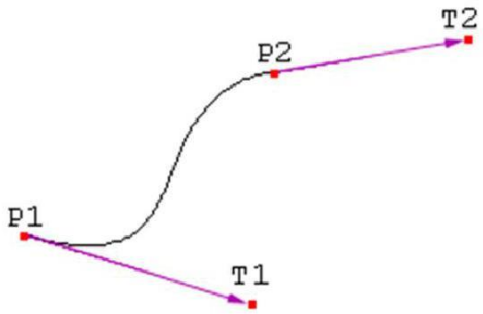


Fig.2.2. Hermite curve

The following vectors needs to compute a Hermite curve:

- P1: the start point of the Hermite curve
- T1: the tangent to the start point
- P2: the endpoint of the Hermite curve
- T2: the tangent to the endpoint

These four vectors are basically multiplied with four Hermite basis functions h1(s), h2(s), h3(s) and,h4(s) and added together.

$h1(s) = 2s^3 - 3s^2 + 1$
$h2(s) = -2s^3 + 3s^2$
$h3(s) = s^3 - 2s^2 + s$
$h4(s) = s^3 - s^2$

Figure 2.3 shows the functions of Hermite Curve of the 4 functions (from left to right: h1, h2, h3, h4).

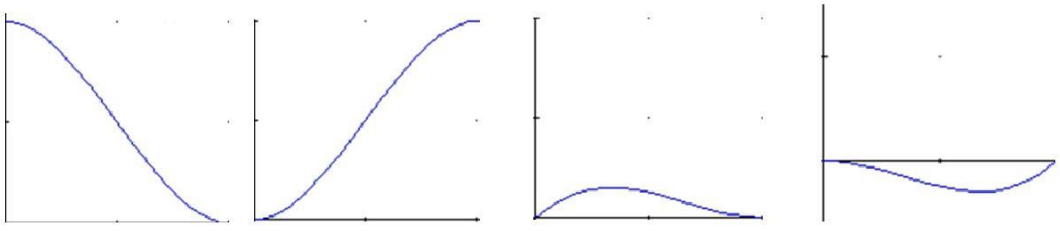


Fig.2.3. Functions of Hermite curve

A closer view at functions ‘h1’ and ‘h2’, the result shows that function ‘h1’ starts at one and goes slowly to zero and function ‘h2’ starts at zero and goes slowly to one.

At the moment, multiply the start point with function ‘h1’ and the endpoint with function ‘h2’. Let s varies from zero to one to interpolate between start and endpoint of Hermite Curve. Function ‘h3’ and function ‘h4’ are used to the tangents in the similar way. They make confident that the Hermite curve bends in the desired direction at the start and endpoint.

2.4. Bezier curve

Bezier curves are extensively applied in CAD to model smooth curves. As the curve is totally limited in the convex hull of its control points P0, P1,P2 & P3, the points can be graphically represented and applied to manipulate the curve logically. The control points P0 and P3 of the polygon lie on the curve (Fig.2.4.). The other two vertices described the order, derivatives and curve shape. The Bezier curve is commonly tangent to first and last vertices.

Cubic Bezier curves and Quadratic Bezier curves are very common. Higher degree Bezier curves are highly computational to evaluate. When more complex shapes are required, Bezier curves in low order are patched together to produce a composite Bezier curve. A composite Bezier curve is usually described to as a ‘path’ in vector graphics standards and programs. For smoothness assurance, the control point at which two curves meet should be on the line between the two control points on both sides.

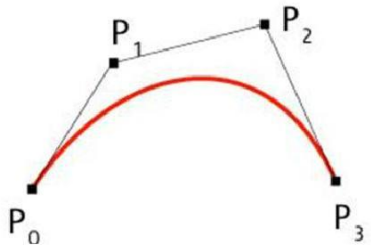


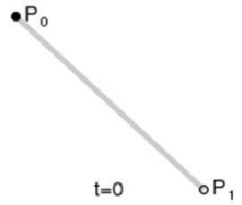
Fig.2.4. Bezier curve

A general adaptive method is recursive subdivision, in which a curve's control points are verified to view if the curve approximates a line segment to within a low tolerance. If not, the curve is further divided parametrically into two segments, $0 \leq t \leq 0.5$ and $0.5 \leq t \leq 1$, and the same process is used recursively to each half. There are future promote differencing techniques, but more care must be taken to analyze error transmission.

Analytical methods where a Bezier is intersected with every scan line engage finding roots of cubic polynomials and having with multiple roots, so they are not often applied in practice. A Bezier curve is described by a set of control points P_0 through P_n, where ‘n’ is order of curve. The initial and

end control points are commonly the end points of the curve; but, the intermediate control points normally do not lie on the curve.

(i) Linear Bezier curves



2.5. Linear Bezier curve

As shown in the figure 2.5, the given points P_0 and P_1, a linear Bezier curve is merely a straight line between those two points. The Bezier curve is represented by

$$\mathbf{B}(t) = \mathbf{P}_0 + t(\mathbf{P}_1 - \mathbf{P}_0) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \ , \ t \in [0,1]$$

And it is similar to linear interpolation.

(ii) Quadratic Bezier curves

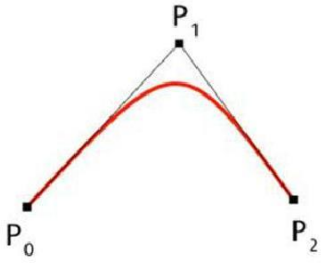


Fig.2.6. Quadratic Bezier curve

As shown in the figure 2.6, a quadratic Bezier curve is the path defined by the function B(*t*), given points P_0, P_1, and P_2,

$$\mathrm{B}(t) = (1-t)[(1-t)\mathrm{P}_0 + t\mathrm{P}_1] + t[(1-t)\mathrm{P}_1 + t\mathrm{P}_2] \ , \ t \in [0,1],$$

This can be interpreted as the linear interpolate of respective points on the linear Bezier curves from P_0 to P_1 and from P_1 to P_2 respectively. Reshuffle the preceding equation gives:

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2 \ , \ t \in [0,1].$$

The derivative of the Bezier curve with respect to the value 't' is

$$\mathbf{B}'(t) = 2(1-t)(\mathbf{P}_1 - \mathbf{P}_0) + 2t(\mathbf{P}_2 - \mathbf{P}_1).$$

From which it can be finished that the tangents to the curve at P_0 and P_2 intersect at P_1. While 't' increases from zero to one, the curve departs from P_0 in the direction of P_1, then turns to land at P_2 from the direction of P_1.

The following equation is a second derivative of the Bezier curve with respect to 't':

$$\mathbf{B}''(t) = 2(\mathbf{P}_2 - 2\mathbf{P}_1 + \mathbf{P}_0).$$

A quadratic Bezier curve is represent a parabolic segment. Since a parabola curve is a conic section, a few sources refer to quadratic Beziers as 'conic arcs'.

(iii) Cubic Bezier curves

As shown in figure 2.7, four control points P_0, P_1, P_2 and P_3 in the higher-dimensional space describe as a Cubic Bezier curve. The curve begins at P_0 going on the way to P_1 and reaches at P_3 coming from the direction of P_2. Typically, it will not pass through control points P_1 / P_2, these points are only there to give directional data. The distance between P_0 and P_1 determines 'how fast' and 'how far' the curve travels towards P_1 before turning towards P_2.

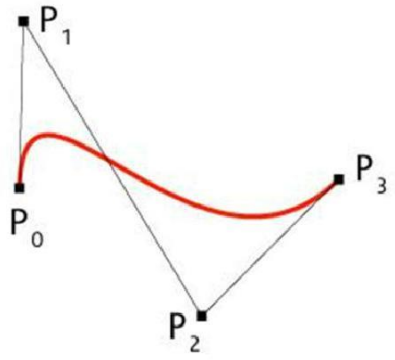


Fig.2.7. Cubic Bezier curve

The function $\mathbf{B}_{\mathbf{P_i}, \mathbf{P_j}, \mathbf{P_k}}(t)$ for the quadratic Bezier curve written by points $\mathbf{P_i}$, $\mathbf{P_j}$, and $\mathbf{P_k}$, the cubic Bezier curve can be described as a linear blending of two quadratic Bezier curves:

$$\mathbf{B}(t) = (1-t)\mathbf{B}_{\mathbf{P_0},\mathbf{P_1},\mathbf{P_2}}(t) + t\mathbf{B}_{\mathbf{P_1},\mathbf{P_2},\mathbf{P_3}}(t) \ , \ t \in [0,1].$$

The open form of the curve is:

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3 \ , \ t \in [0,1].$$

For several choices of P_1 and P_2 the Bezier curve may meet itself.

Any sequence of any four dissimilar points can be changed to a cubic Bezier curve that goes via all four points in order. Given the beginning and ending point of a few cubic Bezier curve, and the

points beside the curve equivalent to t = 1/3 and t = 2/3, the control points for the original Bezier curve can be improved.

The following equation represent first derivative of the cubic Bezier curve with respect to t:

$$B'(t) = 3(1-t)^2(P_1 - P_0) + 6(1-t)t(P_2 - P_1) + 3t^2(P_3 - P_2).$$

The following equation represent second derivative of the Bezier curve with respect to t:

2.4.1. Properties Bezier curve

- The Bezier curve starts at P_0 and ends at P_n; this is known as 'endpoint interpolation' property.
- The Bezier curve is a straight line when all the control points of a cure are collinear.
- The beginning of the Bezier curve is tangent to the first portion of the Bezier polygon.
- A Bezier curve can be divided at any point into two sub curves, each of which is also a Bezier curve.
- A few curves that look like simple, such as the circle, cannot be expressed accurately by a Bezier; via four piece cubic Bezier curve can similar a circle, with a maximum radial error of less than one part in a thousand (Fig.2.8).

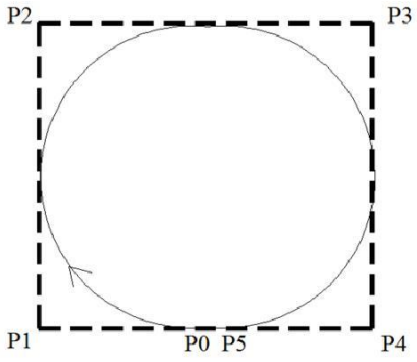
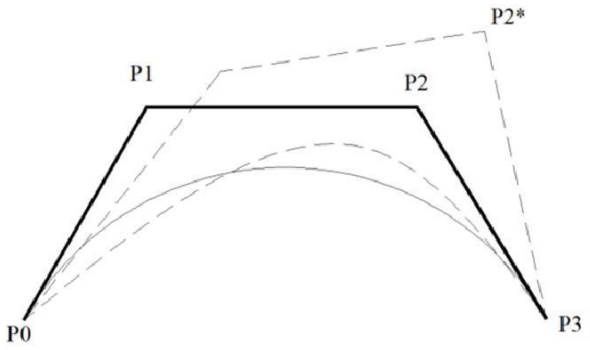


Fig.2.8. Circular Bezier curve

- Each quadratic Bezier curve is become a cubic Bezier curve, and more commonly, each degree 'n' Bezier curve is also a degree 'm' curve for any $m > n$.
- Bezier curves have the different diminishing property. A Bezier curves does not 'ripple' more than the polygon of its control points, and may actually 'ripple' less than that.

- Bezier curve is similar with respect to t and (1-t). This represents that the sequence of control points defining the curve can be changes without modify of the curve shape.
- Bezier curve shape can be edited by either modifying one or more vertices of its polygon or by keeping the polygon unchanged or simplifying multiple coincident points at a vertex (Fig .2.19).



2.9. Bezier curve shape

2.4.2. Construction of Bezier curves

(i) Linear curves:

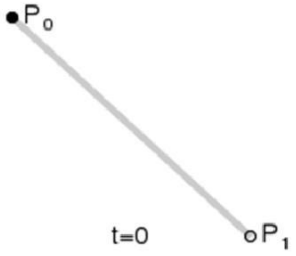


Fig.2.10. Construction of linear Bezier curve

The figure 2.10 shows the function for a linear Bezier curve can be via of as describing how far B(*t*) is from P_0 to P_1 with respect to 't'. When t equals to 0.25, B(*t*) is one quarter of the way from point P_0 to P_1. As 't' varies from 0 to 1, B(*t*) shows a straight line from P_0 to P_1.

(ii) Quadratic curves

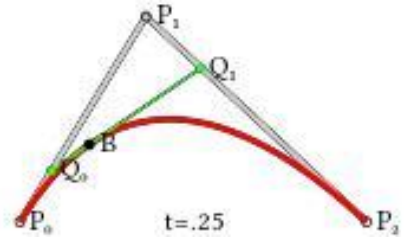


Fig.2.11. Construction of linear Quadratic curve

As shown in figure 2.11, a quadratic Bezier curves one can develop by intermediate points Q_0 and Q_1 such that as 't' varies from 0 to 1:

- Point Q_0 (t) modifying from P_0 to P_1 and expresses a linear Bezier curve.
- Point Q_1 (t) modifying from P_1 to P_2 and expresses a linear Bezier curve.
- Point B (t) is interpolated linearly between Q_0(t) to Q_1(t) and expresses a quadratic Bezier curve.

(iii) Higher-order curves

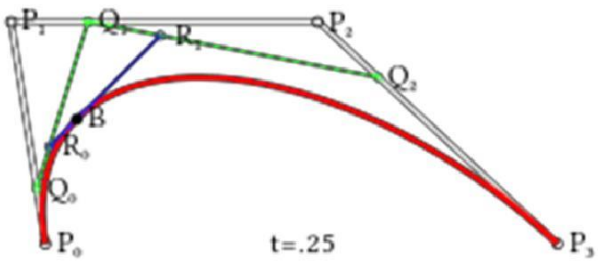


Fig.2.12. Construction of Higher-order curve

As shown in figure 2.12, a higher-order curves one requires correspondingly higher intermediate points. For create cubic curves, intermediate points Q_0, Q_1, and Q_2 that express as linear Bezier curves, and points R_0 and R_1 that express as quadratic Bezier curves.

2.4.3. Rational Bezier curve

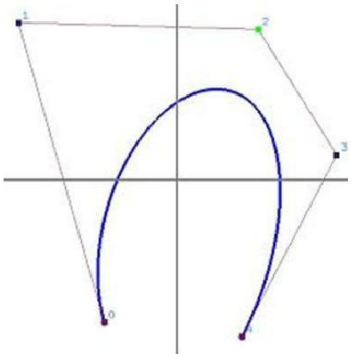


Fig.2.13. Rational Bezier Curve

The rational Bezier curve includes variable weights (w) to provide closer approximations to arbitrary shapes. For Rational Bezier Curve, the numerator is a weighted Bernstein form Bezier and the denominator is a weighted sum of Bernstein polynomials. Rational Bezier curves can be used to signify segments of conic sections accurately, including circular arcs (Fig.2.13).

UNIT III VISUAL REALISM

Hidden – Line-Surface-Solid removal algorithms – shading – colouring – computer animation.

Visual Realism

3.1. Introduction

Visual Realism is a method for interpreting picture data fed into a computer and for creating pictures from difficult multidimensional data sets. Visualization can be classified as :

- Visualization in geometric modeling
- Visualization in scientific computing.

Visualization in geometric modeling is helpful in finding connection in the design applications. By shading the parts with various shadows, colors and transparency, the designer can recognize undesired unknown interferences. In the design of complex surfaces shading with different texture characteristics can use to find any undesired quick modifications in surface changes.

Visualization in computing is viewed as a technique of geometric modeling. It changes the data in numerical form into picture display, allowing users to view their simulations and computations. Visualization offers a process of seeing the hidden. Visualization in scientific computing is of great interest to engineers during the design process.

Existing visualization methods are:

- Parallel projections
- Perspective projection.
- Hidden line removal
- Hidden surface removal
- Hidden solid removal
- Shaded models

Hidden line and surface removal methods remove the uncertainty of the displays of 3D models and is accepted the first step towards visual realism. Shaded images can only be created for surface and

solid models. In multiple step shading process, the first step is removing the hidden surfaces / solids and second step is shades the visible area only. Shaded images provide the maximum level of visualization.

The processes of hidden removal need huge amounts of computing times and also upper end hardware services. The creation and maintenance of such a models are become complex. Hence, creating real time images needs higher end computers with the shading algorithms embedded into the hardware.

3.2. Hidden line removal

Hidden line removal (HLR) is the method of computing which edges are not hidden by the faces of parts for a specified view and the display of parts in the projection of a model into a 2D plane. Hidden line removal is utilized by a CAD to display the visual lines. It is considered that information openly exists to define a 2D wireframe model as well as the 3D topological information. Typically, the best algorithm is required for viewing this information from an available part representation.

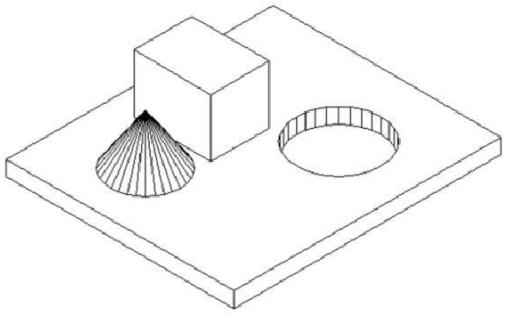


Fig.3.1. Hidden line removal

3D parts are simply manufactured and frequently happen in a CAD design of such a part. In addition, the degrees of freedom are adequate to show the majority of models and are not overwhelming in the number of constraints to be forced. Also, almost all the surface-surface intersections and shadow computations can be calculated analytically which results in significant savings in the number of computations over numerical methods.

3.2.1. Priority algorithm

Priority algorithm is basis on organization all the polygons in the view according to the biggest Z-coordinate value of each. If a face intersects more than one face, other visibility tests besides the Z-depth required to solve any issue. This step comprises purposes of wrapper.

Imagines that objects are modeled with lines and lines are generated where surfaces join. If only the visible surfaces are created then the invisible lines are automatically removed.

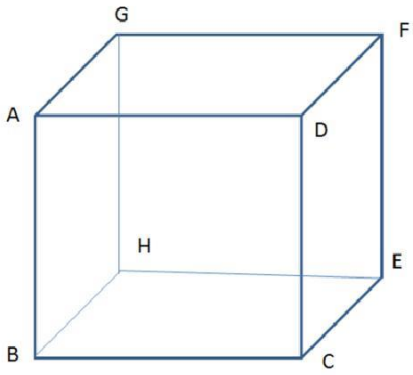


Fig.3.2. Priority algorithm

Face	Priority
ABCE	1
ADFG	1
DCEF	1
ABHG	2
EFGH	2
BCEH	2

ABCD, ADFG, DCEF are given higher priority-1. Hence, all lines in this faces are visible, that is, AB, BC, CD, DA, AD, DF, FG, AG, DC, CE, EF and DF are visible.

AGHB, EFGH, BCEH are given lower priority-2. Hence, all lines in this faces other than priority-1 are invisible, that is BH, EH and GH. These lines must be eliminated.

3.3. Hidden surface removal

The hidden surface removal is the procedure used to find which surfaces are not visible from a certain view. A hidden surface removal algorithm is a solution to the visibility issue, which was one of the first key issues in the field of three dimensional graphics. The procedure of hidden surface identification is called as hiding, and such an algorithm is called a 'hider'. Hidden surface identification is essential to render a 3D image properly, so that one cannot see through walls in virtual reality.

Hidden surface identification is a method by which surfaces which should not be visible to the user are prohibited from being rendered. In spite of benefits in hardware potential there is still a requirement for difficult rendering algorithms. The accountability of a rendering engine is to permit for bigger world spaces and as the world's size approaches infinity the rendering engine should not slow down but maintain at constant speed.

There are many methods for hidden surface identification. They are basically a work out in sorting, and generally vary in the order in which the sort is executed and how the problem is subdivided. Sorting more values of graphics primitives is generally done by divide.

3.3.1. Z - buffer algorithm

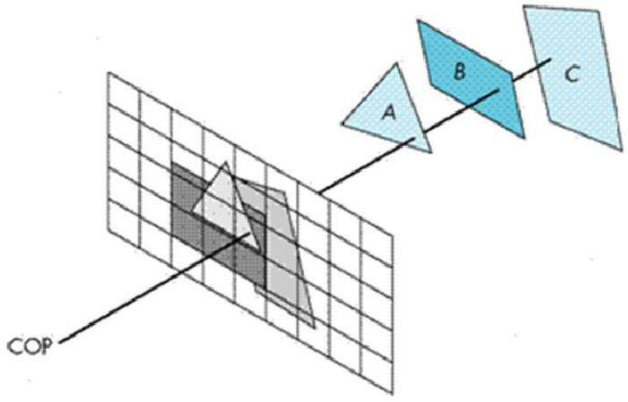


Fig.3.3. Z- buffer algorithm

In Z-buffering, the depth of 'Z' value is verified against available depth value. If the present pixel is behind the pixel in the Z-buffer, the pixel is eliminated, or else it is shaded and its depth value changes the one in the Z-buffer. Z-buffering helps dynamic visuals easily, and is presently introduced effectively in graphics hardware.

- Depth buffering is one of the easiest hidden surface algorithms
- It keeps follow of the space to nearest object at every pixel position.

- Initialized to most negative z value.
- when image being drawn, if its z coordinate at a position is higher than z buffer value, it is drawn, and new z coordinate value is stored; or else, it is not drawn
- If a line in three dimensional is being drawn, then the middle z values are interpolated: linear interpolation for polygons, and can calculate z for more difficult surfaces.

Algorithm:

```
loop on y;
        loop on x;
                zbuf[x,y] = infinity;
loop on objects
        {
        loop on y within y range of this object
                {
                loop on x within x range of this scan line of this object
                        {
                        if z(x,y) < zbuf[x,y] compute z of this object at this pixel &
                                test zbuf[x,y] = z(x,y) update z-buffer
                                        image[x,y] = shade(x,y) update image (typically RGB)
                        }
                }
        }
```

Basic operations:

1. compute y range of an object
2. compute x range of a given scan line of an object
3. Calculate intersection point of a object with ray through pixel position (x,y).

3.3.2. Painter's algorithm

The painter's algorithm is called as a priority fill, is one of the easiest results to the visibility issue in three dimensional graphics. When projecting a 3D view onto a 2D screen, it is essential at various points to be finalized which polygons are visible, and which polygons are hidden.

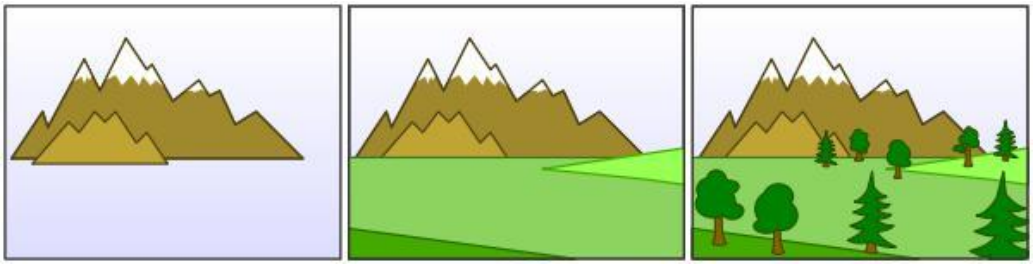


Fig.3.4. Painter's algorithm

The 'painter's algorithm' shows to the method employed by most of the painters of painting remote parts of a scene before parts which are close thereby hiding some areas of distant parts. The painter's algorithm arranges all the polygons in a view by their depth and then paints them in this order, extreme to closest. It will paint over the existing parts that are usually not visible hence solving the visibility issue at the cost of having painted invisible areas of distant objects. The ordering used by the algorithm is referred a 'depth order', and does not have to respect the distances to the parts of the scene: the important characteristics of this ordering is, somewhat, that if one object has ambiguous part of another then the first object is painted after the object that it is ambiguous. Thus, a suitable ordering can be explained as a topological ordering of a directed acyclic graph showing between objects.

Algorithm:

```
sort objects by depth, splitting if necessary to handle intersections;
    loop on objects (drawing from back to front)
        {
        loop on y within y range of this object
            {
            loop on x within x range of this scan line of this object
                {
                image[x,y] = shade(x,y);
                }
            }
        }
```

Basic operations:

1. compute 'y' range of an object
2. compute 'x' range of a given scan line of an object
3. compute intersection point of a given object with ray via pixel point (x,y).

4. evaluate depth of two objects, determine if A is in front of B, or B is in front of A, if they don't overlap in xy, or if they intersect
5. divide one object by another object

Advantage of painter's algorithm is the inner loops are quite easy and limitation is sorting operation.

3.3.3. Warnock algorithm

The Warnock algorithm is a hidden surface algorithm developed by John Warnock that is classically used in the area of graphics. It explains the issues of rendering a difficult image by recursive subdivision of a view until regions are attained that is trivial to evaluate. Similarly, if the view is simple to compute effectively then it is rendered; else it is split into tiny parts which are likewise evaluated for simplicity. This is a algorithm with run-time of O(np), where *p* is the number of pixels in the viewport and *n* is the number of polygons.

The inputs for Warnock algorithm are detail of polygons and a viewport. The good case is that if the detail of polygons is very simple then creates the polygons in the viewport. The continuous step is to divide the viewport into four equally sized quadrants and to recursively identify the algorithm for each quadrant, with a polygon list changed such that it contains polygons that are detectable in that quadrant.

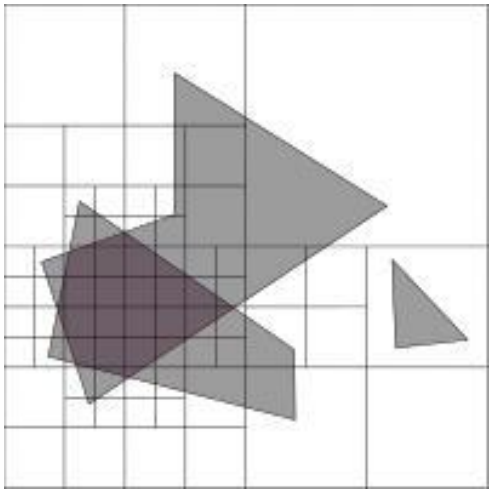


Fig.3.5. Warnock algorithm

1. Initialize the region.
2. Generate list of polygons by sorting them with their z values.
3. Remove polygons which are outside the area.
4. Identify relationship of each polygon.
5. Execute visibility decision analysis:

    a) Fill area with background color if all polygons are disjoint,

    b) Fill entire area with background color and fill part of polygon contained in area with color of polygon if there is only one contained polygon,

    c) If there is a single surrounding polygon but not contained then fill area with color of surrounding polygon.

    d) Set pixel to the color of polygon which is closer to view if region of the pixel (x,y) and if neither of (a) to (d) applies calculate z- coordinate at pixel (x,y) of polygons.
6. If none of above is correct then subdivide the area and Go to Step 2.

3.4. Hidden Solid Removal

The hidden solid removal issue involves the view of solid models with hidden line or surface eliminated. Available hidden line algorithm and hidden surface algorithms are useable to hidden solid elimination of B-rep models.

The following techniques to display CSG models:

1. Transfer the CSG model into a boundary model.
2. Use a spatial subdivision strategy.
3. Based on ray sorting.

3.4.1. Ray-Tracing algorithm

A ray tracing is a method for creating an image by tracing the path of light via pixels in an image plane and reproducing the effects of its meets with virtual objects. The procedure is capable of creating a high degree of visual realism, generally higher than that of usual scan line techniques, but at a better computational. This creates ray tracing excellent suited for uses where the image can be rendered gradually ahead of time, similar to still images and film and TV visual effects, and more badly suited for real time environment like video games where speed is very important. Ray tracing is simulating a wide range of optical effects, such as scattering, reflection and refraction.

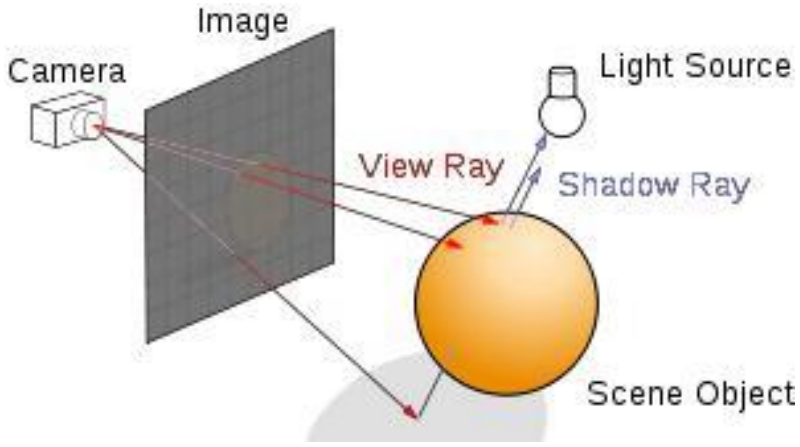


Fig.3.6. Ray-Tracing algorithm

Ray-Tracing algorithm

```
For every pixel in image
{
  Generate ray from eye point passing via this pixel
  Initialize Nearest 'T' to 'INFINITY'
 Initialize Nearest Object to NULL
  For each object in scene
{
    If ray intersects this image
            {
             If t of intersection is less than Nearest T
                        {
                        Set Nearest T to t of the intersection
                        Set Nearest image to this object
                        }
            }
  }
  If Nearest image is NULL
            {
              Paint this pixel with background color
            }
```

Else

{

Shoot a ray to every light source to check if in shadow

If surface is reflective, generate reflection ray

If transparent, generate refraction ray

Apply Nearest Object and Nearest T to execute shading function

Paint this pixel with color result of shading function

}

}

Optical ray tracing explains a technique for creating visual images constructed in three dimensional graphics environments, with higher photorealism than either ray casting rendering practices. It executes by tracing a path from an imaginary eye via every pixel in a virtual display, and computing the color of the object visible via it.

Displays in ray tracing are explained mathematically by a programmer. Displays may also incorporate data from 3D models and images captured like a digital photography.

In general, every ray must be tested for intersection with a few subsets of all the objects in the view. Once the nearest object has been selected, the algorithm will calculate the receiving light at the point of intersection, study the material properties of the object, and join this information to compute the finishing color of the pixel. One of the major limitations of algorithm, the reflective or translucent materials may need additional rays to be re-cast into the scene.

Advantages of Ray tracing:

1. A realistic simulation of lighting over other rendering.
2. An effect such as reflections and shadows is easy and effective.
3. Simple to implement yet yielding impressive visual results.

Limitation of ray tracing:

Scan line algorithms use data consistency to divide computations between pixels, while ray tracing normally begins the process a new, treating every eye ray separately.

3.5. Shading

Shading defines to describe depth perception in three dimensioning models by different levels of darkness. Shading is applied in drawing for describes levels of darkness on paper by adding media heavy densely shade for darker regions, and less densely for lighter regions.

There are different techniques of shading with cross hatching where perpendicular lines of changing closeness are drawn in a grid pattern to shade an object. The closer the lines are combining, the darker the area appears. Similarly, the farther apart the lines are, the lighter the area shows.

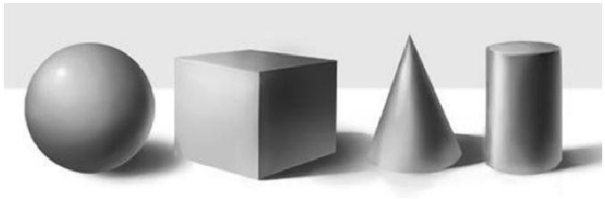


Fig.3.7. Shading

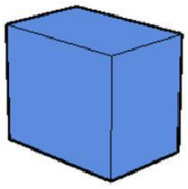


Fig.3.8. Image with edge lines

The image shown in figure 3.8 has the faces of the box rendered, but all in the similar color. Edge lines have been rendered here as well which creates the image easier to view.

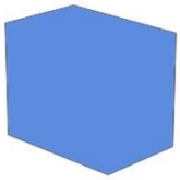


Fig.3.9. Image without edge lines

The image shown in figure 3.9 is the same model rendered without edge lines. It is complicated to advise where one face of the box ends and the next starts.

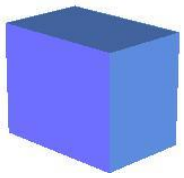


Fig.3.10. Image with Shading

The image shown in figure 3.10 has shading enabled which makes the image extra realistic and makes it easier to view which face is which.

3.5.1. Shading techniques:

In computer graphics, shading submits to the procedure of changing the color of an object in the 3D view, a photorealistic effect to be based on its angle to lights and its distance from lights. Shading is performed through the rendering procedure by a program called a 'Shader'. Flat shading and Smooth shading are the two major techniques using in Computer graphics.

Assembly modelling – interferences of positions and orientation – tolerance analysis-massproperty calculations – mechanism simulation and interference checking.

Assembly of parts

4.1. Introduction

In today's global situation, two main things are significant for the industry: cost reduction and environment protection. Since the late 70's it has been developed that the assembly procedure normally signify one third of the product cost. Hence, it is essential to design appropriate plans for parts assembly: manufacturing, and disassembly: recycling.

A realistic assembly procedure can increase efficiency, cost reduction and improve the recycling of product. To overcome these problems, various simulations based on digital mock-ups of products are required. Even though modeling and analysis software, presently applied at various stages of the Product Development Process, can suggest results to several of the above stated needs, the progress of a committed assembly and disassembly combine simulation stage is still a need.

To attain an optimum assembly method, various complex software for assembly analysis and, as well as simulation programs based on multi agent methods or which apply contact data between assembly components, were created. Newly, Virtual Reality (VR) has broadly developed towards Assembly realistic simulation.

As the contact between objects is at the basis of the assembly simulations need 3D objects shapes, the contact detection is addressed here as the first step in the Assembly simulation process. The equivalent procedure establishes links between shapes, contact mock-ups and component kinematics, which gives a basic set of meaningful data

All mechanical parts are applying one of the common CAD modelers. Thus, the existing assembly modules of 3D CAD software and their definite method to modeling assemblies have a tough influence on how products are calculated. Also, for the realistic simulation, the data exchange CAD to Virtual Reality is one of the significant problems presently faced by the virtual prototyping community.

4.2. Assembly modeling

Assembly modeling is a technique applied by CAD and product visualization software systems to utilize multiple files that shows components within a product. The components within an assembly are called as solid / surface models.

The designer usually has approach to models that others are functioning on concurrently. For example, different people may be creating one machine that has different components. New parts are

extra to an assembly model as they are generated. Every designer has approach to the assembly model, during a work in progress, and while working in their own components. The design development is noticeable to everyone participated. Based on the system, it might be essential for the users to obtain the most recent versions saved of every individual component to update the assembly.

The personal data files defining the 3D geometry of personal components are assembled together via a number of sub assembly levels to generate an assembly explaining the complete product. Every CAD methods support the bottom-up construction. A few systems, through associative copying of geometry between components allow top-down construction. Components can be situated within the assembly applying absolute coordinate position methods.

Mating conditions are defines of the relative location of mechanism between each other; for example axis position of two holes or distance between two faces. The final place of all objects based on these relationships is computing using a geometry constraint engine built into the CAD package.

The significance of assembly modeling in obtaining the full advantages of Product Life-cycle Management has directed to ongoing benefits in this technology. These contain the benefit of lightweight data structures that accept visualization of and interaction with huge amounts of data related to product, interface between PDM systems and active digital mock up method that combine the skill to visualize the assembly mock up with the skill to design and redesign with measure, analyze and simulate.

4.2.1. Assembly Concepts

When components are additional to an assembly, parent and child relationships are created. These relationships are displayed by graphically as an assembly tree. Parts are parametrically connected by position constraints. These constraints have data about how a part should be placed within the assembly hierarchy and how it should respond if other components are edited.

Functioning within the framework of an assembly is prepared easier by accepting to apply more commands to other parts and sub-assemblies. These contain the Annotation Text, Inquire, Point, Datum Plane and Pattern Component commands. Bigger assembly performance is improved by removing unwanted redraws and improved display management while zooming.

Assembly models have additional data than simply the sum of their components. With assembly modeling interference verifies between parts and assembly specific data such as mass properties.

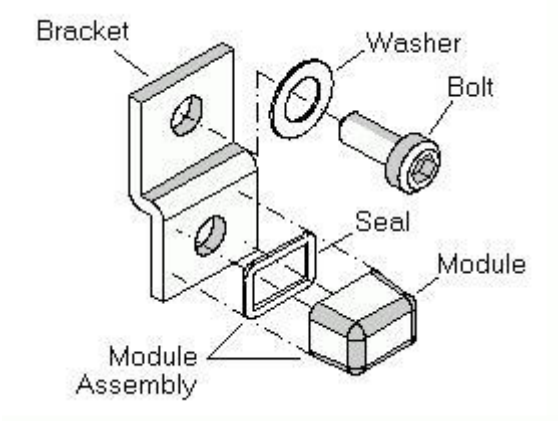


Fig.4.1. Assembly of parts

4.2.2. Bottom up Assembly design

In a 'bottom up' assembly design, complex assemblies are divided into minor subassemblies and parts. Every part is considered as individual part by one or more designers. The parts can be archived in a **library** in one or more 3D Files. This is the high effective way to generate and manage complex assemblies.

Every part is included into the active part making a component request and thus an assembly. The component will be the child of the active part and then it will be the active part. Hence an instance of the actual part is applied; it revises automatically if the archived part is edited by activating.

Bottom up Hierarchy:

The 'bottom up' assembly design hierarchy of the basic assembly is shown in figure 4.2. All the parts exist prior to **Part1**. When **Part1** is generated, it becomes the active. It would utilize the menu sequence to add **Bracket** and it becomes the active part.

Insert > Component

Or

Assembly Design Tool Bar >

As per example shown in figure 4.2., 'Bracket' is a child of **Part1.** The dashed line represents that 'Bracket' exists in the 3D file **Parts Z3**. The dotted line represents that '**Bracket'** is inserted into **Part-1**. After **Bracket** is added, **Part1** is **redefined**. **Bolt** and **Washer** are then added the same process and **Part-1** is **reactivated** again.

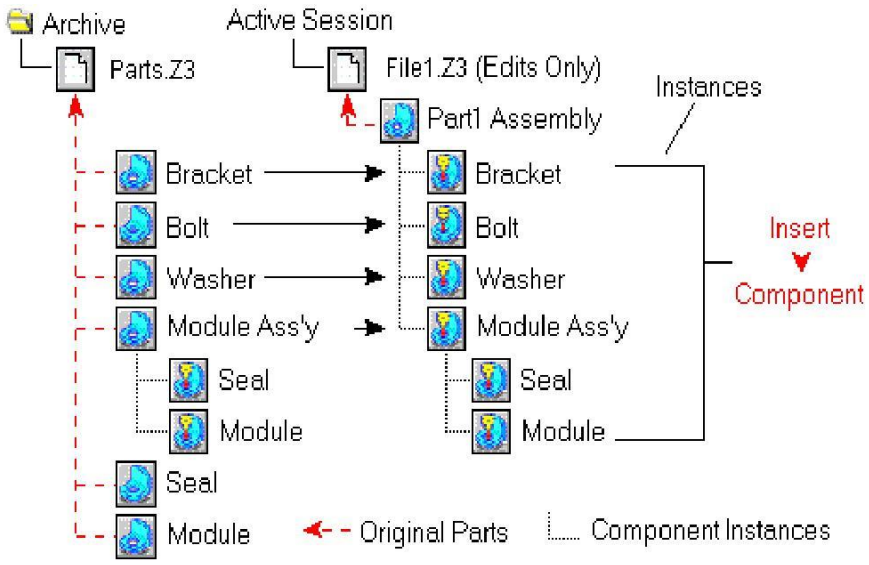


Fig.4.2. Bottom up Design – Part 1

Module of subassembly is added similar as '**Bracket', 'Bolt',** and '**Washer'** again becoming a child of Part-**1**. But, because **Module Subassembly** already has the two items **Seal** and **Module**, they are added and continue as its children.

Sequence of operations (Fig. 4.2.):

- **File-1** has 1 part.
- **Part-1** has 4 components.
- **Module Subassembly** has 2 components.
- All of the items are illustrations of the original parts that reside in the ZW3D file **Parts Z3.**
- If **File-1** is eliminated from the active assembly before it is saved and **Part1** are removed. The original parts placed in the file **Parts Z3** are not changed.
- If **File-1** is **saved** and **Part1** is also saved.
- If **File-1** is erased and **Part1** is also erased.

4.2.3. Top down Assembly Design

In a 'top down' assembly design all parts are classically designed by the similar person within a single part. 3D assembly handles 'top down' method by allowing to **design and creation of** a component while work in the active part. Hence, the active part will be an assembly part.

The part becomes a child of the active part and then it will be the active part. The part, when generated, is an instance of a base part which will be a root object located in the active file. Every part is activated and modified as needed. The 'top down' assembly design has its benefits. If the project is terminated or to go in a different new direction, removing the file will remove the part and all of its

components.

Top down Hierarchy

The 'top down' assembly method is shown in a figure 4.3 and one of the components exist prior to **Part-1**. When **Part-1** is generated, it will be the active part. The following command sequence to generate Bracket and create it the active part.

Assembly Design Tool Bar >

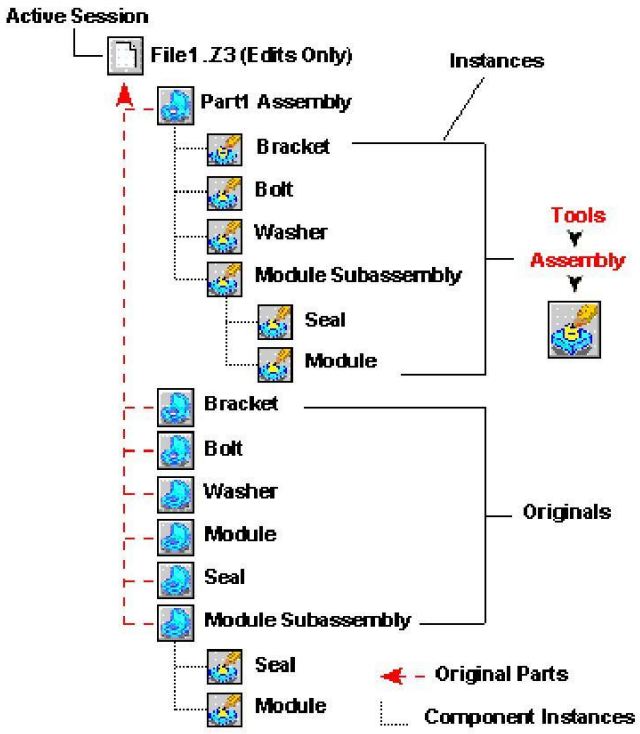


Fig.4.3. Top down Design – Part 1

Bracket is a child of **Part-1.** The dashed line illustrates that by default when **Bracket** is generated; it is attached to **File-1**. The dotted line illustrates that **Bracket** is attached into **Part-1**. When **Bracket** is executed **Part1** is **reactivated**. **Bolt** and **Washer** are then generated using the similar process and **Part-1** is **reactivated** again.

Subassembly Module is generated like the **Bracket, Bolt,** and **Washer** again will be a child of **Part1.** But, **Module Subassembly** remains active when **seal** is developed. Seal will be the active part and by default also exists in **File-1** but is inserted into **Module Subassembly** hence it was active at the

time of s**eal** was created. **Subassembly Module** is then **reactivated** and **Module** is generated like a **Seal.**

Sequence of operations (Fig 4.3):

- File-1 has 7.
- Part-1 contains 4 components, which are illustrations of the basic parts located in File-1.

- Subassembly Module contains 2 components which are also illustrations of the basic parts located in File-1.
- If File-1 is saved it has all of its original parts.
- If File-1 is erased, it and all of its basic parts are erased.

4.3. Interference of position and orientation

Designers and manufacturers should check jointly that a provided product can be assembled, without interference between parts, before the product to be manufactured. Similarly, all the CAD tools presently have the potential to directly analyze the possibility of a specified assembly plan for a product.

An assessment of previous assembly sequence and optimization research explains that most previous assembly planners apply either feature-mating or interference-free techniques to find assembly part interference interaction. In both feature-mating and interference-free techniques focused upon the basic geometrical data and restrictions for the designed product, which are generally contained in connected CAD files.

When completely automate the procedure of creating a professional assembly plan, geometrical information for CAD models should be automatically taken from CAD files, analyzed for interference relationships between components in the assembly, and then designed for utilized the assembly analysis tools. Most of the previous assembly sequence planners do not have the potential to complete the three tasks; they need users to manually input part attributes or interference data, which is so time-consuming.

4.3.1. Determining Interference Relationships between Parts

In automated assembly schemes, most parts are assembled along with the principal axis. Hence, to fine interference between parts while assembly, the projected technique referred six assembly directions along with the principal assembly axis: $+x$, $-x$, $+y$, $-y$, $+z$, and $-z$. But, the method could be improved, to think other assembly directions, as required. The projected system uses projection of part coordinates onto planes in three principal axis (x, y ,z) to find the obstruction between parts sliding along some of the six principal assembly axis. The projections overlap between any two parts in a specified axis direction shows a potential interference between the two parts, when one of the two parts slides along the specified direction, with respect to the other. Vertex coordinates for overlapped

projections are then evaluated to find if real collisions would happen between parts with overlapped projections. The planned process stores the determined interference data for allocated assembly direction in a group of interference free matrices, for compatibility with previous planners of assembly.

The swept volume interference and the multiple interference detection systems are appropriate for three-dimensional interference determination between B-REP entities. But, both techniques were developed for real-time interference detection between two moving parts in a simulation environment. As a result, these two techniques are expensive in computationally. For the assembly planning issue, actual collision finding capacity along subjective relative motion vectors is not require. Instead, a efficient computational technique is required for finding if two parts will collide when they are assembled in a specified order along any one of the six principle assembly axis.

4.2.3. Interference-free matrix

An interference-free matrix shows interference between two components, when one component is moved, in a given assembly direction, into an assembled location, with another component already in an assembled location. Assembly actions that result in interferences are denoted as '0' in the matrix, and assembly actions that do not result in interferences are denoted as '1' in the matrix.

As shown in figure 4.4., the interference-free matrix of an assembly having three parts, for assembly movement sliding from infinity of negative toward infinity of positive along the +x direction is as follows:

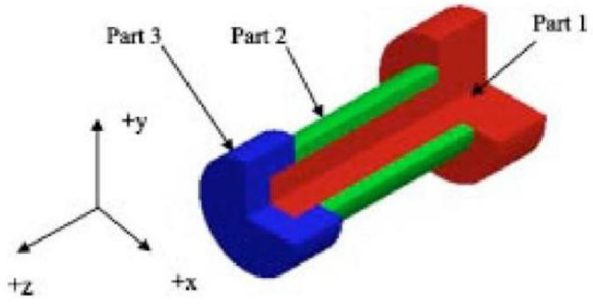


Fig.4.4. Interference of three parts

Interference-free matrix for sliding in the +*x* direction:

$$\begin{array}{c} \text{Part} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

The row in the Interference-free matrix indicate the components being shifted during a given assembly operation, and the column indicate the parts that have previously been assembled. Hence, since matrix element (2, 1) is equal to '0', if Part-1 is assembled initially, and after that Part-2 is assembled in the direction of +*x*, Part-2 will collide with Part-1. Similarly, matrix element (1, 2) is equal to '1', if Part-2 is assembled initially, and then Part-1 is assembled in the direction of +*x*, Part-1 will not collide with Part-2. As a part cannot be assembled after itself, all elements in the diagonal matrix are set to '0'. As a whole, six matrices are utilized to show interference relationships between parts in the six principal axes. When robotically creating interference-free matrices, the projected algorithm finds matrix elements row by row. When two parts would interfere through assembly in a given direction, the program inserts a '0' in the corresponding matrix position; or inserts as a '1'.

4.4. Geometric Tolerance

The function of geometric tolerance is to explain the engineering objective of components and assemblies. The datum reference frame can explain how the part. Tolerance can accurately define the dimensional needs for a part, permitting over 50% more tolerance than coordinate dimensioning in a few cases. Suitable purpose of tolerance will confirm that the part described on the drawing has the preferred form, fit and purpose with the highest possible tolerances (Fig.4.5).

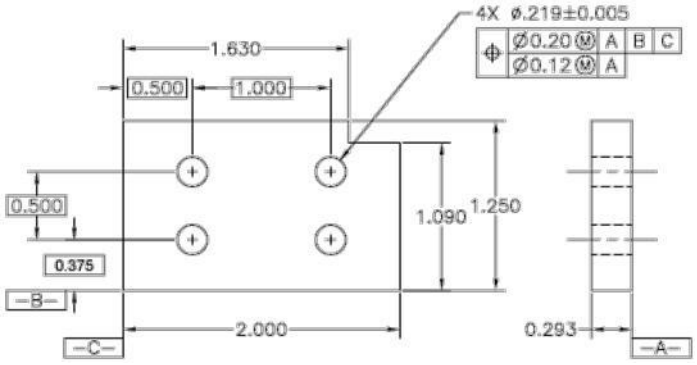


Fig.4.5. Geometric Tolerance

4.3.1. Fundamental rules for Geometric Tolerance

1. All dimensions should have a tolerance. Each attribute on every manufactured component is subject to change; hence, the limits of acceptable difference must be defined. Plus and minus tolerances

may be used to dimensions from a common tolerance block.

2. Dimensions describe the geometry and allowable change. Measurement and scaling of the drawing is not permitted excluding in certain cases.
3. Engineering drawings describe the necessities of completed parts. Each dimension and tolerance needed to define the completed part shall be shown on the drawing. If extra dimensions would be useful, but are not necessary, they may be noted as reference.
4. Dimensions should be used to attributes and arranged in such a way as to show the purpose of the features. In addition, dimensions should not be subject to more than one explanation.
5. Descriptions of manufacturing systems should be avoided. The geometry should be explained without defining the technique of manufacture.
6. If some sizes are needed during manufacturing but are not wanted in the final geometry they should be noticeable as non-mandatory.
7. All dimensioning and tolerance should be placed for utmost readability and should be used to visible lines in true profiles.
8. When geometry is usually restricted by code, the dimension(s) shall be integrated with code number in comments below the dimension.
9. If not openly declared, all dimensions and tolerances are only suitable when the item is in free.
10. Dimensions and tolerances indicate to the full length, width, and depth.

4.3.2. Tolerance Symbols

Symbols for tolerances are bilateral unless otherwise defined. For example, the location of a hole has a tolerance of .020mm. This indicates that the hole can move +/- .010 mm, which is an equal bilateral tolerance. It does not consider that the hole can move +.015/-.005 mm, which is an unequal bilateral tolerance. (Fig.4.6.).

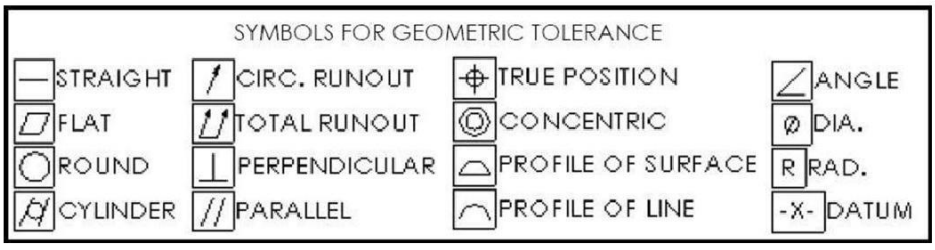


Fig.4.6. Symbols for Geometric Tolerance

4.5. Tolerance Analysis

Tolerance analysis is a title to a different approaches applied in product design to know how deficiencies in parts as they are manufactured, and in assemblies, influence the ability of a product to meet customer needs. Tolerance analysis is a way of accepting how basis of deviation in part dimensions and assembly constraints distribute across parts and assemblies, and how that total deviation

affects the ability of a drawing to reach its design necessities within the process capabilities of organizations and supply chains.

Tolerance openly affects the cost and performance of products. In electrical machines, safety needs that the power supply to be situated a minimum gap from adjacent components, such as one more sheet-metal component, in order to remove electrical short circuits. Tolerance analysis will describe whether the small clearances specified will meet the safety requirement, assigned manufacturing and assembly variability force on the minimum clearance.

4.5.1. Tolerance stack-up

Tolerance stack-up computations show the collective effect of part tolerance with respect to an assembly need. The tolerances 'stacking up' would describe to adding tolerances to obtain total part tolerance, then evaluating that to the existing gap in order to see if the design will work suitably. This simple evaluation is also defined as 'worst case analyses'. Worst case analysis is suitable for definite needs where failure would signify failure for a company. It is also needful and suitable for problems that occupy a low number of parts. Worst case analysis is always carried out in a single direction that is a 1-D analysis. If the analysis has part dimensions that are not parallel to the assembly measurement being defined, the stack-up approach must be edited since 2D variation such as angles, or any variation that is not parallel with the 1-D direction, does not influence the measurement of assembly with a 1-to-1 ratio.

The tolerance stacking issue occurs in the perception of assemblies from interchangeable parts because of the inability to create or join parts accurately according to nominal. Either the applicable part dimension changes around various nominal value from part by part or it is the act of assembly that directs to variation. For example, as two parts are combined through matching holes pair there is not only variation in the location of the holes relative to nominal centers on the parts but also the slippage difference of matching holes relative to each other when safe.

Thus there is the opportunity that the assembly of such interacting parts will not move or won't come closer as planned. This can generally be judged by different assembly criteria, say G1, G2,... Here we will be discussed with just one assembly criterion, say G, which can be noted as a function of the part dimensions L1,...,Ln. A example is shown in Figure 4.7., where n = 6 and is the clearance gap of interest. It finds whether the stack of cogwheels will locate within the case or not. Thus it is preferred to have G > 0, but for performance of functional causes one may also require to limit G.

$$G = L1 - (L2 + L3 + L4 + L5 + L6)$$
$$= L1 - L2 - L3 - L4 - L5 - L6$$

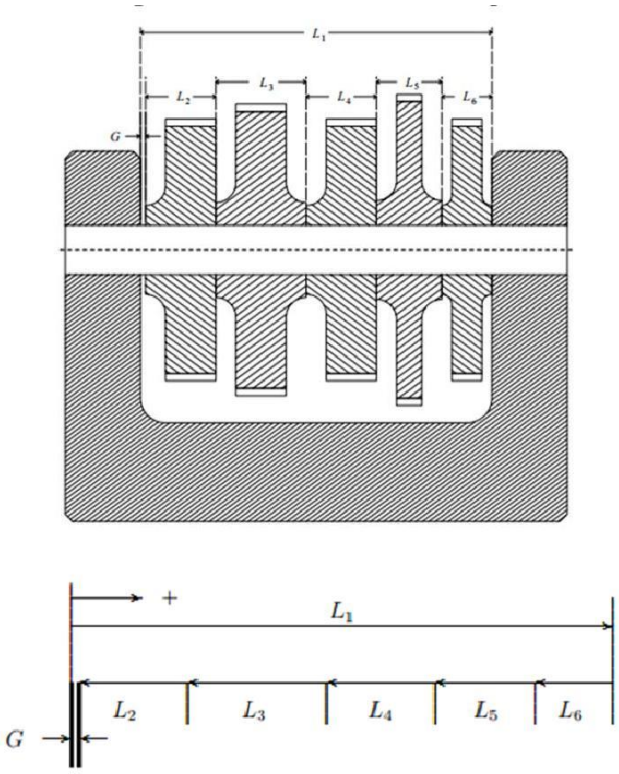


Fig.4.7. Tolerance Stack-up

As per the example, the required lengths 'Li 'may vary from the nominal lengths 'λi' by a small value. If there is higher variation in the 'Li' there may well be important problems in accepting G > 0. Thus it is sensible to limit these changes via tolerances. For similar tolerances, 'Ti', represent an 'upper limit' on the absolute variation between actual and nominal values of the i th detail part dimension, it is means that |Li − λi| ≤ Ti. It is mostly in the interpretation of this last inequality that the different methods of tolerance stacking vary.

The nominal value 'γ'of G is typically computed by replacing in equation L1 − L2 − L3 − L4 − L5 − L6, the actual values of Li's by the corresponding nominal values of λi, that is γ = λ1 − λ2 − λ3 − λ4 − λ5 − λ 6 .

Assuming $|\epsilon_i| = |L_i - \lambda_i| \leq T_i$ for all $i = 1, 2, \ldots, n$ we can bound $|G - \gamma|$ by

$$T_{\text{assy}}^{\text{arith}} = |a_1|T_1 + |a_2|T_2 + \ldots + |a_n|T_n \ .$$

If $|a_i| = 1$ for all i, this simplifies to

$$T_{\text{assy}}^{\text{arith}} = T_1 + T_2 + \ldots + T_n \ .$$

4.5.2. Statistical method for tolerance analysis (RSS) **:**

In RSS method, tolerance stacking a significant new element is added to the assumptions, specifically which the detail differences from nominal are random and independent from part by part. It is expensive in the sense that it frequently commanded very close tolerances. That all variations from nominal should dispose themselves in worst case method to defer the higher assembly tolerance is a relatively unlikely proposition. On the other hand, it had the advantage of assurance the resulting assembly tolerance. Statistical tolerance in its typical form operates under two basic hypotheses:

As per Centered Normal Distribution, somewhat considering that the 'Li' can occur anywhere within the tolerance distribution [λi − Ti, λi + Ti], assume that the 'Li' are normal random variables, that is change randomly according to a normal distribution, centered on that similar interval and with a ±3σ distribute equal to the span of that interval, hence 99.73% of all 'Li' values occur within this gap. As per the normal distribution is such that the 'Li' fall with upper frequency in the middle near 'λi' and with low frequency closer the interval endpoints. The match of the ±3σ distribution with the span of the detail tolerance span is hypothetical to state that almost all parts will satisfy the detail tolerance limits as shown in figure 4.8.

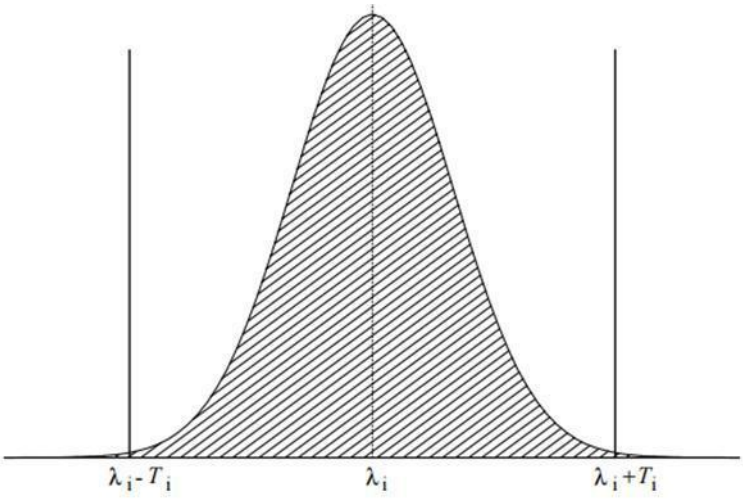


Fig.4.8. Centered Normal Distribution

Statistical tolerance stacking formula is given below:

$$T_{\text{assy}}^{\text{stat}} = \sqrt{a_1^2 T_1^2 + a_2^2 T_2^2 + \ldots + a_n^2 T_n^2}$$

$$= \sqrt{T_1^2 + T_2^2 + \ldots + T_n^2}$$

Where, $a_i = \pm 1$ for all i = 1,...,n.

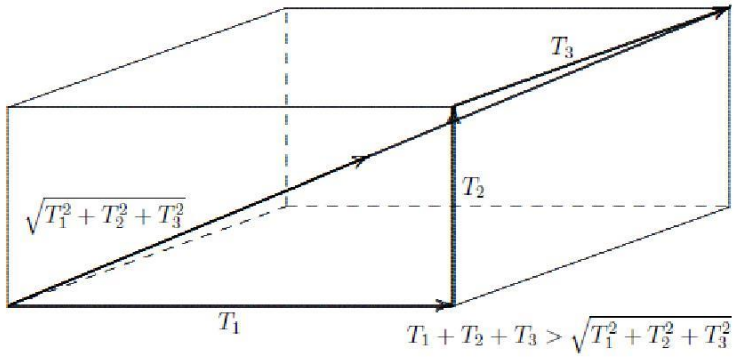


Fig.4.9. RSS cube

Typically $T_{stat\ assy}$ is considerably smaller than $T_{arith\ assy}$. For n=3, the scale of this variation is simply visualized and valued by a rectangular box with side lengths T1, T2 and T3. To obtain from one corner of the box to the diagonally opposite corner, one can cross the gap T21 + T22 + T23 along that diagonal and follow the three edges with lengths T1, T2, and T3 for a total length $T_{arith\ assy}$ = T1 + T2 + T3 as shown in figure 4.9.

4.5.3. Second Order Tolerance Analysis

Due to the manufacturing methods changing for various types of components, the distribution moments vary as well. RSS only applies standard deviation and does not contain the upper moments of skewness and kurtosis that describe the effects tool wear, form aging and other classical manufacturing situations. Second Order Tolerance Analysis includes all types of distribution moments as shown in figure 4.9

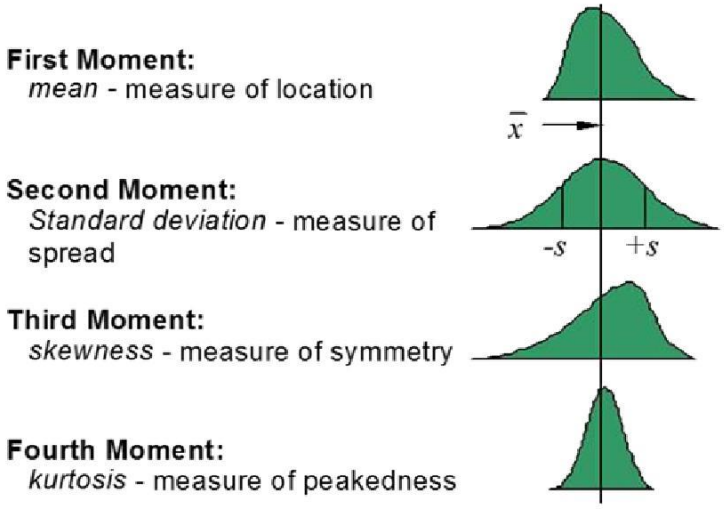


Fig.4.9. Second order Tolerance Analysis

Second Order Tolerance Analysis is required to find what output is going to be when the assembly function is not linear. In classical mechanical engineering developments kinematic changes and other assembly performances result in non-linear assembly operations. Second order estimates are more complex so manual calculations are not suitable but the computation is greatly improved and becomes feasible within tolerance analysis software.

4.5.4. Importance of Tolerance Analysis

With smaller product lifecycles, quicker to market, and higher cost pressures, the uniqueness that distinguishes a product from its competitors. Engineers are moving to the next order of resolution in order to improve cycle time and quality and to reduce costs. They are showing nearer at why they did not get the correct part and assembly dimension values they needed from manufacturing and then are trying to optimize the tolerances on the following version of the product. Optimization of tolerance during design has a high impact on the output of manufacturing, and better yields direct impact on product cost and quality. Tolerance Analysis before trying to manufacture a product helps engineers avoid time taking iterations later in the design cycle.

The electronics industry is attaining customer satisfaction purposes via a physical shrinking of their components while adding more capabilities. As electronic devices high densely packaged, the significance increases to more accurately understanding the interaction of manufacturing variation and tolerances in design. Similarly, in the aircraft, automotive and medical device productions, liability costs are increasing while environmental needs are being more forcefully forced such that companies requires to understand high precisely what may reason a failure.

Advantages of Tolerance Analysis

1. Accurate part assembly.
2. Elimination of assembly rework
3. Improvement in assembly quality.
4. Reduction of assembly cost.
5. High customer satisfaction.
6. Effectiveness of out-sourcing.

Limitations of Tolerance Analysis

1. Time consuming process.
2. Skill require for complex assemblies.

4.6. Mass property calculations

The first step in finding mass properties is to set up the location of the X, Y, and Z axis. The correctness of the calculations will depend completely on the knowledge used in choosing the axis. Hypothetically, these axes can be at any position relative to the object being considered, offered the axes are equally perpendicular. But, in reality, except the axes are chosen to be at a position that can be precisely measured and identified, the calculations are meaningless.

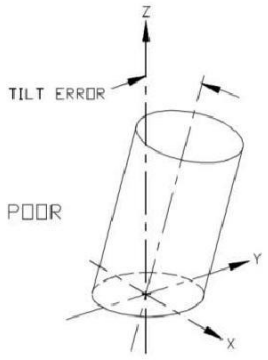


Fig.4.10. Accuracy of axis – Vertical

As shown in the figure 4.10, the axes do not create a best reference hence a small error in squareness of the base of the cylinder origins the object to tilt away from the vertical axis.

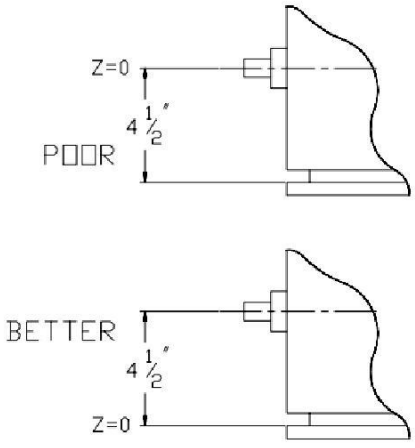


Fig.4.11. Accuracy of axis – Horizontal

An axis should always pass via a surface that is firmly linked with the bulk of the component. As shown in the figure 4.11, it would be best to position the origin (Z=0) at the end of the component rather than the fitting that is freely dimensioned virtual to the end.

4.6.1. Calculating Center of gravity location

The center of gravity of an object is:

- described the 'center of mass' of the object.
- the location where the object would balance.
- the single point where the static balance moments are all zero about three mutually perpendicular axis.
- the centroid of object the volume when the object is homogeneous.
- the point where the total mass of the component could be measured to be concentrated while static calculations.
- the point about where the component rotates in free space
- the point via the gravity force can be considered to perform
- the point at which an exterior force must be used to create translation of an object in space

Center of gravity location is stated in units of length along the three axes (X, Y, and Z). These three components of the vector distance from the base of the coordinate system to the Center of gravity location. CG of composite masses is computed from moments considered about the origin. The essential dimensions of moment are Force and Distance. On the other hand, Mass moment may be utilized any units of Mass times Distance. For homogeneous components, volume moments may also be considered. Care should be taken to be confident that moments for all parts are defined in compatible units.

Component distances for CG position may be either positive or negative, and in reality their polarity based on the reference axis position. The CG of a homogeneous component is determined by determining the Centroid of its volume. In practical, the majority of components are not homogeneous, so that the CG must be calculated by adding the offset moments along all of the three axes.

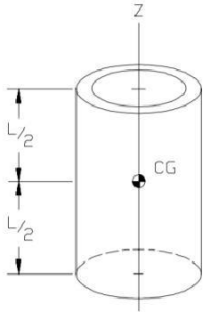


Fig.4.12. Center of Gravity

UNIT V CAD STANDARDS

Standards for computer graphics- Graphical Kernel System (GKS) - standards for exchangeimages- Open Graphics Library (OpenGL) - Data exchange standards - IGES, STEP, CALSetc. - communication standards.

CAD Standards

5.1. Introduction

The purpose of CAD standard is that the CAD software should not be device-independent and should connect to any input device via a device driver and to any graphics display via a device drive.

The graphics system is divided into two parts: the kernel system, which is hardware independent and the device driver, which is hardware dependent. The kernel system, acts as a buffer independent and portability of the program. At interface 'X' , the application program calls the standard functions and sub routine provided by the kernel system through what is called language bindings. These functions and subroutine, call the device driver functions and subroutines at interface 'Y' to complete the task required by the application program (Fig.5.1.).

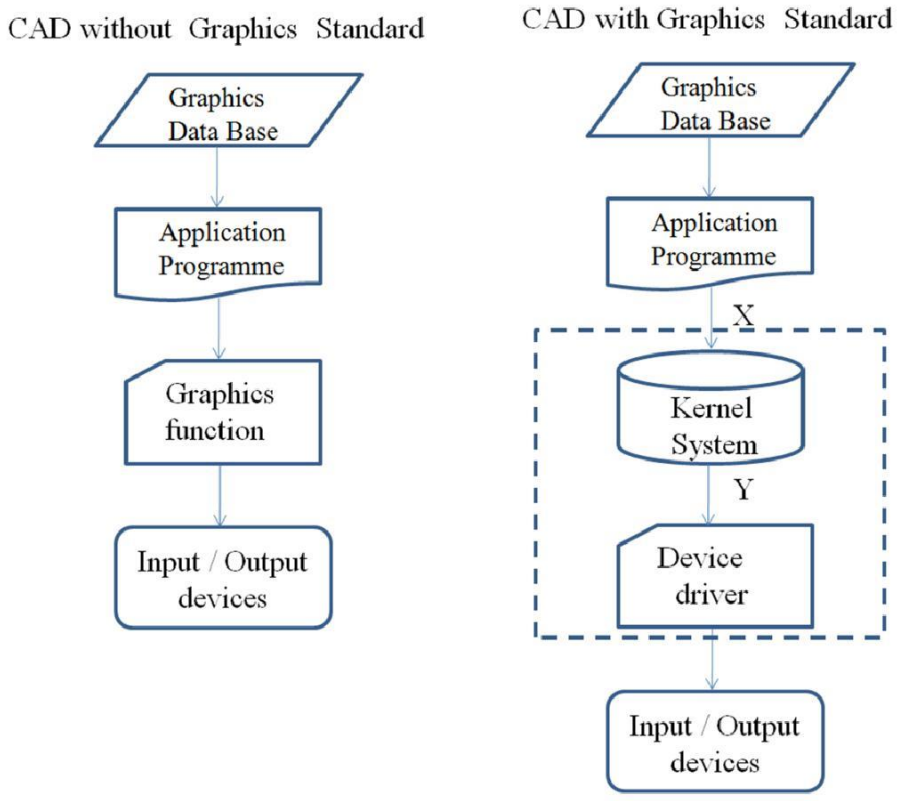


Fig.5.1. Graphics Standard

5.2. Various standards in graphics programming

The following international organizations involved to develop the graphics standards:

- ACM (Association for Computer Machinery)
- ANSI (American National Standards Institute)
- ISO (International Standards Organization)
- GIN (German Standards Institute)

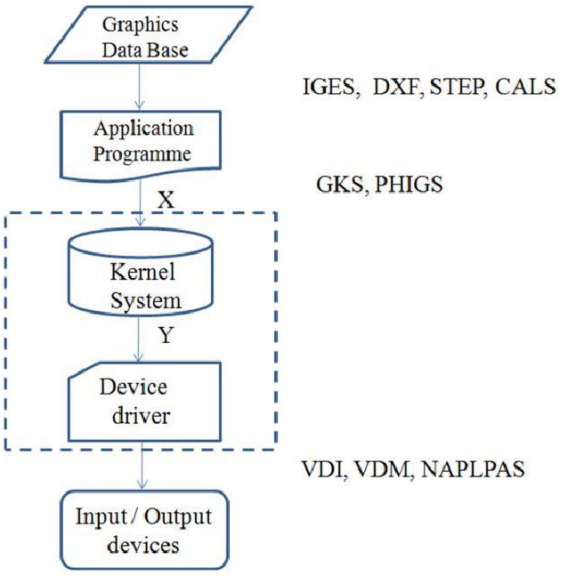


Fig.5.2. Graphics Standards in Graphics Programming

As a result of these international organization efforts, various standard functions at various levels of the graphics system developed. These are:

1. IGES (Initial Graphics Exchange Specification) enables an exchange of model data basis among CAD system.
2. DXF (Drawing / Data Exchange Format) file format was meant to provide an exact representation of the data in the standard CAD file format.
3. STEP (Standard for the Exchange of Product model data) can be used to exchange data between CAD, Computer Aided Manufacturing (CAM) , Computer Aided Engineering (CAE) , product data management/enterprise data modeling (PDES) and other CAx systems.

4. CALS (Computer Aided Acquisition and Logistic Support) is an US Department of Defense initiative with the aim of applying computer technology in Logistic support.
5. GKS (Graphics Kernel System) provides a set of drawing features for two-dimensional vector graphics suitable for charting and similar duties.
6. PHIGS (Programmer's Hierarchical Interactive Graphic System) The PHIGS standard defines a set of functions and data structures to be used by a programmer to manipulate and display 3-D graphical objects.
7. VDI (Virtual Device Interface) lies between GKS or PHIGS and the device driver code. VDI is now called CGI (Computer Graphics Interface).
8. VDM (Virtual Device Metafile) can be stored or transmitted from graphics device to another. VDM is now called CGM (Computer Graphics Metafile).
9. NAPLPS (North American Presentation- Level Protocol Syntax) describes text and graphics in the form of sequences of bytes in ASCII code.

5.3. Graphics Kernel System (GKS)

The Graphical Kernel System (GKS) was the first ISO standard for computer graphics in low-level, established in 1977. GKS offers a group of drawing aspects for 2D vector graphics appropriate for mapping and related duties. The calls are defined to be moveable across various programming languages, graphics hardware, so that applications noted to use GKS will be willingly portable to different devices and platforms.

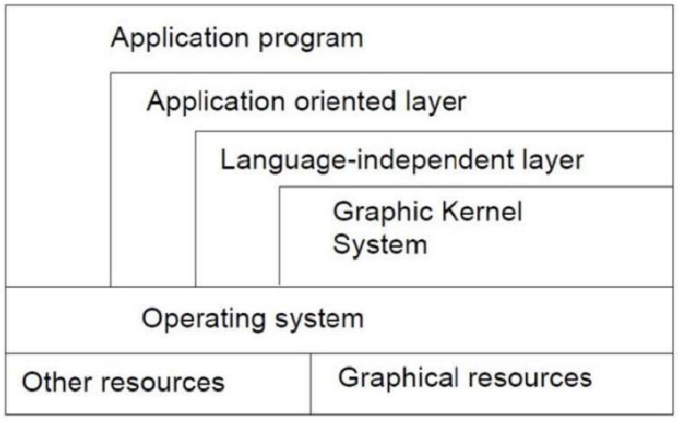


Fig.5.3. Layers of GKS

The following documents are representing GKS standards:

- The language bindings are called in ISO 8651 standard.
- ANSI X3.124 (1985) is part of ANSI standard.

- ISO/IEC 7942 noted in ISO standard, first part of 1985 and two to four parts of 1997-99.
- ISO 8805 and ISO 8806.

The main uses of the GKS standard are**:**

- To give for portability of application graphics programs.
- To assist in the learning of graphics systems by application programmers.
- To offer strategy for manufacturers in relating practical graphics capabilities.

The GKS consists of three basic parts:

i) A casual exhibition of the substances of the standard which contains such things as how text is placed, how polygonal zones are to be filled, and so onward.

ii) An official of the **descriptive** material in (i), by way of conceptual the ideas into separate functional explanations. These functional descriptions have such data as descriptions of input and output parameters, specific descriptions of the result of every function should have references into the **descriptive** material in (i), and a description of fault situation. The functional descriptions in this division are language autonomous.

iii) Language bindings are an execution of the abstract functions explained in (ii). in a explicit computer language such as C.

GKS arrange its functionality into twelve functional stages, based on the complexity of the graphical input and output. There are four stages of output (m, 0, 1, 2) and three stages of input (A, B, C). NCAR GKS has a complete execution of the GKS C bindings at level 0 A.

5.3.1. GKS Output Primitives

GKS is based on a number of elements that may be drawn in an object know as graphical primitives. The fundamental set of primitives has the word names POLYLINE, POLYMARKER, FILLAREA, TEXT and CELLARRAY, even though a few implementations widen this basic set.

i) POLYLINES

The GKS function for drawing line segments is called ‘POLYLINE’. The ‘POLYLINE’ command takes an array of X-Y coordinates and creates line segments joining them. The elements that organize the look of a ‘POLYLINE’ are (Fig.5.3):

- Line type : solid, dashed or dotted.
- Line width scale factor : thickness of the line.

- Polyline color index : color of the line.

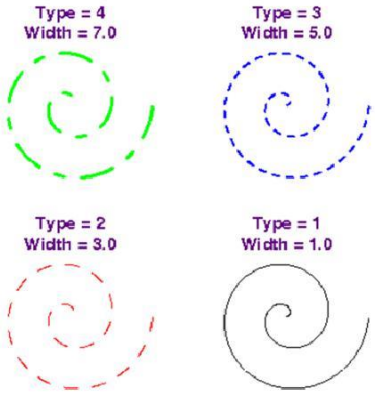


Fig.5.3. GKS POLYLINES

ii) POLYMARKERS

The GKS 'POLYMARKER' function permits to draw symbols of marker centered at coordinate points. The features that control the look of 'POLYMARKERS' are (Fig.5.4.):

- Marker characters : dot, plus, asterisk, circle or cross.
- Marker size scale factor : size of marker
- Polymarker color index : color of the marker.

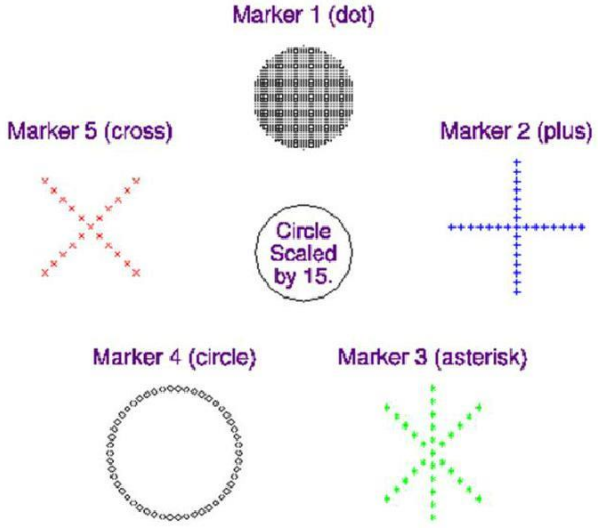


Fig.5.4. GKS POLYMARKERS

iii) FILLAREA

The GKS 'FILL AREA' function permits to denote a polygonal shape of a zone to be filled with various interior shapes. The features that control the look of fill areas are (Fig.5.5.):

- FILL AREA interior style : solid colors, hatch patterns.
- FILL AREA style index : horizontal lines; vertical lines; left slant lines; right slant lines; horizontal and vertical lines; or left slant and right slant lines.
- Fill area color index : color of the fill patterns / solid areas.

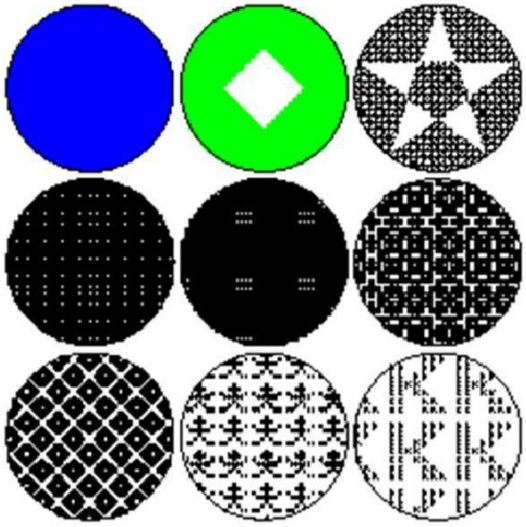


Fig.5.5. GKS FILLAREA

iv) TEXT

The GKS TEXT function permits to sketch a text string at a specified coordinate place. The features that control the look of text are:

- Text font and precision : text font should be used for the characters
- Character expansion factor : height-to-width ratio of each character.
- Character spacing : additional white space should be inserted between characters
- Text color index : color the text string
- Character height : size of the characters
- Character up vector : angle the text
- Text path : direction the text should be written (right, left, up, or down).
- Text alignment : vertical and horizontal centering options for the text string.

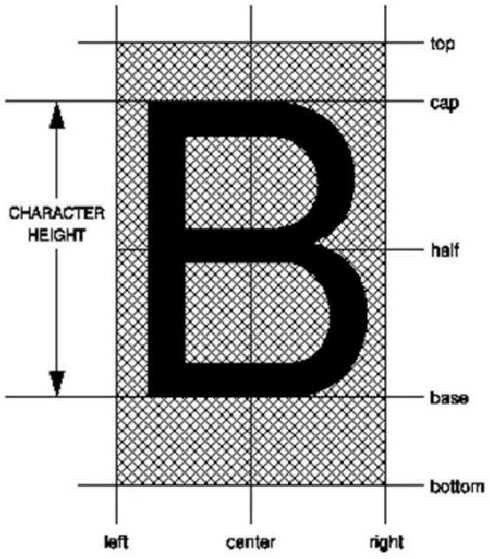


Fig.5.6. GKS TEXT

v) CELL ARRAY

The GKS CELL ARRAY function shows raster like pictures in a device autonomous manner. The CELL ARRAY function takes the two corner points of a rectangle that indicate, a number of partitions (M) in the X direction and a number of partitions (N) in the Y direction. It then partitions the rectangle into M x N sub rectangles noted as cells.

Fig.5.7. GKS CELL ARRAY

5.4. Standard for exchange images

A graphics standard proposed for interactive Three Dimensional applications should assure different criteria. It should be introduced on platforms with changing graphics abilities without sacrificing the graphics quality of the primary hardware and without compromising control over the hardware's function. It must offer a normal interface that permits a programmer to explain rendering processes quickly.

To end with, the interface should be flexible adequate to contain additions, hence that as new graphics operations become important, these operations can be given without sacrificing the original interface. OpenGL meets these measures by giving a simple interface to the basic operations of 3D graphics rendering. It supports basic graphics primitives, basic rendering operations and lighting calculations. It also helps advanced rendering attributes such as texture mapping.

5.4.1. Open Graphics Library

OpenGL draws primitives into a structured buffer focus to a various selectable modes. Every Point, line, polygon, or bitmap are called as a primitive. Each mode can be modified separately; the parameters of one do not affect the parameters of others. Modes defined, primitives detailed, and other OpenGL operations explained by giving commands in the form of procedure calls.

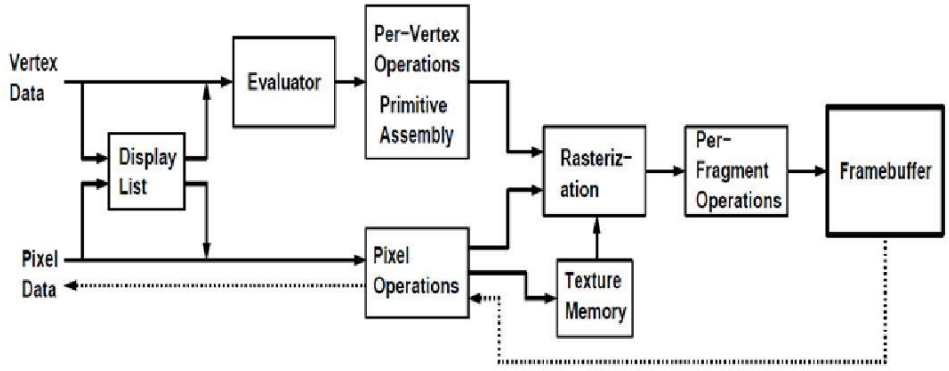


Fig.5.7. Schematic diagram of OpenGL

Figure 5.7 shows a schematic diagram of OpenGL. Commands go into OpenGL on the left. The majority commands may be collected in a 'display list' for executing at a later time. If not, commands are successfully sent through a pipeline for processing.

The first stage gives an effective means for resembling curve and surface geometry by estimating polynomial functions of input data. The next stage works on geometric primitives explained

by vertices. In this stage vertices are converted, and primitives are clipped to a seeing volume in creation for the next stage.

All ‘fragment’ created is supplied to the next stage that executes processes on personal fragments before they lastly change the structural buffer. These operations contain restricted updates into the structural buffer based on incoming and formerly saved depth values, combination of incoming colors with stored colors, as well as covering and other logical operations on fragment values.

To end with, rectangle pixels and bitmaps by pass the vertex processing part of the pipeline to move a group of fragments in a straight line to the individual fragment actions, finally rooting a block of pixels to be written to the frame buffer. Values can also be read back from the frame buffer or duplicated from one part of the frame buffer to another. These transfers may contain several type of encoding or decoding.

5.4.2. Features of OpenGL

i) Based on IRIS GL

OpenGL is supported on Silicon Graphics’ Integrated Rater Imaging System Graphics Library (IRIS GL). Though it would have been potential to have designed a totally new Application Programmer’s Interface (API), practice with IRIS GL offered insight into what programmers need and don’t need in a Three Dimensional graphics API. Additional, creation of OpenGL similar to Integrated Rater Imaging System Graphics Library where feasible builds OpenGL most likely to be admitted; there are various successful IRIS GL applications, and programmers of IRIS GL will have a simple time switching to OpenGL.

ii) Low-Level

A critical target of OpenGL is to offer device independence while still permitting total contact to hardware. Therefore the API gives permission to graphics operations at the lowest level that still gives device independence. Hence, OpenGL does not give a suggestion for modeling complex geometric objects.

iii) Fine-Grained Control

Due to minimize the needs on how an application utilizing the Application Programmer’s Interface must save and present its information, the API must give a suggestion to state entity parts of geometric entities and operations on them. This fine-grained control is necessary so that these

mechanism and operations may be defined in any order and so that control of rendering operations is comfortable to contain the needs of various applications.

iv) Modal

A modal Application Programmer's Interface arises in executions in which processes function in parallel on different primitives. In that cases, a mode modify must be transmit to all processors so that all collects the new parameters before it processes its next primitive. A mode change is thus developed serially, stopping primitive processing until all processors have collected the modifications, and decreasing performance accordingly.

v) Frame buffer

Most of OpenGL needs that the graphics hardware has a frame buffer. This is a realistic condition since almost all interactive graphics run on systems with frame buffers. Some actions in OpenGL are attained only during exposing their execution using a frame buffer. While OpenGL may be applied to give data for driving such devices as vector displays, such use is minor.

vi) Not Programmable

OpenGL does not give a programming language. Its function may be organized by turning actions on or off or specifying factors to operations, but the rendering algorithms are basically fixed. One basis for this decision is that, for performance basis, graphics hardware is generally designed to apply particular operations in a defined order; changing these operations with random algorithms is generally infeasible. Programmability would variance with maintenance of the API close to the hardware and thus with the objective of maximum performance.

vii) Geometry and Images

OpenGL gives support for managing both 3D and 2D geometry. An Application Programmer's Interface for utilize with geometry should also give guidance for reading, writing, and copying images, because geometry and images are regularly joint, as when a Three Dimensional view is laid over a background image. Various per-fragment processes that are applied to fragments beginning from geometric primitives apply uniformly well to fragments corresponding to pixels in an image, making it simple to mix images with geometry.