

COMPUTER AIDED DESIGN (CAD)

UNIT – I Fundamentals of Computer Graphics

Presented by
Dr.M.Soundarrajan,
AP/MECH,
Muthayammal Engineering College,
Rasipuram.

CAD

- CAD also known as Computer Aided Drafting/Design.
- There are 3 different types of CAD (2D, 2.5D and 3D).
- The software is used to create and design models of these types and test them.

Uses of CAD

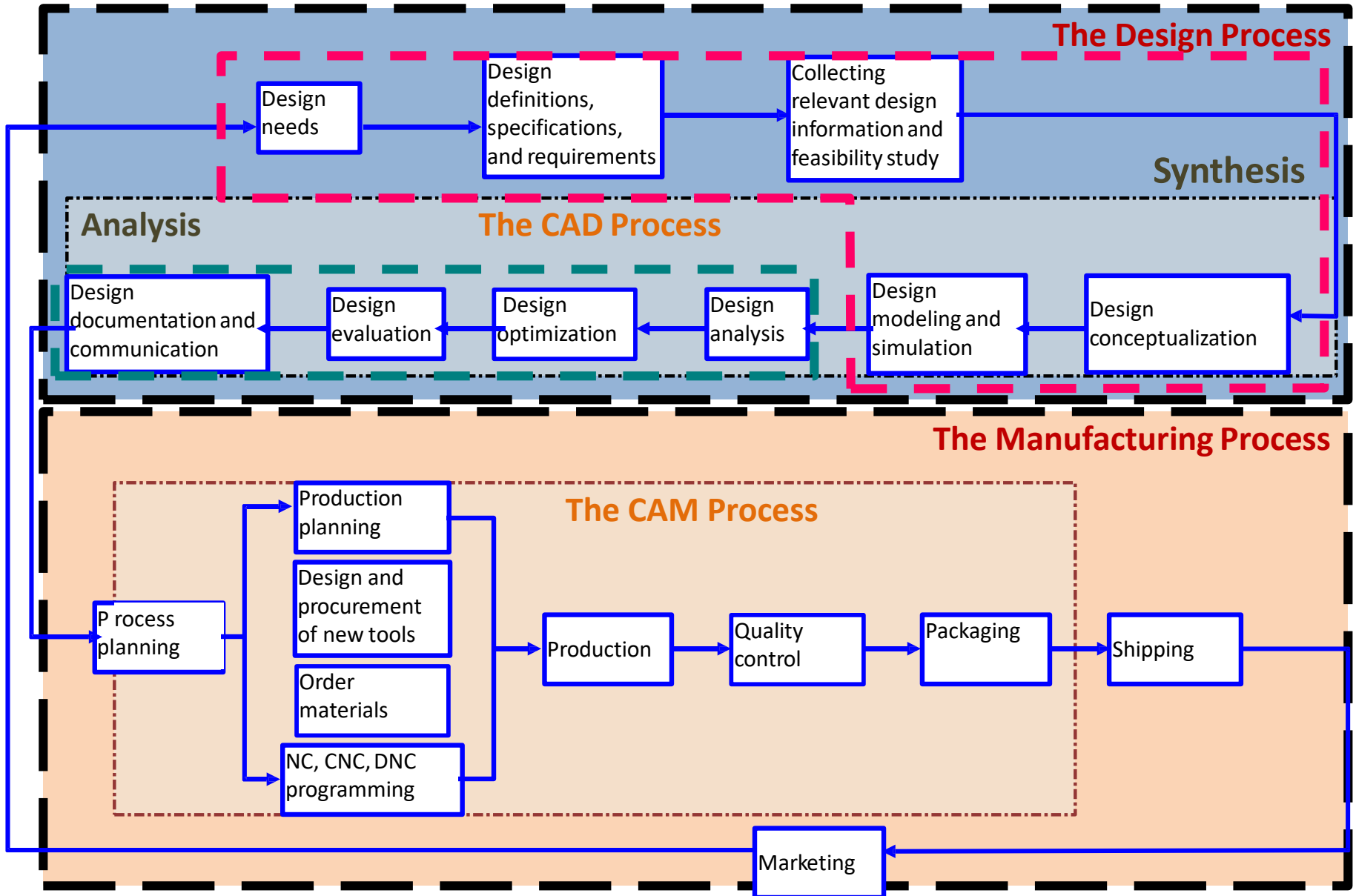
CAD is used to design a variety of different products for a variety of fields such as

- Architecture
- Electronics
- Automotive engineering
- Industrial Design
- Machinery
- Visual Art
- Medical Design

The Product Cycle and CAD/CAM

- The product begins with a need which is identified based on customers and market demands.
- In order to establish the scope and definition of CAD/CAM in an engineering environment and identify existing and future related tools, a study of a typical product cycle is necessary.

Typical Product Life Cycle



- Inspection to finished product
 - Two main process
 - Design Process
 - » Synthesis
(Sketches, Layout drawings- CAD/CAM system)
 - » Analysis
(Design Modeling & Simulation)
 - Manufacturing Process
(Process Planning & Production)
(Outcome → Production Plan, tools procurement, material order, CNC Programming)

Concurrent Engineering

- It is a strategy where all the tasks involved in product development are done in parallel.

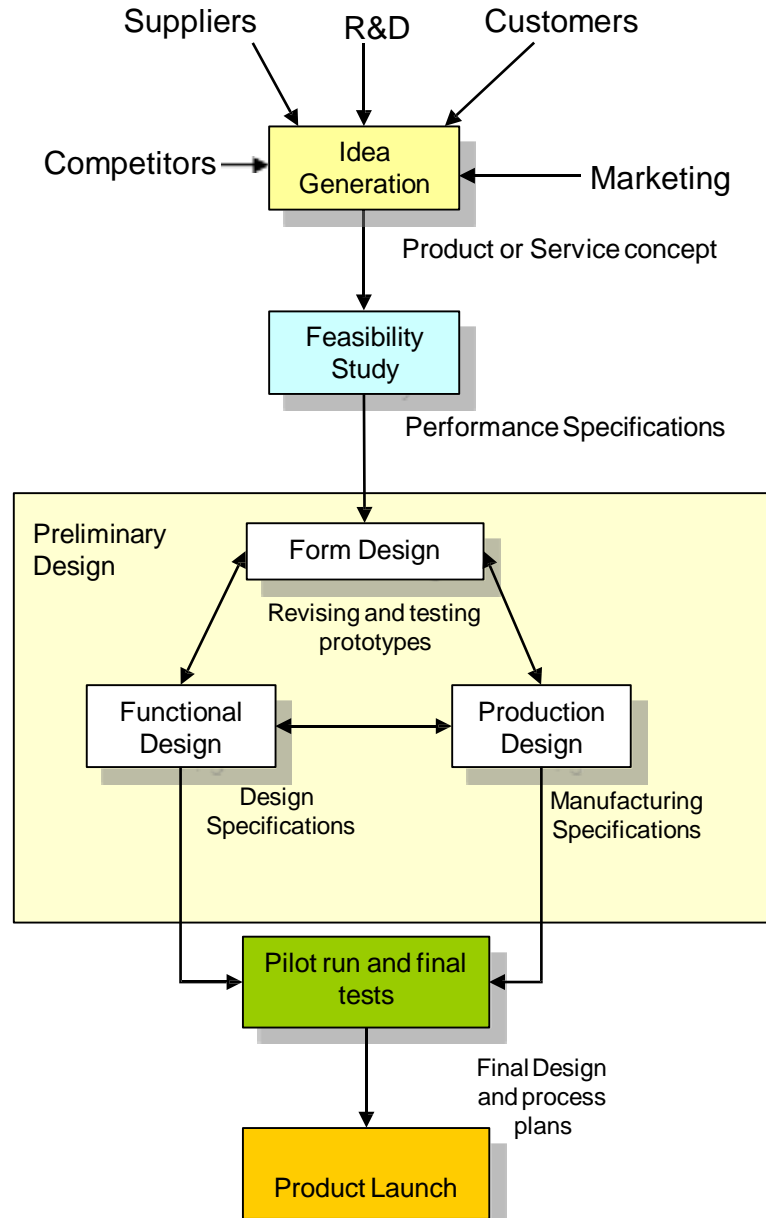
Collaboration between all individuals, groups and departments within a company.

- Customer research
- Designers
- Marketing
- Accounting
- Engineering

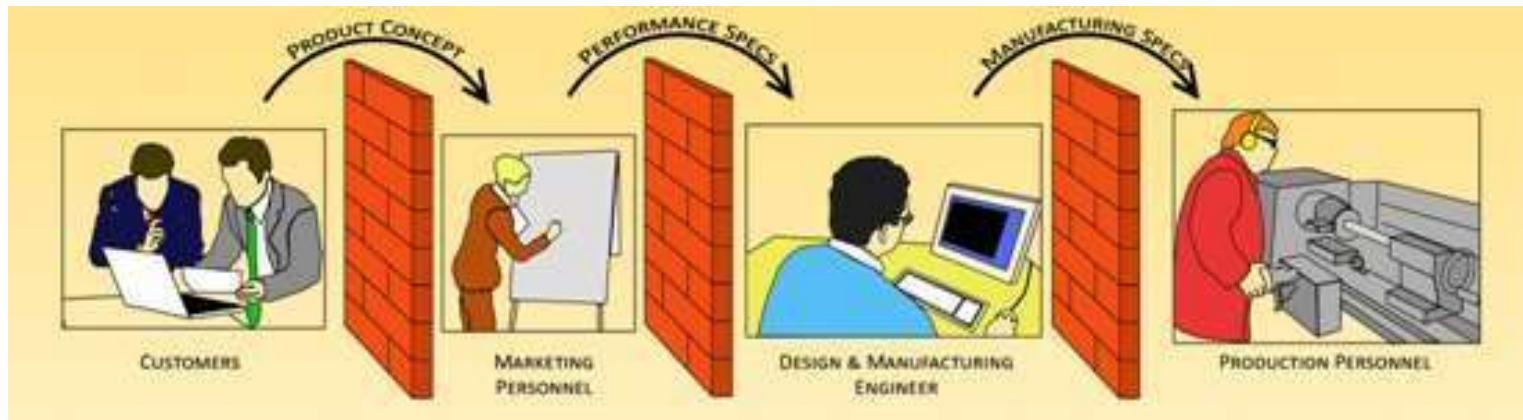
Concurrent Engineering

Commercial
Design Process

Linear Process



Sequential Vs Concurrent Engineering



Traditional Process = Linear

Vs

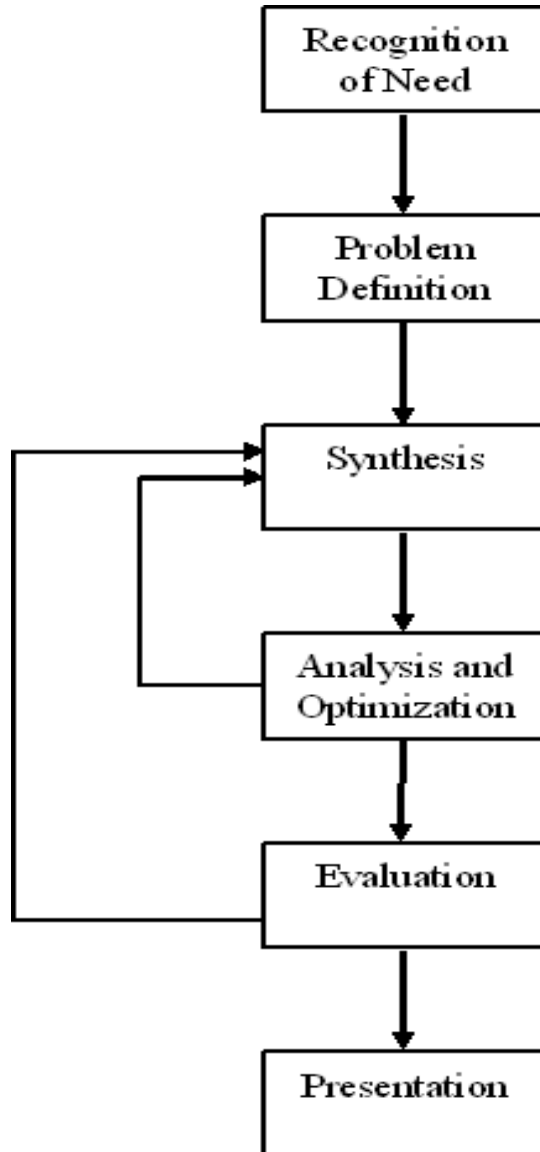
Concurrent Engineering = Team collaboration



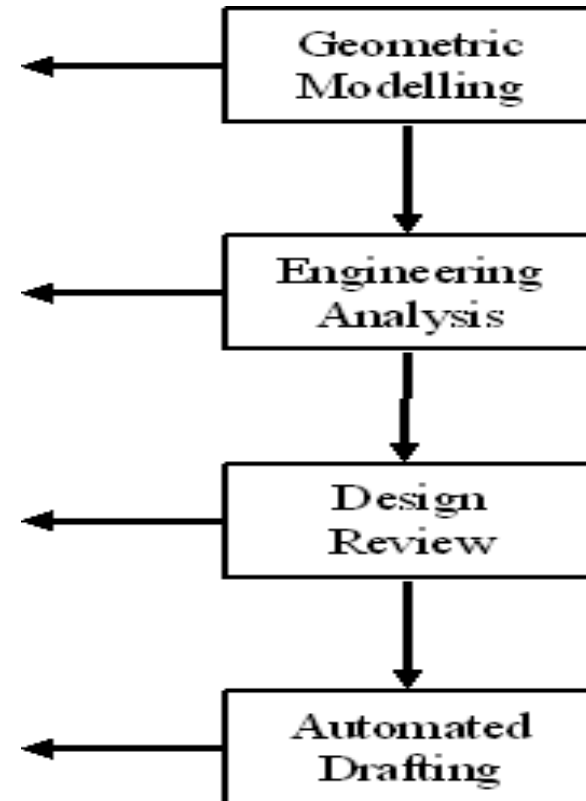
Benefits of Concurrent Engineering

- Reduces time from design concept to market launch by 25%.
- Reduces Capital investment by 20%.
- Supports total quality from the start of production with earlier opportunities for continuous improvement.
- Simplifies after-sales service.
- Increases product life-cycle profitability throughout the supply system.

The Design Process : Then and Now



Before CAD



After CAD

CAD/CAM Systems

1. Hardware

2. Software

✓ GUI

✓ Client/Standalone

✓ Database

✓ Works on all OS [Unix, Linux, Windows, Macintosh]

CAD/CAM Applications

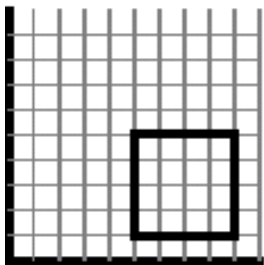
- Geometric Module → Modelling/editing, documentation
- Application Module → Utilize model for Design Analysis.
- Programming Module → Customization by programming
- Communication Module → IGES, STEP file
- Collaborative Module → collaborative design via internet

2D Transformations

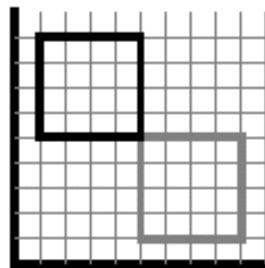
Basic 2D Transformations

- Translation
- Scaling
- Rotation

original

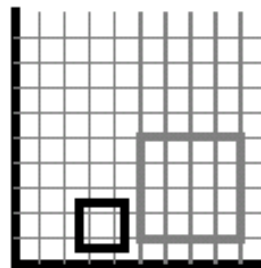


translation



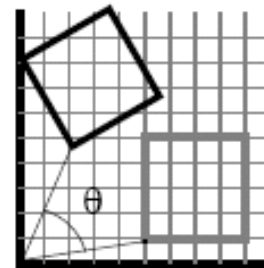
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

scaling



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

rotation



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

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

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

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & shx & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Basic 3D Transformations

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

Identity

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation

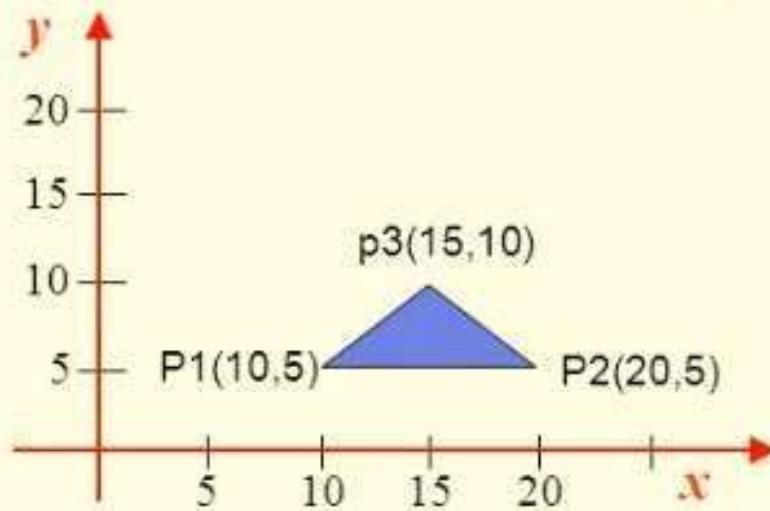
$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Mirror over X axis

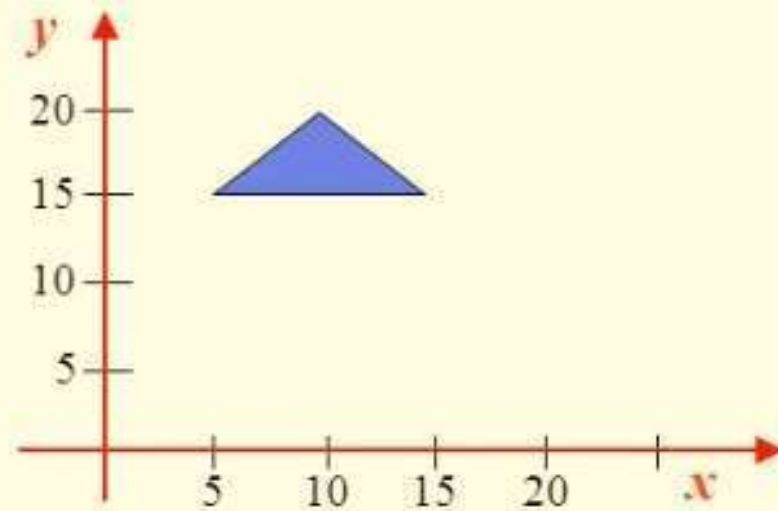
Example:

- Moving a polygon from position (a) to position (b) with the translation vector $(-5, 10)$, i.e.

$$T = \begin{pmatrix} -5 \\ 10 \end{pmatrix}$$



(a)



(b)

Example: Rotate a polygonal object defined by vertices A(0,0), B(1,0), C(1,1) and D(0,1) by 45 about the origin

$$v = [A \quad B \quad C \quad D] = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

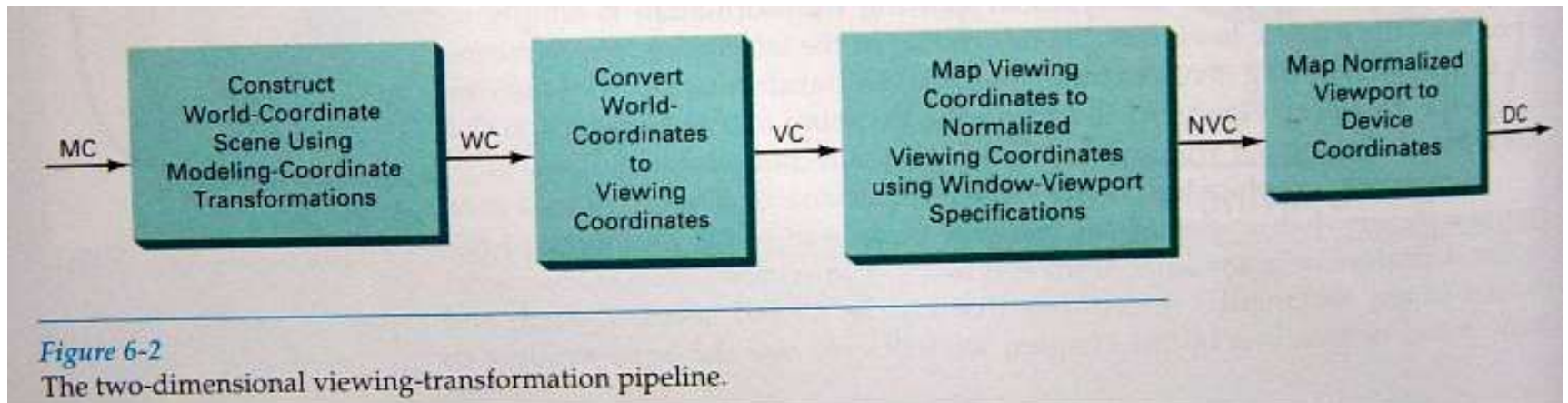
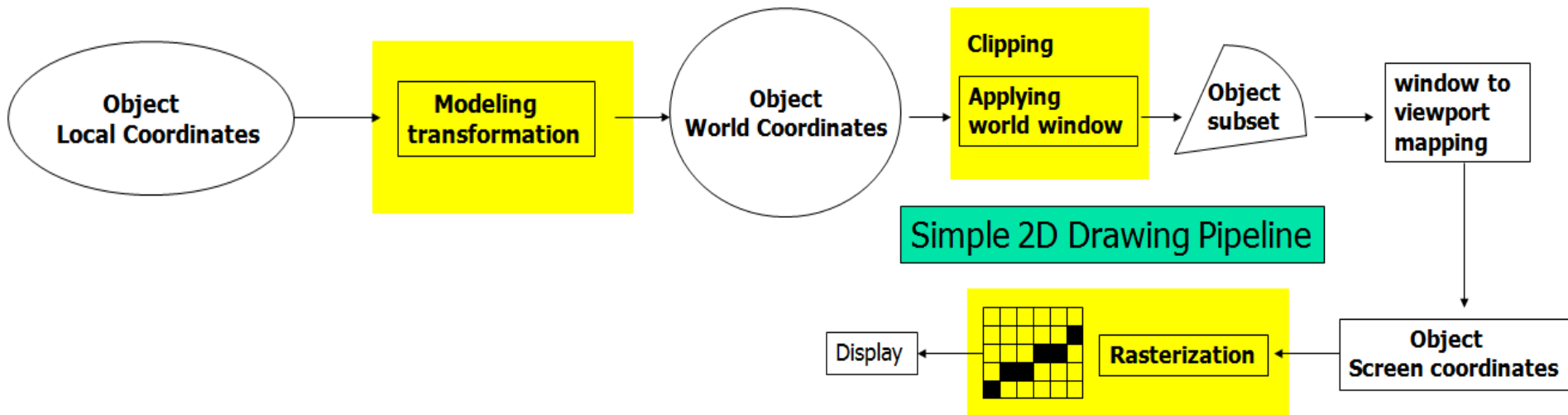
$$T_{Rotate} = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$v' = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{2}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Two Dimensional Viewing

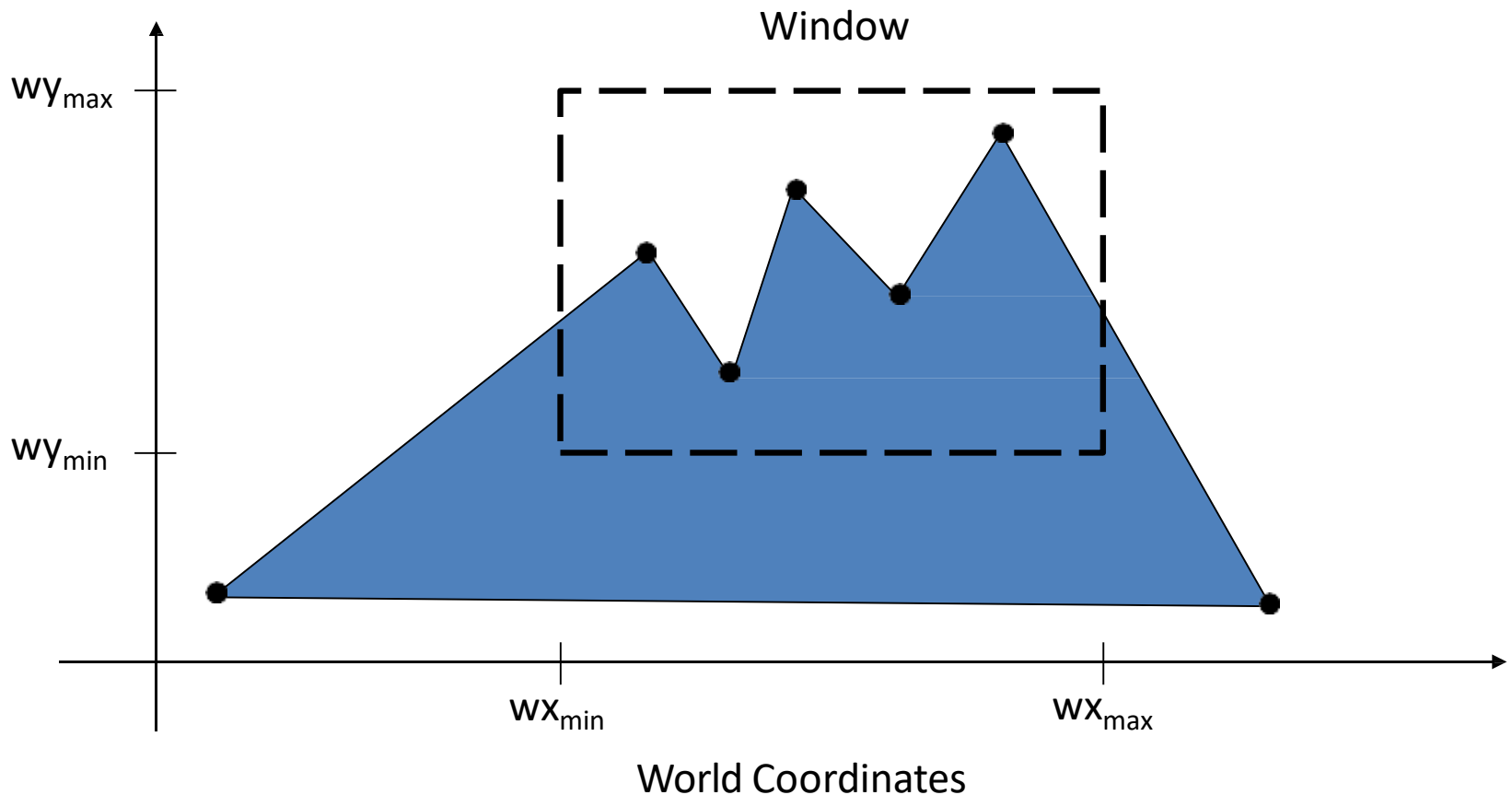
- The Viewing Transformation/ Pipeline
- Line Drawing
 - Several Algorithms
- Clipping
 - Point clipping
 - Line clipping
 - Area (Polygon) clipping
 - Curve clipping
 - Text clipping

Viewing Transformation



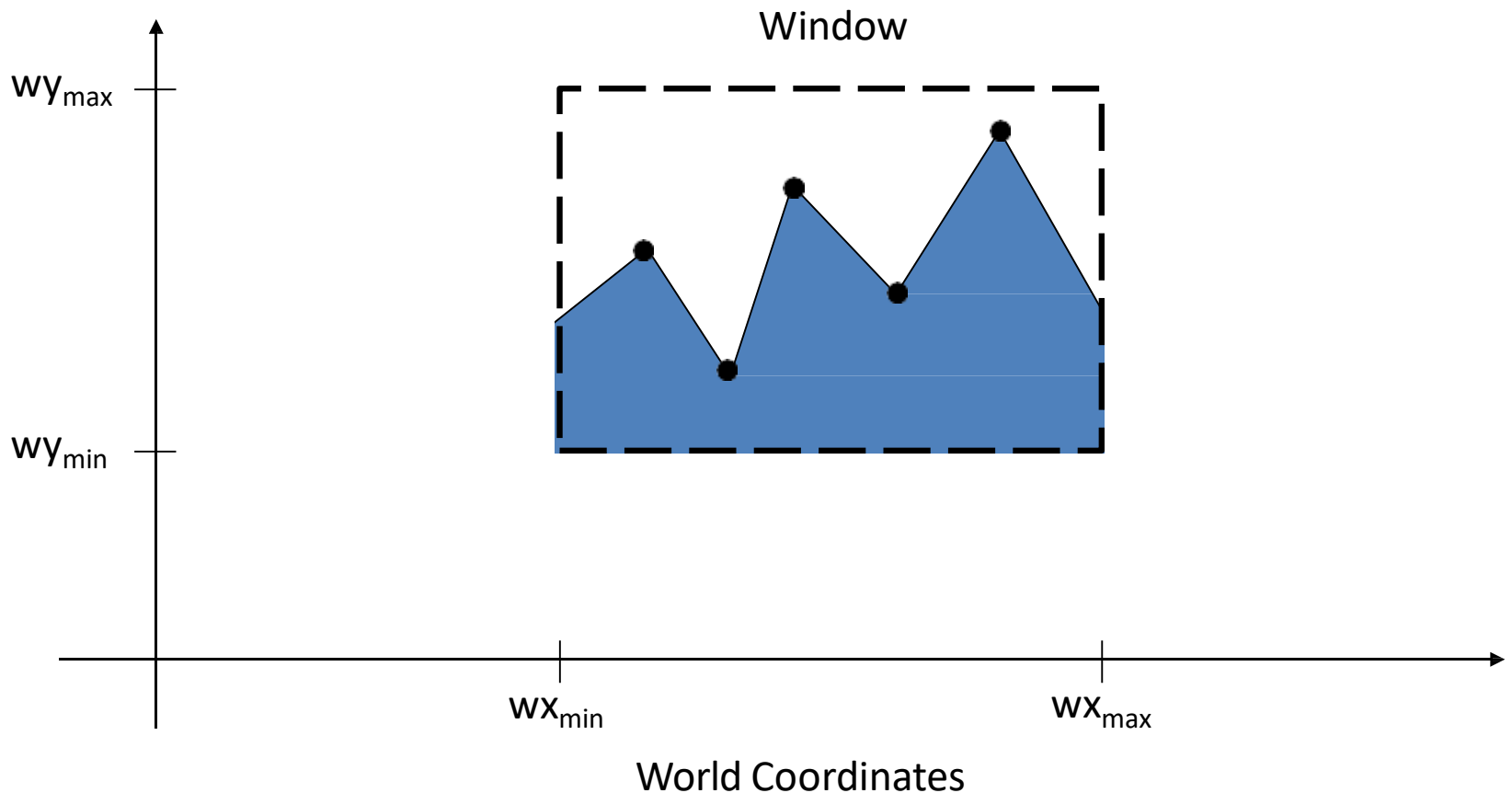
Windowing

When we display a scene only those objects within a particular window are displayed



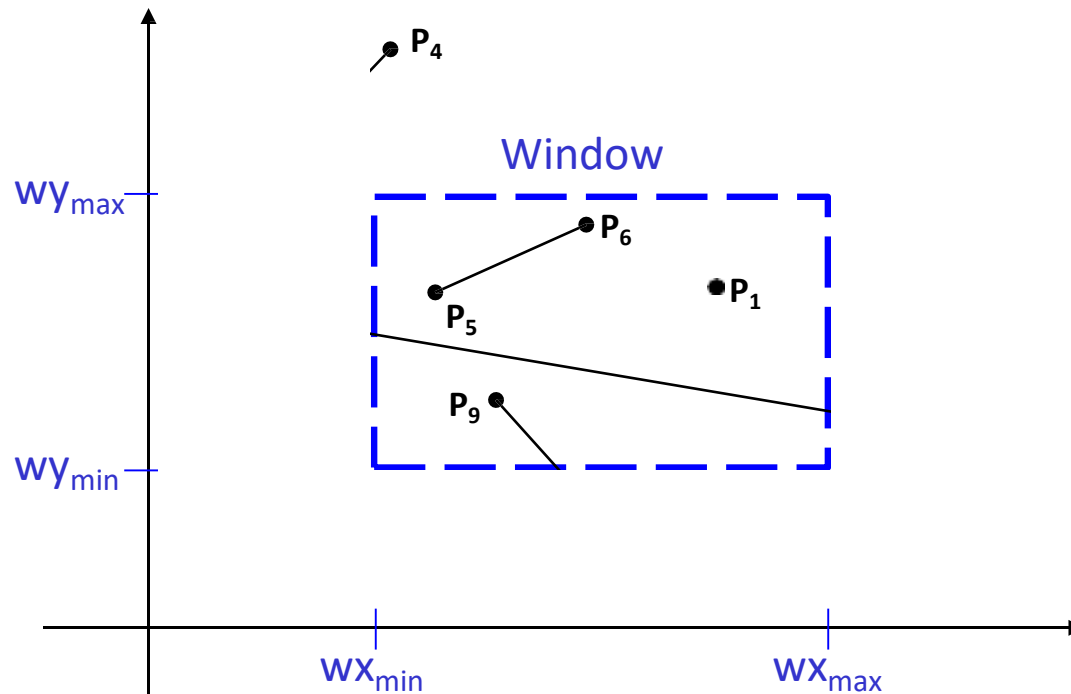
Windowing

Because drawing things to a display takes time we *clip* everything outside the window



Clipping

- Remove objects that are outside the world window.
- For the image below consider which lines and points should be kept and which ones should be clipped

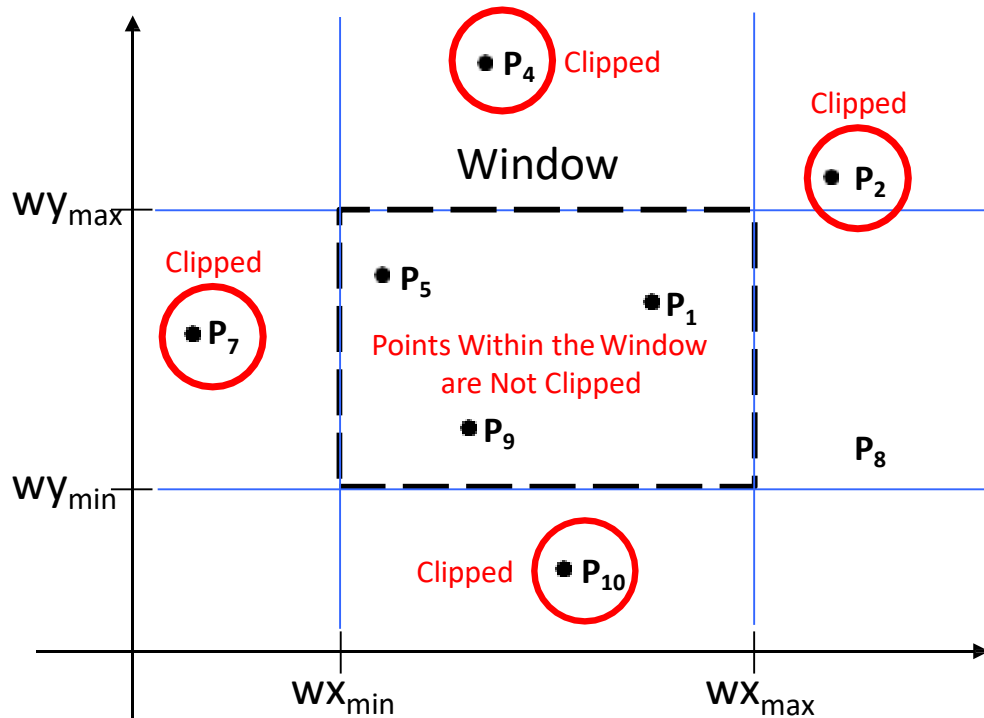


Point Clipping

Easy - a point (x,y) is not clipped if:

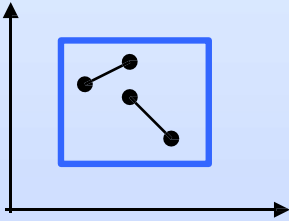
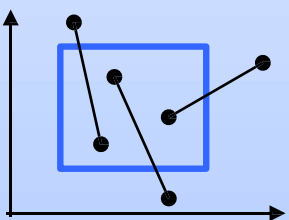
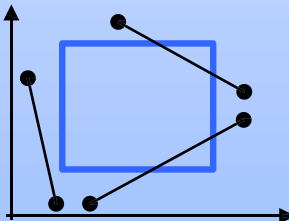
$$wx_{min} \leq x \leq wx_{max} \text{ AND } wy_{min} \leq y \leq wy_{max}$$

otherwise it is clipped



Line Clipping

Harder - examine the end-points of each line to see if they are in the window or not

Situation	Solution	Example
Both end-points inside the window	Don't clip	 A 2D coordinate system with x and y axes. A blue square window is drawn. A black line segment is drawn entirely within the boundaries of the square window. Both endpoints of the line are marked with black dots and are located inside the square.
One end-point inside the window, one outside	Must clip	 A 2D coordinate system with x and y axes. A blue square window is drawn. A black line segment is drawn such that one endpoint is inside the square window and the other endpoint is outside the window. The portion of the line inside the window is highlighted with a thicker line.
Both end-points outside the window	Don't know!	 A 2D coordinate system with x and y axes. A blue square window is drawn. A black line segment is drawn such that both endpoints are outside the square window. The portion of the line that passes through the window is highlighted with a thicker line.

Cohen-Sutherland Clipping Algorithm



Dr. Ivan E. Sutherland co-developed the Cohen-Sutherland clipping algorithm. Sutherland is a graphics giant and includes amongst his achievements the invention of the head mounted display.

Salient Features

- An efficient line clipping algorithm
- The key advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated

Cohen-Sutherland: World Division

World space is divided into regions based on the window boundaries

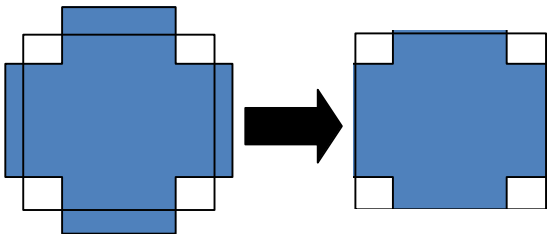
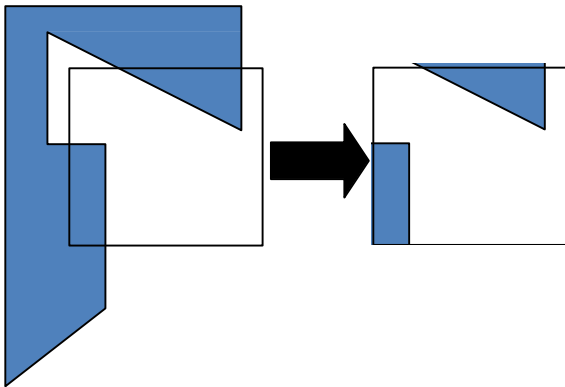
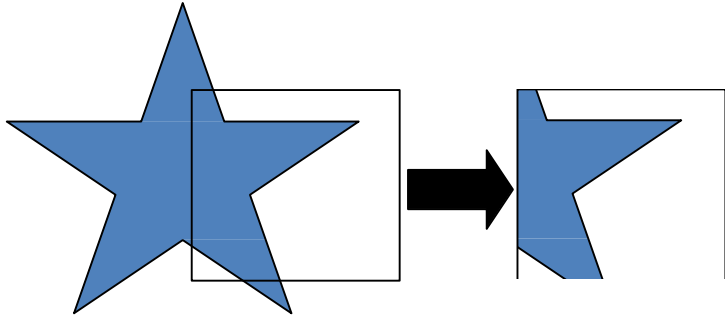
- Each region has a unique four bit region code
- Region codes indicate the position of the regions with respect to the window

4	3	2	1
above	below	right	left

Region Code Legend

1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

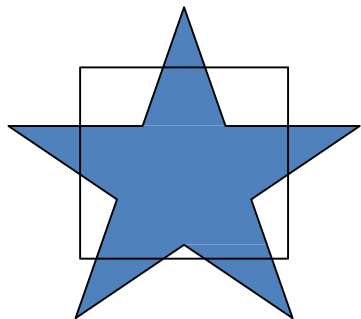
Area Clipping



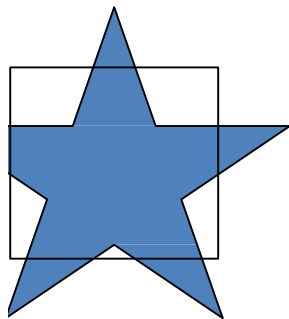
- Similarly to lines, areas must be clipped to a window boundary
- Consideration must be taken as to which portions of the area must be clipped

Sutherland-Hodgman: Area Clipping Algorithm

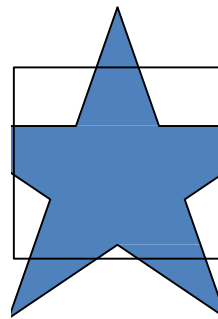
- A technique for clipping areas developed by Sutherland & Hodgman.
- Put simply the polygon is clipped by comparing it against each boundary in turn.



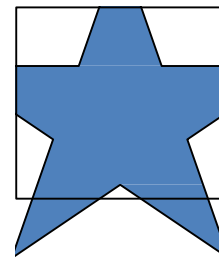
Original Area



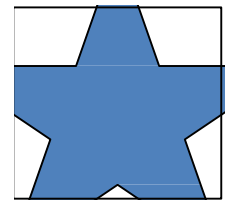
Clip Left



Clip Right



Clip Top



Clip Bottom

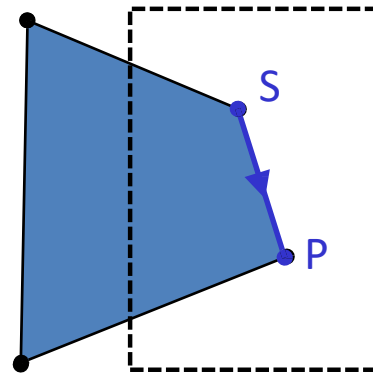
Sutherland-Hodgman: Area Clipping Algorithm (cont...)

To clip an area against an individual boundary:

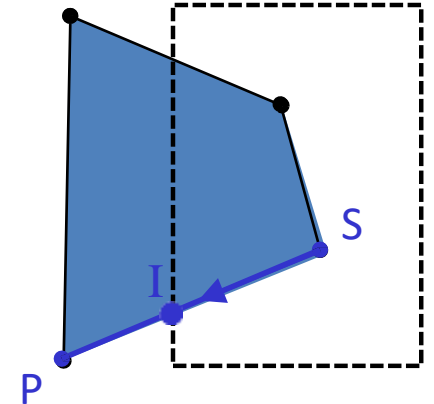
- Consider each vertex in turn against the boundary
- Vertices inside the boundary are saved for clipping against the next boundary
- Vertices outside the boundary are clipped
- If we proceed from a point inside the boundary to one outside, the intersection of the line with the boundary is saved
- If we cross from the outside to the inside intersection point and the vertex are saved

Sutherland-Hodgman Example

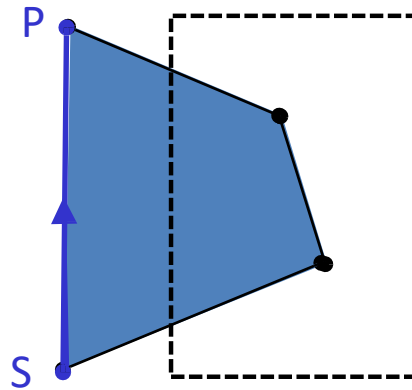
Each example shows the point being processed (P) and the previous point (S) and the saved points define area clipped to the boundary in question



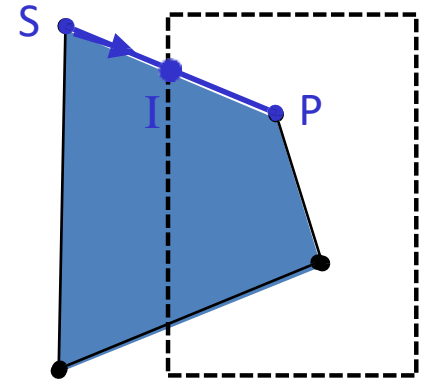
Save Point P



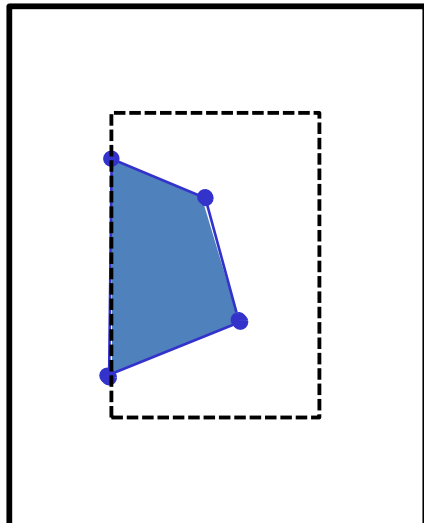
Save Point I



No Points Saved



Save Points I & P



Other Clipping

Curve clipping

- Use bounding rectangle to test for overlap with a rectangular clip window.

Text clipping

- All-or-none string-clipping
- All-or-none character-clipping
- Clip the components of individual characters

$$y = mx + b$$

$$m \rightarrow \text{slope}; m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}$$

case(i): if Slope is less than 1 & positive

Line starts from left
 $x_{k+1} = x_k + 1$

$$y_{k+1} = y_k + m$$

Line starts from right

$$x_{k+1} = x_k - 1$$

$$y_{k+1} = y_k - m$$

case(ii): if slope is greater than 1 & positive

Line starts from left

$$x_{k+1} = x_k + \frac{1}{m}$$

Line starts from right: $x_{k+1} = x_k - \frac{1}{m}$

$$y_{k+1} = y_k - 1$$

case(iii):

$m < 1$ & negative

$$x_{k+1} = x_k - 1$$

$$y_{k+1} = y_k + m$$

case(iv)

$m > 1$ & negative

$$x_{k+1} = x_k + \frac{1}{m}$$

$$y_{k+1} = y_k - 1$$

P:1

Draw a straight line connecting two end points (2,7) & (15,10).

Given:

$$x_1 = 2; x_2 = 15$$

$$y_1 = 7; y_2 = 10$$

$$x_2 - x_1 = 15 - 2 = 13$$

$$y_2 - y_1 = 10 - 7 = 3$$

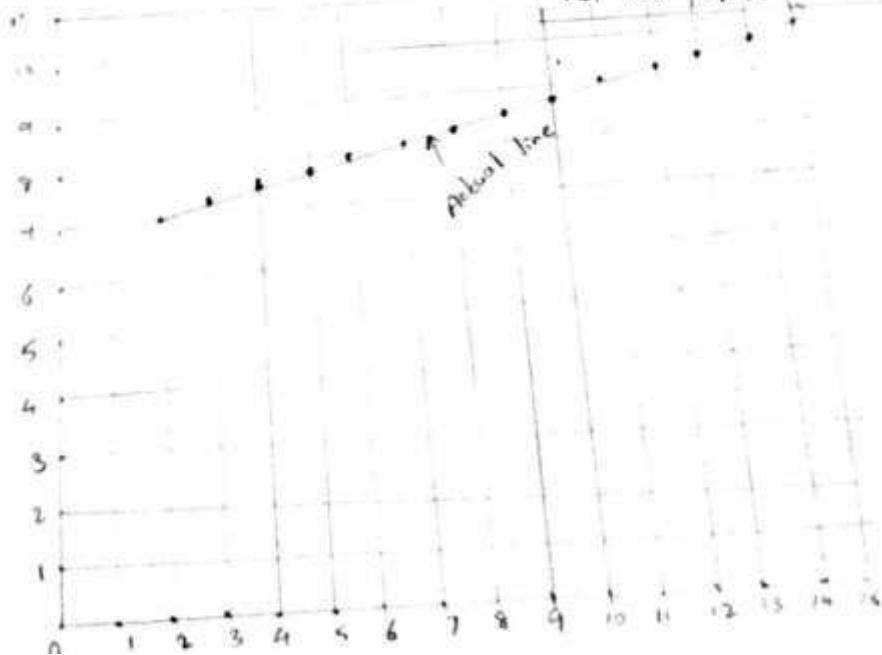
$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3}{13} = 0.23076$$

$m < 1$ & positive

$$x_{k+1} = x_k + 1$$

$$y_{k+1} = y_k + m$$

K	x_{k+1}	y_{k+1} Real value	Integer value
0	2	-	7
1	3	7.230	7
2	4	7.46	7
3	5	7.69	8
4	6	7.92	8
5	7	8.15	8
6	8	8.38	8
7	9	8.61	9
8	10	8.84	9
9	11	9.07	9
10	12	9.3	9
11	13	9.53	10
12	14	9.76	10
13	15	9.99	10



Bresenham's line drawing Algorithm:

(20)

$0 < m < 1$

If $0 < m < 1$ & slope between 0° & 45°

Step:1 \Rightarrow I/p the line end points AB as (x_1, y_1) and (x_2, y_2) respectively.

Step:2 \Rightarrow plot the first end point (x_1, y_1)

Step:3 \Rightarrow Calculate the initial value of decision parameters, P_k i.e. P_0

$$\Delta x = x_2 - x_1; \Delta y = y_2 - y_1$$

$$P_0 = 2\Delta y - \Delta x$$

Step:4 \Rightarrow At each x_k position, starting from $k=0$, perform the following test

• If $P < 1$ (negative) $\Rightarrow (x_k + 1, y_k)$

$$P_{k+1} = P_k + 2\Delta y$$

• If $P > 1$ (Positive) $\Rightarrow (x_{k+1}, y_{k+1})$

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$

Step:5 \Rightarrow plot the current (x, y)

Step:6 \Rightarrow Repeat step 4 through step 5, Δx times until the second end point is reached i.e. $x \geq x_2$

If $m > 1$ & slope is between 45° & 90° .

The procedure is same except that x & y coordinates are interchanged.

$$P_0 = 2\Delta x - \Delta y$$

• If $P_k < 1$ (negative) $\Rightarrow (x_k, y_k + 1)$

$$P_{k+1} = P_k + 2\Delta x$$

• If $P_k \geq 1$ (Positive) $\Rightarrow (x_{k+1}, y_{k+1})$

$$P_{k+1} = P_k + 2\Delta x - 2\Delta y$$

P.2

Draw the line using Bresenham's algorithm with the end points A(18,8) & B(28,16)

$$\Delta x = x_2 - x_1 = 10$$

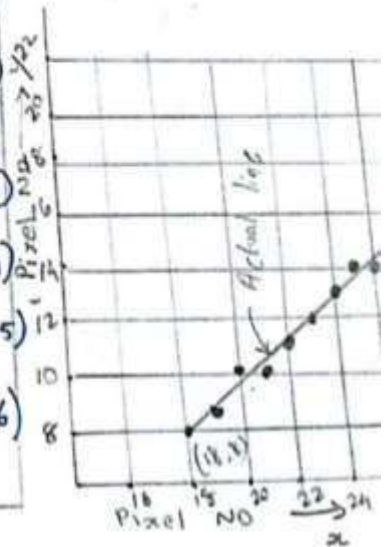
$$\Delta y = y_2 - y_1 = 8$$

$$m = \frac{\Delta y}{\Delta x} = 0.80$$

Since $m < 1$ & +ve

$$P_0 = 2\Delta y - \Delta x$$

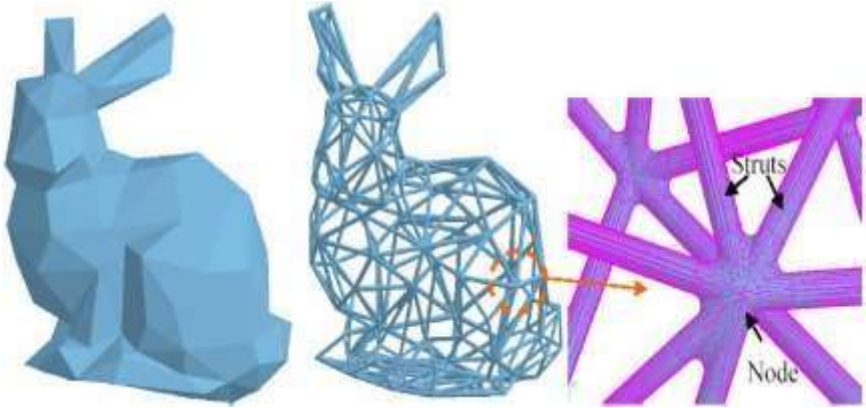
K	Decision Parameter P_k	Value of decision Parameter P_k	Next Pixel (x_k+1, y_k+1)	value of next pixel
0	$P_0 = 2\Delta y - \Delta x$	$2(8) - 10 = 6$	(x_k+1, y_k+1)	(19,9)
1	$P_1 = P_0 + 2\Delta y - 2\Delta x$	2	(x_k+1, y_k+1)	(20,10)
2	$P_2 = P_1 + 2\Delta y - 2\Delta x$	-2	(x_k+1, y_k)	(21,10)
3	$P_3 = P_2 + 2\Delta y$	14	(x_k+1, y_k+1)	(22,11)
4	$P_4 = P_3 + 2\Delta y - 2\Delta x$	10	(x_k+1, y_k+1)	(23,12)
5	$P_5 = P_4 + 2\Delta y - 2\Delta x$	6	(x_k+1, y_k+1)	(24,13)
6	$P_6 = P_5 + 2\Delta y - 2\Delta x$	2	(x_k+1, y_k+1)	(25,14)
7	$P_7 = P_6 + 2\Delta y - 2\Delta x$	-2	(x_k+1, y_k)	(26,14)
8	$P_8 = P_7 + 2\Delta y$	14	(x_k+1, y_k+1)	(27,15)
9	$P_9 = P_8 + 2\Delta y - 2\Delta x$	10	(x_k+1, y_k+1)	(28,16)



Thank You

COMPUTER AIDED DESIGN (CAD)

UNIT – II GEOMETRIC MODELING



UNIT – II GEOMETRIC MODELING

- **Representation of curves**
- **Hermite Curve- Bezier curve**
- **B-spline curves-rational curves**
- **Techniques for surface modelling**
- **Surface patch , Coons and bicubic patches**
- **Bezier and B-spline surfaces**
- **Solid modelling techniques**
- **CSG and B-rep**

Representation of curves

Types of Curve Equations

- Explicit (non-parametric)

$$Y = f(X), Z = g(X)$$

- Implicit (non-parametric)

$$f(X, Y, Z) = 0$$

- Parametric

$$X = X(t), Y = Y(t), Z = Z(t)$$

Analytic Curves vs. Synthetic Curves

- **Analytic Curves** are points, lines, arcs and circles, fillets and chamfers, and conics (ellipses, parabolas, and hyperbolas)
- **Synthetic curves** include various types of splines (cubic spline, B-spline, Beta-spline) and Bezier curves.

Example: A Circle of radius R

- Implicit:

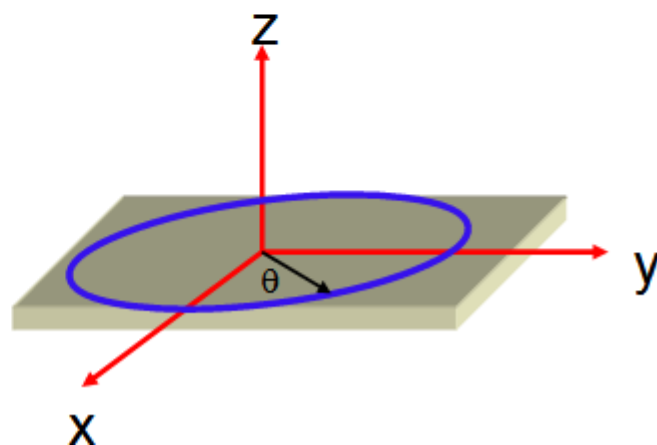
$$x^2 + y^2 + z^2 - R^2 = 0 \quad \& \quad z = 0$$

- Parametric:

$$x(\theta) = R \cos(\theta)$$

$$y(\theta) = R \sin(\theta)$$

$$z(\theta) = 0$$



Basic Concepts :

The order of continuity is a term usually used to measure the degree of continuous derivatives (C^0 , C^1 , C^2).

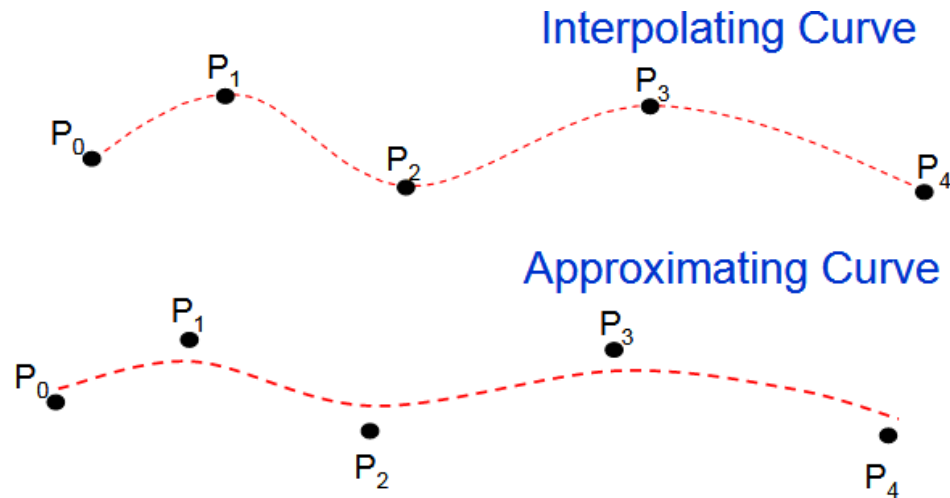


C^0 - Zero-order parametric continuity - the two curves sections must have the same coordinate position at the boundary point.

C^1 - First-order parametric continuity - tangent lines of the coordinate functions for two successive curve sections are equal at their joining point.

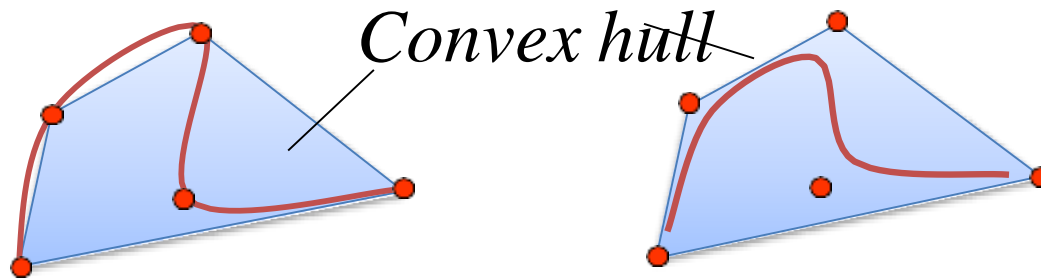
C^2 - second-order parametric continuity - both the first and second parametric derivatives of the two curve sections are the same at the intersection,

Interpolating and approximating curve:



Convex hull

The convex hull property ensures that a parametric curve will never pass outside of the convex hull formed by the four control vertices.



Interpolating spline Approximating spline

Hermite Curve

Hermite curves are designed by using two control points and tangent segments at each control point

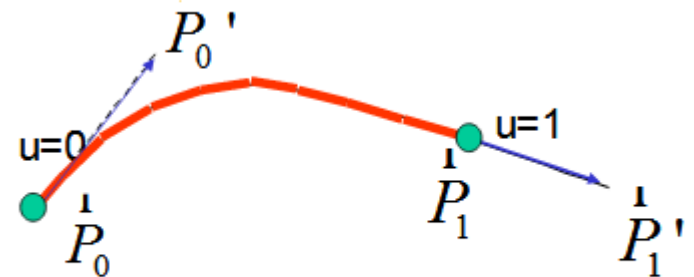
$$\vec{P}(u) = [x(u), y(u), z(u)]^T$$

$$\begin{cases} x(u) = c_{3x}u^3 + c_{2x}u^2 + c_{1x}u + c_{0x} \\ y(u) = c_{3y}u^3 + c_{2y}u^2 + c_{1y}u + c_{0y} \\ z(u) = c_{3z}u^3 + c_{2z}u^2 + c_{1z}u + c_{0z} \end{cases}$$

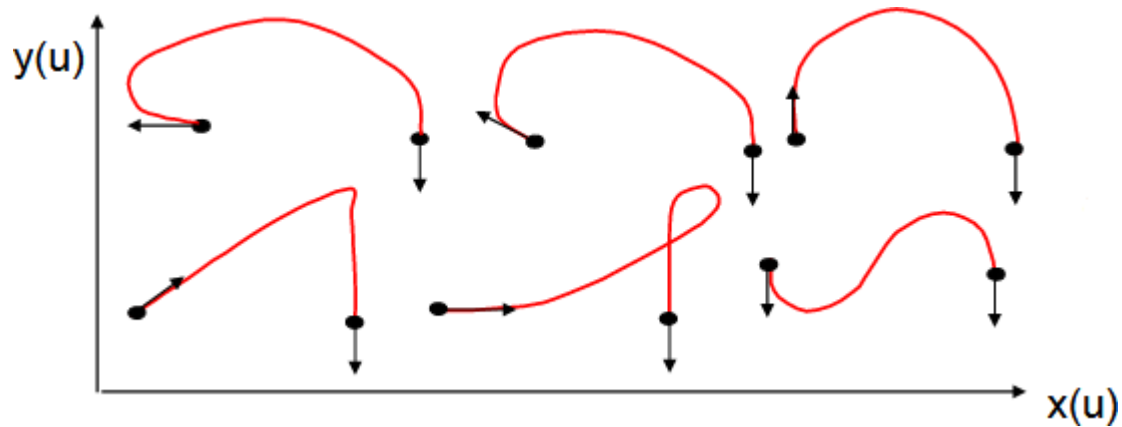
$3 \times 4 = 12$
coefficients to
be determined

$$\vec{p}(u) = [x(u) \ y(u) \ z(u)]^T = \sum_{i=0}^3 \vec{C}_i u^i \quad (0 \leq u \leq 1)$$

$$= [u^3 \ u^2 \ u \ 1] \begin{bmatrix} \vec{C}_3 \\ \vec{C}_2 \\ \vec{C}_1 \\ \vec{C}_0 \end{bmatrix} = [U^T][\vec{C}]$$



Hermite Curve contd...



$$\vec{P} = \sum_{i=0}^3 \vec{C}_i u^i = \vec{C}_3 u^3 + \vec{C}_2 u^2 + \vec{C}_1 u^1 + \vec{C}_0$$

$$\vec{P}' = 3\vec{C}_3 u^2 + 2\vec{C}_2 u + \vec{C}_1$$

Boundary Conditions:
Location of the two
end points and their
slopes

Two
End
Points

$$u = 0 \quad \left\{ \begin{array}{l} \vec{P}_0 = \vec{C}_0 \\ \vec{P}'_0 = \vec{C}_1 \end{array} \right.$$

$$u = 1 \quad \left\{ \begin{array}{l} \vec{P}_1 = \vec{C}_3 + \vec{C}_2 + \vec{C}_1 + \vec{C}_0 \\ \vec{P}'_1 = 3\vec{C}_3 + 2\vec{C}_2 + \vec{C}_1 \end{array} \right.$$

4×3 equations
from two
control points

Hermite Curve contd...

$$\vec{C}_0 = \vec{P}_0$$

$$\vec{C}_1 = \vec{P}'_0$$

$$\vec{C}_2 = 3(\vec{P}_1 - \vec{P}_0) - 2\vec{P}'_0 - \vec{P}'_1$$

$$\vec{C}_3 = 2(\vec{P}_0 - \vec{P}_1) + \vec{P}'_0 + \vec{P}'_1$$

12 unknowns
and
12 equations

$$\vec{P} = \sum_{i=0}^3 \vec{C}_i u^i = \vec{C}_3 u^3 + \vec{C}_2 u^2 + \vec{C}_1 u^1 + \vec{C}_0$$

All
parameters
can be
determined

$$\vec{P}(u) = (2u^3 - 3u^2 + 1)\vec{P}_0 + (-2u^3 + 3u^2)\vec{P}'_0 + (u^3 - 2u^2 + u)\vec{P}'_1 + (u^3 - u^2)\vec{P}_1$$

Hermite
Cubic curve
in vector
form

Hermite Curve contd...

$$\square \quad \vec{P}(u) = (2u^3 - 3u^2 + 1)\vec{P}_0 + (-2u^3 + 3u^2)\vec{P}_1 + (u^3 - 2u^2 + u)\vec{P}'_0 + (u^3 - u^2)\vec{P}'_1$$

Hermite Cubic curve in vector form

In matrix form:

$$\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{P}_0 \\ \vec{P}_1 \\ \vec{P}'_0 \\ \vec{P}'_1 \end{bmatrix}$$

Based on:
Location of the two end points and their slopes

$$= U^T [M_H] \vec{V} \quad 0 \leq u \leq 1$$

$$\square \quad \vec{P}'(u) = (6u^2 - 6u)\vec{P}_0 + (-6u^2 + 6u)\vec{P}_1 + (3u^2 - 4u + 1)\vec{P}'_0 + (3u^2 - 2u)\vec{P}'_1 \quad 0 \leq u \leq 1$$

where $[M_H]$ is the Hermite matrix and \vec{V} is the geometry (or boundary conditions) vector.

Properties:

- The Hermite curve is composed of a linear combinations of tangents and locations (for each u)
- Alternatively, the curve is a linear combination of Hermite basis functions (the matrix M)
- The piecewise interpolation scheme is C^1 continuous
- The blending functions have local support; changing a control point or a tangent vector, changes its local neighbourhood while leaving the rest unchanged

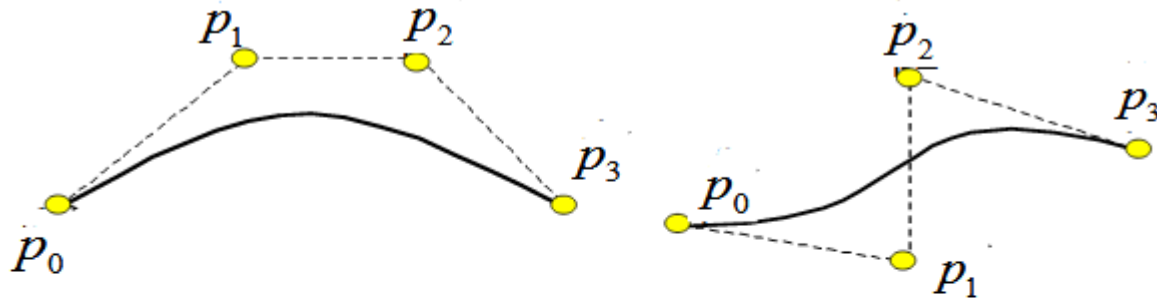
Disadvantages:

Requires the specification of the tangents. This information is not always available.

Limited to 3rd degree polynomial therefore the curve is quite stiff .

Bezier Curve

- A Bezier Curve is obtained by a defining polygon.
- First and last points on the curve are coincident with the first and last points of the polygon.
- Degree of polynomial is one less than the number of points
- Tangent vectors at the ends of the curve have the same directions as the respective spans
- The curve is contained within the convex hull of the defining polygon.



$$\vec{p}(u) = \sum_{i=0}^n \vec{p}_i B_{i,n}(u)$$

n — segment(each polygon)

$n+1$ — vertices (each polygon) and number of control points

$u \in [0, 1]$

Bernstein Polynomial

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

$B_{i,n}(u)$ is a function of the number of curve segments, n .

$n=2$

i	0	1	2
$\frac{n!}{i!(n-i)!}$	$\frac{2!}{0! 2!} = 1$	$\frac{2!}{1! 1!} = 2$	$\frac{2!}{2! 0!} = 1$

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 curve 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 simulate a circle, with a maximum radial error of less than one part in a thousand (Fig.1).

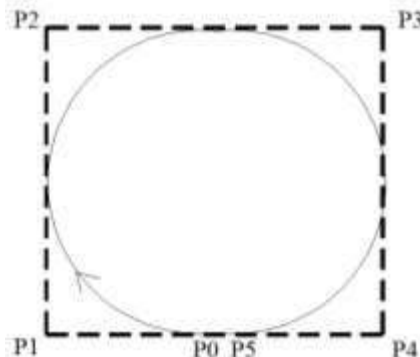


Fig1. Circular Bezier curve

- 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).

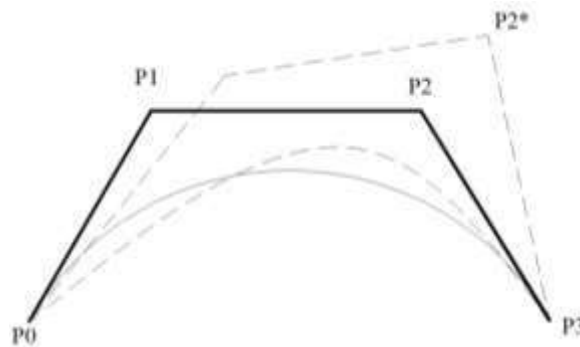
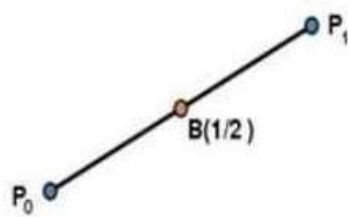
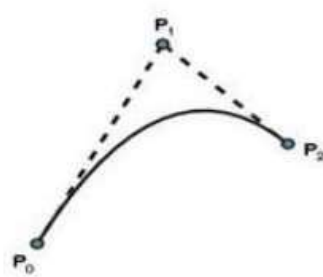


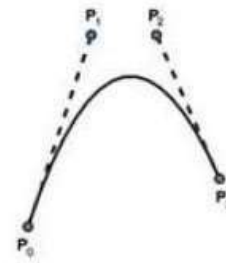
Fig: 2. Bezier curve shpe



Simple Bezier Curve

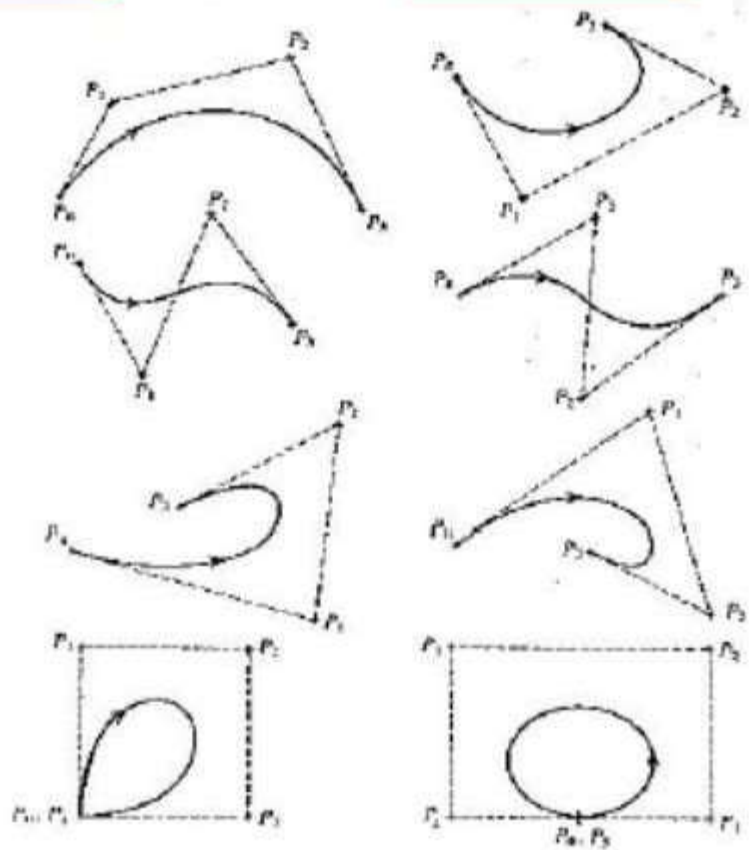


Quadratic Bezier Curve



Cubic Bezier Curve

The order of Bezier curve is a function of the number of control points. Four control points (n=3) always produce a cubic Bezier curve.



B-spline Curve

- It provide local control of the curve shape.
- It also provide the ability to add control points without increasing the degree of the curve.
- B-spline curves have the ability to interpolate or approximate a set of given data points.

The B-spline curve defined by $n+1$ control points \mathbf{P}_i is given by

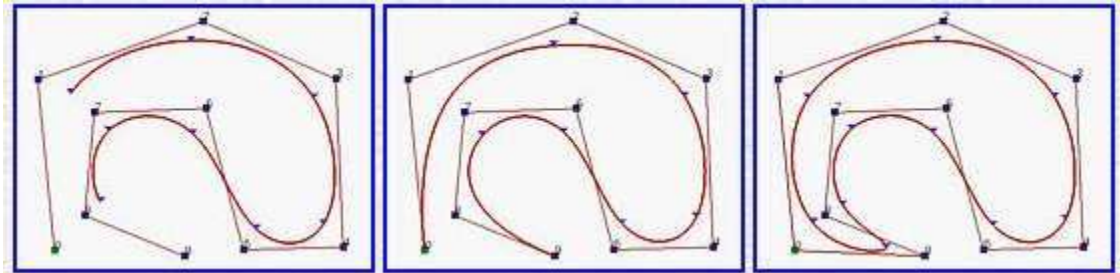
$$\mathbf{P}(u) = \sum_{i=0}^n \mathbf{P}_i N_{i,p}(u), \quad 0 \leq u \leq u_m$$

$N_{i,k}(u)$'s are B-spline basis functions of degree p .

$$m = n + p + 1.$$

- The form of a B-spline curve is very similar to that of a Bézier curve. Unlike a Bézier curve, a B-spline curve involves more information, namely: a set of $n+1$ control points, a knot vector of $m+1$ knots, and a degree p .
- Given $n + 1$ control points $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$ and a knot vector $U = \{u_0, u_1, \dots, u_m\}$, the B-spline curve of degree p defined by these control points and knot vector.
- The knot points divide a B-spline curve into curve segments, each of which is defined on a knot span.

- The degree of a Bézier basis function depends on the number of control points.
- To change the shape of a B-spline curve, one can modify one or more of these control parameters: the positions of control points, the positions of knots, and the degree of the curve.
- If the knot vector does not have any particular structure, the generated curve will not touch the first and last legs of the control polyline as shown in the left figure below.
- This type of B-spline curves is called *open* B-spline curves.



Properties of B-Spline Curve:

Partition of unity: $\sum_{i=0}^n N_{ik}(u) = 1$

Positivity: $N_{ik}(u) \geq 0$

Local support: $N_{ik}(u) = 0$ if $u \notin [u_i, u_{i+k+1}]$

Continuity: $N_{ik}(u)$ is $(k-2)$ times continuously differentiable

The first property ensures that the relationship between the curve and its defining control points is invariant under affine transformations.

The second property guarantees that the curve segment lies completely within the convex hull of \mathbf{P}_i .

The third property indicates that each segment of a B-spline curve is influenced by only k control points or each control point affects only k curve segments, as shown in Figure 1.

It is useful to notice that the **Bernstein polynomial**,
has the same first two properties mentioned above.

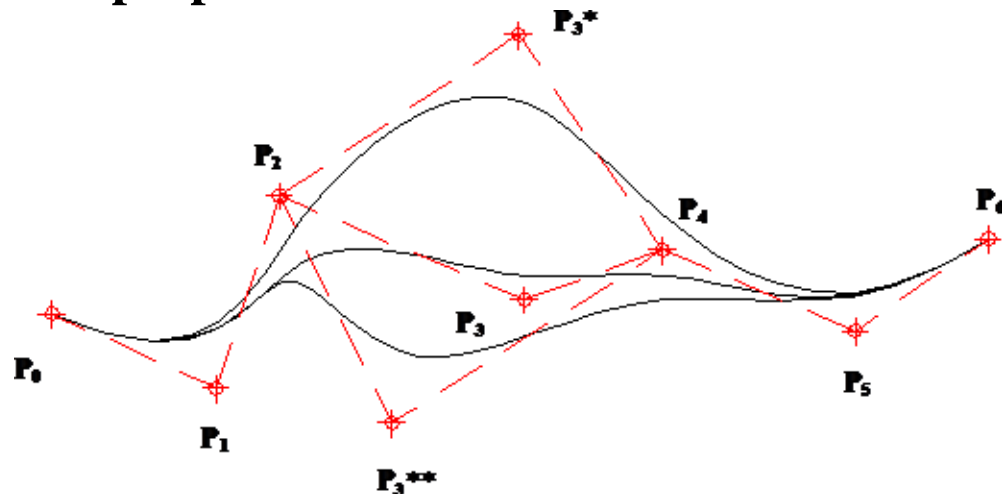


Figure 1. Local control of B-spline curves.

The B-spline function

The B-spline function also has the **property of recursion**, which is defined as

$$N_{ik}(u) = (u - u_i) \frac{N_{i,k-1}(u)}{u_{i+k-1} - u_i} + (u_{i+k} - u) \frac{N_{i+1,k-1}(u)}{u_{i+k} - u_{i+1}}$$

where

$$N_{i1} = \begin{cases} 1, & u_i \leq u \leq u_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

The u_i are called parametric knots or knot values. For an open curve,

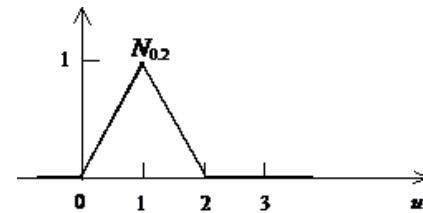
$$u_j = \begin{cases} 0, & j < k \\ j - k + 1, & k \leq j \leq n \\ n - k + 2, & j > n \end{cases}$$

where

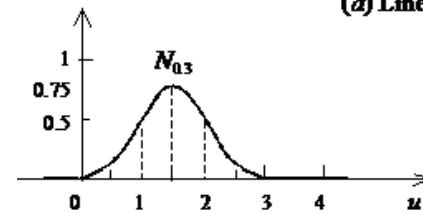
$$0 \leq j \leq n + k$$

and the range of u is

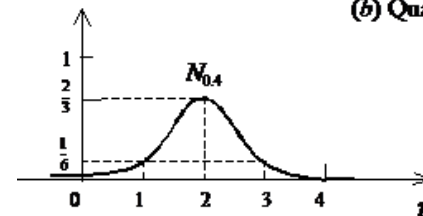
$$0 \leq u \leq n - k + 2$$



(a) Linear function ($k=2$)



(b) Quadratic function ($k=3$)



(c) Cubic function ($k=4$)

4.7.2 Techniques for Surface Modelling

1. The Surface Patch

The patch is the fundamental building block for surfaces. The two variables u and v vary across the patch - the patch may be termed *biparametric*. The parametric variables often lie in the range 0 to 1. Fixing the value of one of the parametric variables results in a curve on the patch in terms of the other variable (known as an isoparametric curve). Fig. 4.48 shows a surface with curves at intervals of u and v of 0 : 1.

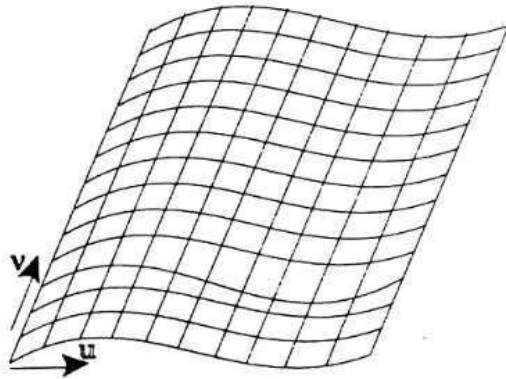


Fig. 4.48 Surface patch.

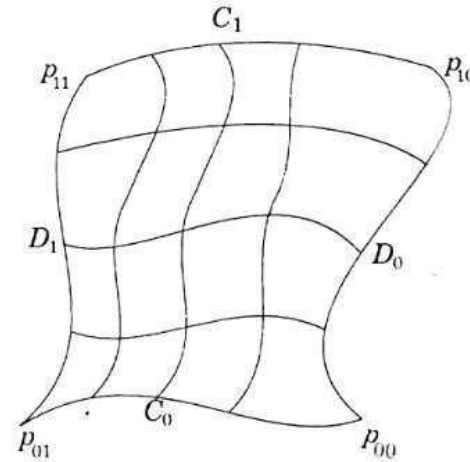


Fig. 4.49 Coons patch.

2. The Coons Patch

The sculptured surfaces often involve interpolation across an intersecting mesh of curves that in effect comprise a rectangular grid of patches, each bounded by four boundary curves. The linearly blended Coons patch is the simplest for interpolating between such boundary curves. This patch definition technique blends the four boundary curves $C_i(u)$ and $D_j(v)$ and the corner points p_{ij} of the patch (Fig. 4.49), with the linear blending functions,

$$f(t) = 1 - t \quad \dots(4.19)$$

$$g(t) = t$$

using the expression

$$\begin{aligned} \vec{p}(u, v) = & \vec{C}_0(u) f(v) + \vec{C}_1(u) g(v) + \vec{D}_0(v) f(u) + \vec{D}_1(v) g(u) \\ & - \vec{p}_{00} f(u) f(v) - \vec{p}_{01} f(u) g(u) - \vec{p}_{10} g(u) f(v) - \vec{p}_{11} g(u) g(v) \end{aligned} \quad \dots(4.20)$$

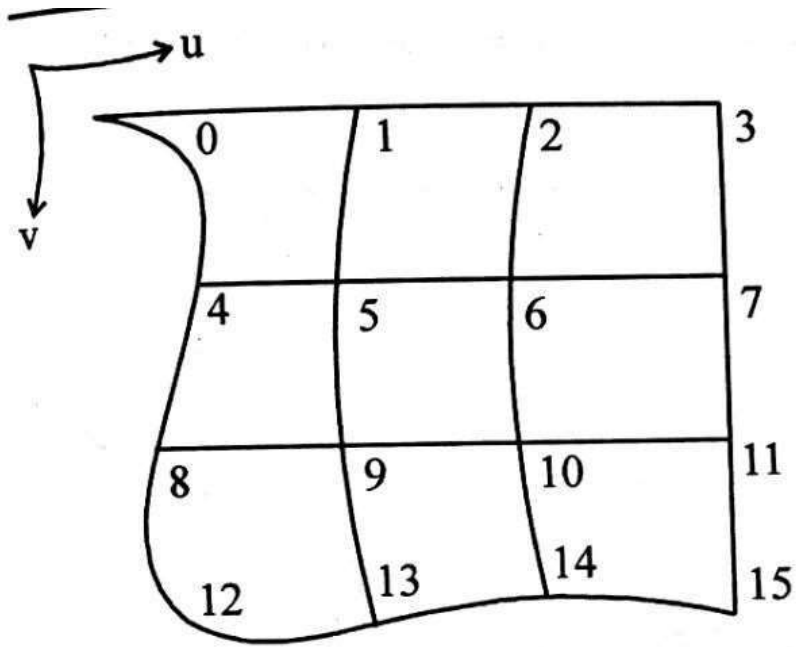
Linear blending has limitations and higher order blending functions such as cubics are used for formulations that allow tangency continuity between adjacent patches.

3. The Bicubic Patch

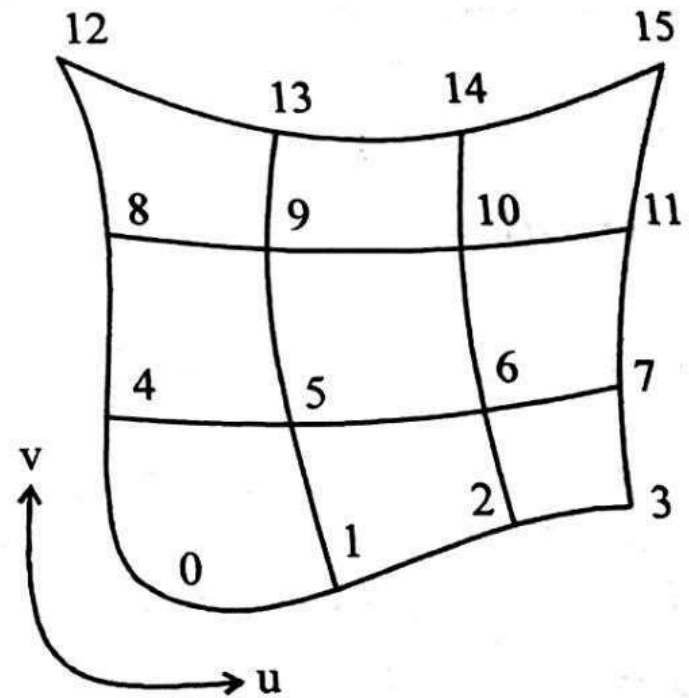
The bicubic patch is used for surface descriptions defined in terms of point and tangent vector information. The general form of the expressions for a bicubic patch are given by :

$$\vec{p}(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 k_{ij} u^i v^j \quad \dots(4.21)$$

This is a vector equation with 16 unknown parameters k_{ij} , which can be found by Lagrang



(a) Typical left handed Bezier bi-cubic patch



(b) Typical right handed Bezier bi-cubic patch

Figure 2.40 Bezier bi-cubic patch

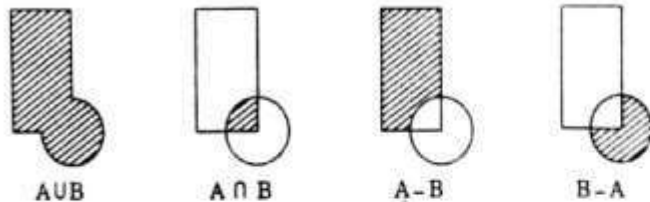
Solid modeling techniques

Constructive Solid Geometry (CSG)

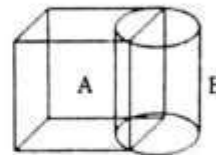
Constructive solid geometry (CSG) is a method used in solid modeling for creating 3D models in CAD. Constructive solid geometry permits a modeler to make a complex surface by applying Boolean operators to join objects. The simplest solid objects utilized for the demonstration are called primitives. Classically they are the items of simple shape like prisms, pyramids, spheres and cylinders.

CGS Boolean operation

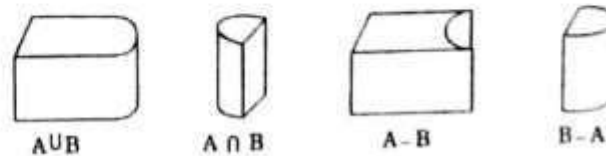
Boolean Operations



(a) Two dimensional



Primitives

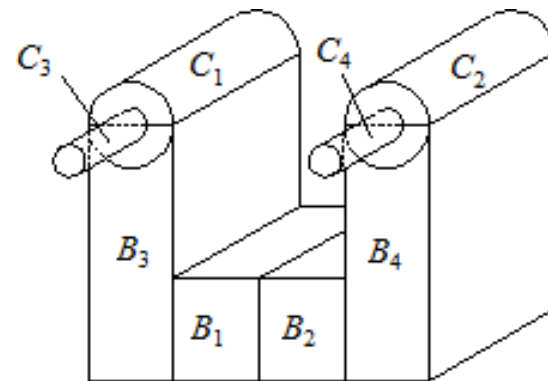
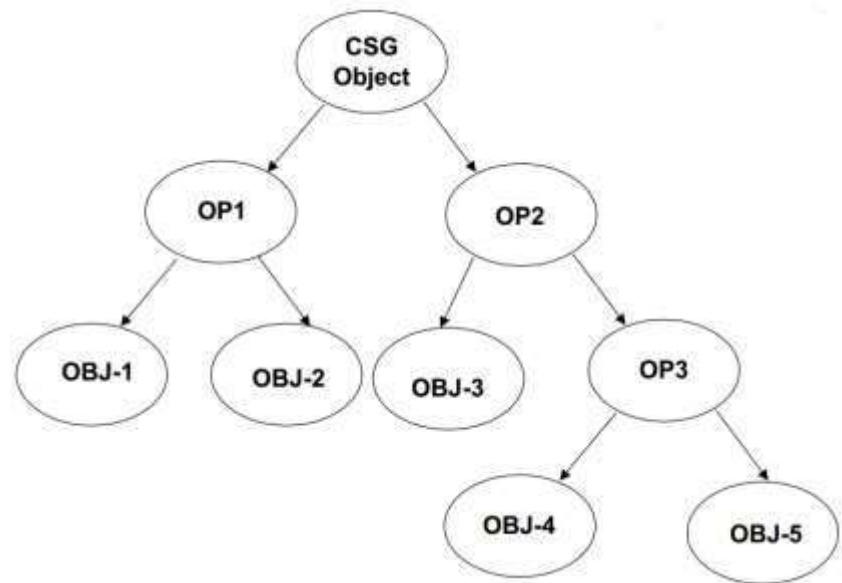


(b) Three dimensional

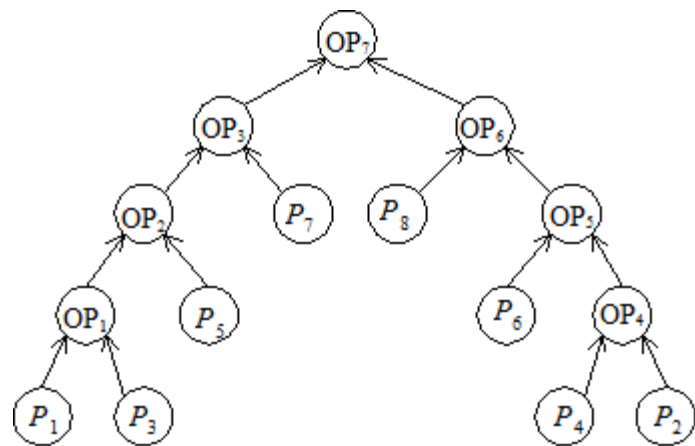
Boolean operations of a block A and cylinder B.

CSG tree

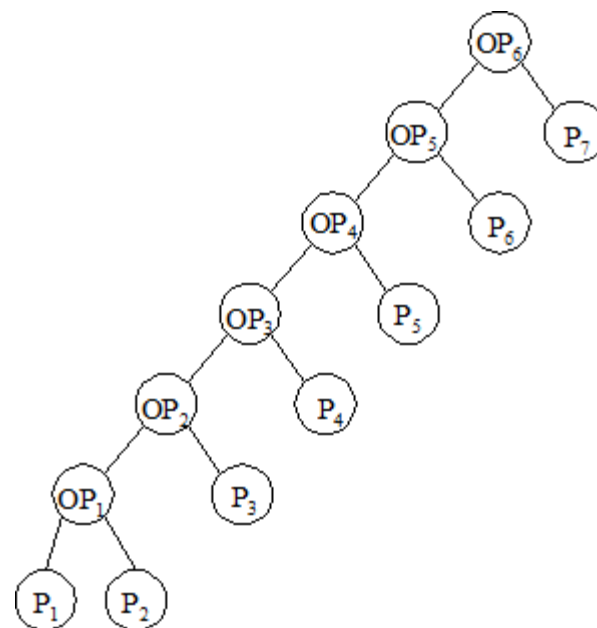
- A CSG tree is defined as an inverted ordered binary tree whose leaf nodes are primitives and interior nodes are regularized set operations.
- The creation of a balanced, unbalanced, or a perfect CSG tree depends solely on the user and how he/she decomposes a solid into its primitives.
- The general rule to create balanced trees is to start to build the model from an almost central position and branch out in two opposite directions or vice versa.
- Another useful rule is that symmetric objects can lead to perfect trees if they are decomposed properly.



Primitives



CSG tree of a typical solid



An unbalanced CSG tree

A balanced tree is defined as a tree whose left and right sub trees have almost an equal number of nodes.

A perfect tree is one whose $n_L - n_R$ is equal to zero.

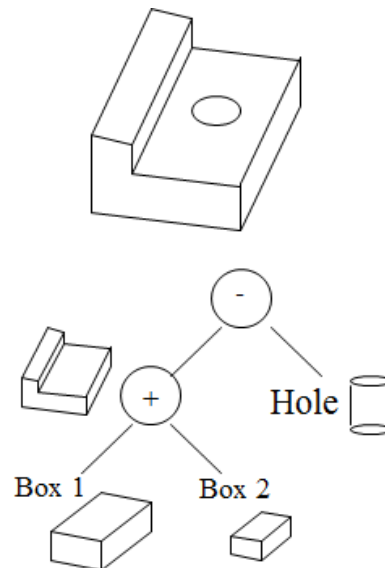
$$n_L = n_R = n - 1$$

$n \rightarrow$ number of primitives

$(n-1) \rightarrow$ number of Boolean operations

$(2n-1) \rightarrow$ number of nodes

Example:



Advantages of Solid Modeling:

- Memory required will be less.
- Creation of fully valid geometrical solid model.
- Complex shapes may be developed with the available set of primitives.
- Less skill is enough.
- Easy to construct out of primitives and Boolean operations.

Limitations of Solid Modeling:

- New computational effort and time are essential wherever the model is to be shown in the screen.
- Getting fillet, chamfer and taper in the model is very difficult.

Boundary Representation (B- rep)

- Boundary representation is one of the two most popular and widely used schemes to create solid models of physical objects.
- A B-rep model or boundary model is based on the topological notion that a physical object is bounded by a set of faces.
- These faces are regions or subsets of closed and orientable surfaces.
- A closed surface is one that is continuous without breaks.
- An orientable surface is one in which it is possible to distinguish two sides by using the direction of the surface normal to point to the inside or outside of the solid model under construction.
- Each face is bounded by edges and each edge is bounded by vertices.
- Thus, topologically, a boundary model of an object is comprised of faces, edges, and vertices of the object linked together in such a way as to ensure the topological consistency of the model.

Vertex (V) : It is a unique point (an ordered triplet) in space

Edge (E): It is finite, non-self intersecting, directed space curve bounded by two vertices that are not necessarily distinct

Face (F) : It is defined as a finite connected, non-self-intersecting, region of a closed oriented surface bounded by one or more loops

Loop (L) : It is an ordered alternating sequence of vertices and edges

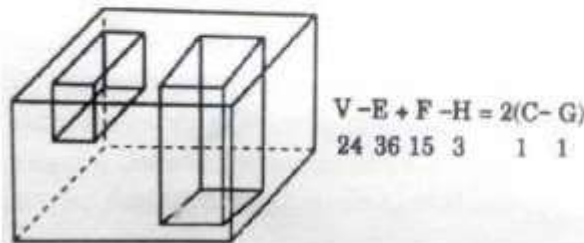
Genus (G) : It is the topological name for the number of handles or through holes in an object

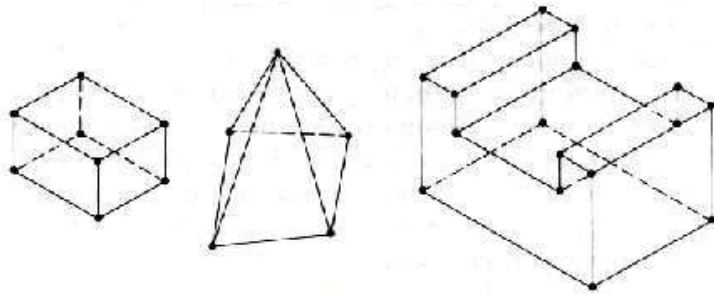
Body/Shell(B) : It is a set of faces that bound a single connected closed volume. A minimum body is a point

Euler's formula

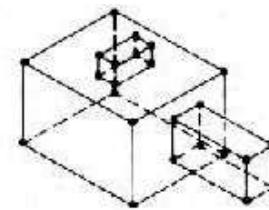
Euler – Poincaré Law for closed objects : $F - E + V - L = 2(B - G)$

Euler – Poincaré Law for open objects : $F - E + V - L = B - G$

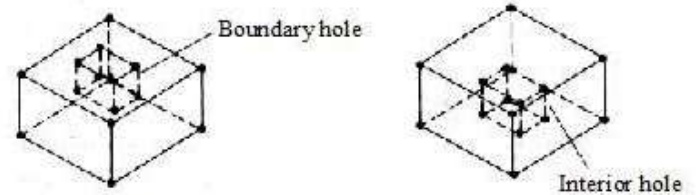




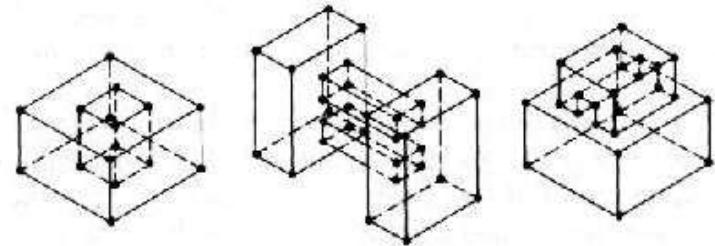
(a) Simple polyhedra



(b) Polyhedra with faces of inner loops



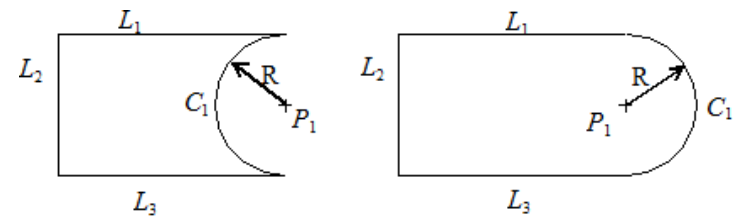
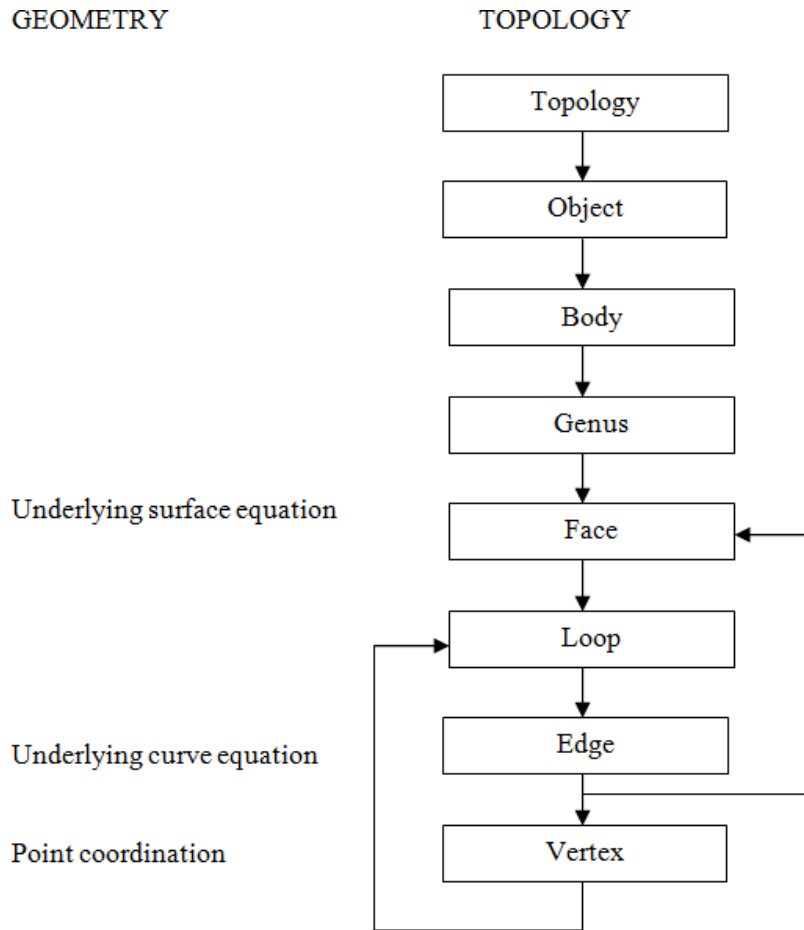
(c) Polyhedra with not through holes



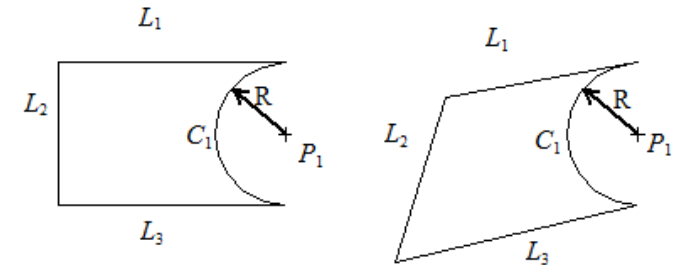
(d) Polyhedra with handles (through holes)

Solid number	F	E	V	L	B	G
1	6	12	8	0	1	0
2		8	5	0	1	0
3	10	24	16	0	1	0
4	16	36	24	2	1	0
5	11	24	16	1	1	0
6	12	24	16	0	2	0
7	10	24	16	2	1	1
8	20	48	32	4	1	1
9	14	36	24	2	1	1

Data Structure of B-rep



(a) Same geometry but different topology



(b) Same topology but different geometry

Advantages of B-rep:

- 1.It is traditionally a popular modeling method related closely to traditional drafting.
- 2.It is very suitable tool to build quite extraordinary shapes like aircraft and automobiles,
that are difficult to build using primitives .
- 3.It is comparatively simple to convert a B-rep model into a wireframe model because its boundary definition is similar to the wireframe definitions.
- 4.In applications B-rep algorithms are reliable and competitive to CSG based algorithms .

Limitations of B-Rep:

- 1.It requires large storage memory space as it stores the explicit definitions of the model boundaries.
2. Sometimes geometrically valid solids are not possible.
3. Approximate B-rep is not suitable for manufacturing applications.

Thank You

UNIT III

VISUAL REALISM

VISUALIZATION



- Visualization can be defined as a technique for creating images , diagrams or animations to communicate ideas.
- Projection and shading are common methods for visualizing geometric models.
- CAD uses isometric and perspective projection in addition to orthographic projection for generating rich visual images with complete design information.



- To project 3D to 2D objects we need to remove the ambiguities of the different views, which can be got by the elimination of hidden lines , surfaces , solid removal approaches.
- **Shading,**
- **Lighting**
- **Transparency**
- **Coloring** approaches provide more visual realism



Model clean up process

- Generate orthographic views
- Eliminate hidden lines
- Changing necessary hidden lines as dashed line or adding dimension and text to the different views.

Application of realism

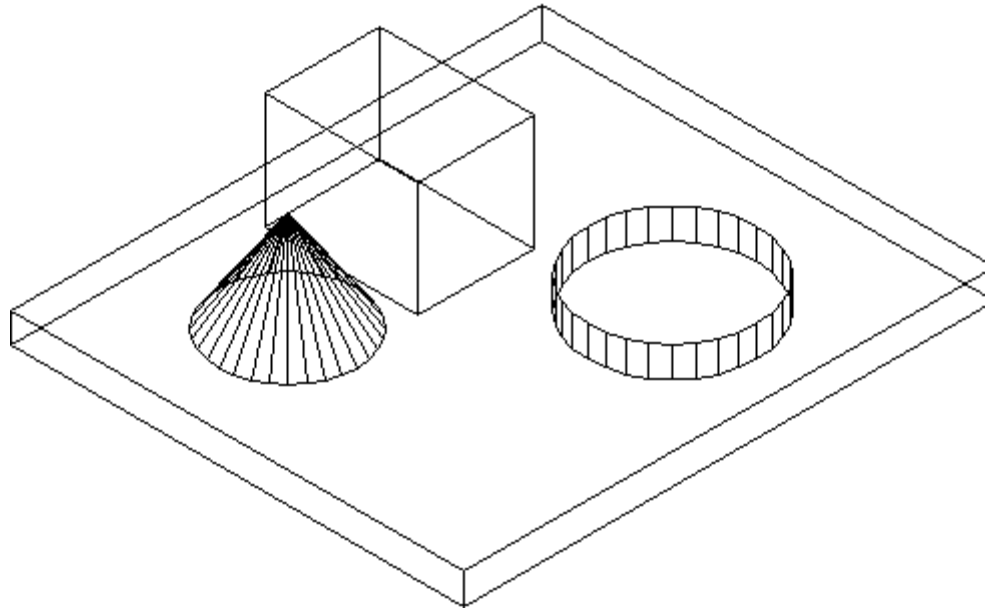


- **Robot Simulations** : Visualization of movement of their links and joints and end effector movement etc.
- CNC programs verification of tool movement along the path prescribed and estimation of cup height and surface finish etc.
- **Discrete Even Simulation** : Most of DES packages provide the user to create shop floor environment on the screen to visualize layout of facilities, movement of material handling systems, performance of machines and tools.
- **Scientific Computing** : Visualization of results of FEM analysis like iso-stress and iso-strain regions, deformed shapes and stress contours. Temperature and heat flux in heat-transfer analysis. Display and animation of mode shape in vibration analysis.
- **Flight Simulation** : Cockpit training for pilots is first being provided with flight simulators, which virtually simulates the surrounding that an actual flight will pass through.

No Lines Removed



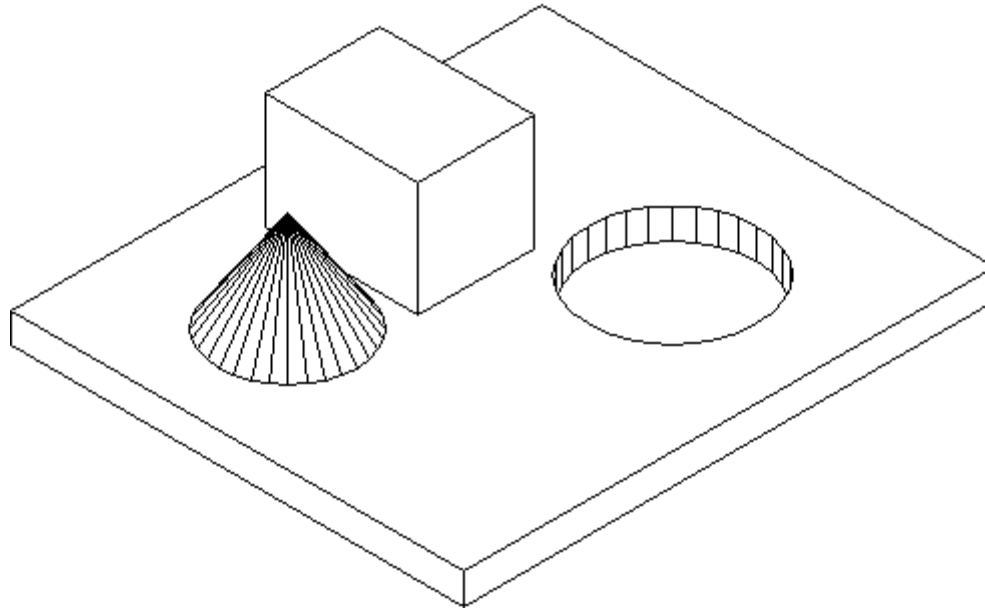
Wireframe



Hidden Lines Removed



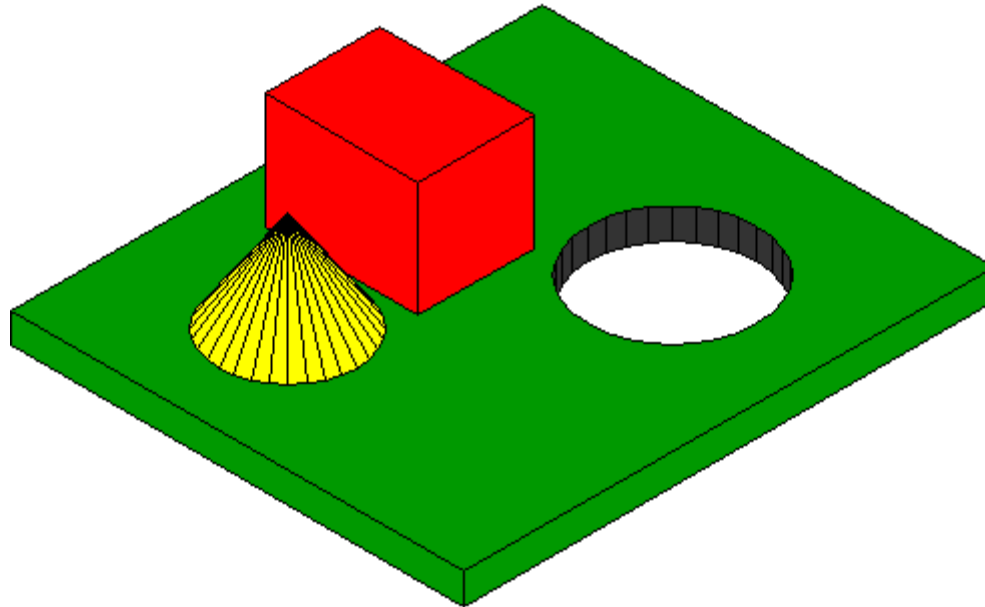
Hidden Line Removal



Hidden Surfaces Removed



Hidden Surface Removal





HIDDEN LINE REMOVAL

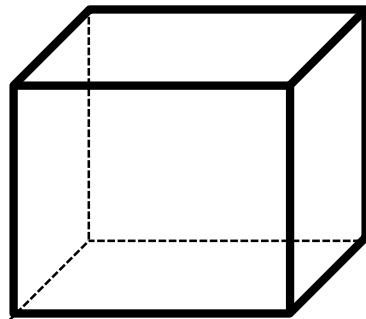
- *“For a given three dimensional scene, a given viewing point and a given direction eliminate from an appropriate two dimensional projection of the edges and faces which the observer cannot see”*
- **Object space method**
- **Image space method**



Two main types of algorithms:

- **Object space:** Determine which part of the object are visible. Also called as World Coordinates. Object is described in physical coordinate system.
- It compares the object and parts to each other within the scene definition to determine which surface is visible.

- **Image space:** Determine per pixel which point of an object is visible. Also called as Screen Coordinates. Visibility is decided point by point at each pixel position on view plane.
- Zooming does not degrade the quality.



Object space

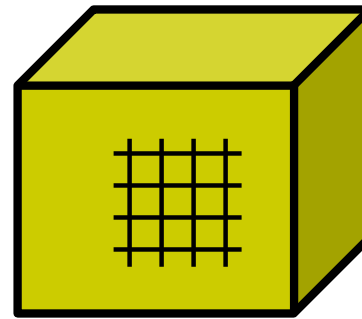


Image space



*Two main Hidden Surface Removal Algorithm
Techniques:*

*Object space: Hidden Surface Removal for all
objects*

Image space:

Objects: 3 D Clipping

Transformed to Screen Coordinates

Hidden Surface Removal

HIDDEN LINE ELIMINATION PROCESS



3D OBJECT DATA

GEOMETRY AND TOPOLOGY



2D OBJECT DATA

TRANSFORMATIONS-CONTAINS VISIBLE AND INVISIBLE EDGES.



SORTING OF 2D IMAGE DATA



APPLICATION OF VISIBILITY TECHNIQUES

CHECKS FOR OVERLAPPING

DEPTH COMPARISON IS USED TO DETERMINE PART OR ALL OF POLYGON IS
HIDDEN



ELIMINATION OF HIDDEN LINES



DISPLAY OF RESULTS

VISIBILITY TECHNIQUES



- MINIMAX TEST
- CONTAINMENT TEST
- SURFACE TEST
- COMPUTING SILHOUETTES
- EDGE INTERSECTION
- SEGMENT COMPARISONS
- HOMOGENITY TEST

MINIMAX TEST

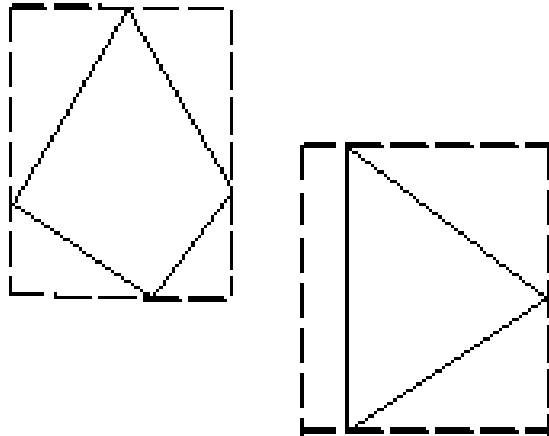


- Minimax test (also called the overlap or bounding box test) checks if two polygons overlap. **The test provides a quick method to determine if two polygons do not overlap.**
- It surrounds each polygon with a box by finding its extents (minimum and maximum x and y coordinates) and then checks for the intersection for any two boxes in both the X and Y directions.
- If two boxed do not intersect, their corresponding polygons do not overlap (see Figure 1). In such a case, no further testing of the edges of the polygons is required.
- If the minimax test fails (two boxes intersect), the two polygons may or may not overlap, as shown in Figure 1. Each edge of one polygon is compared against all the edges of the other polygon to detect intersections. The minimax test can be applied first to any two edges to speed up this process

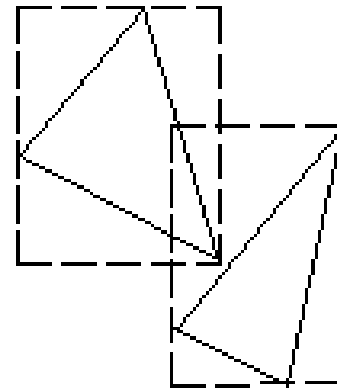
MINIMAX TEST



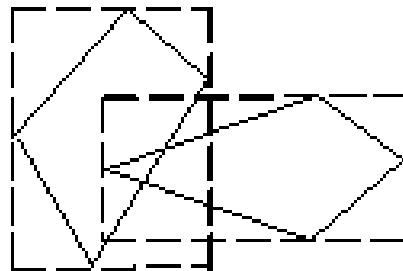
Y_v



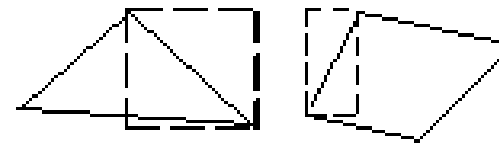
(a) Boxes and polygons do not overlap



(b) Boxes overlap and polygons do not



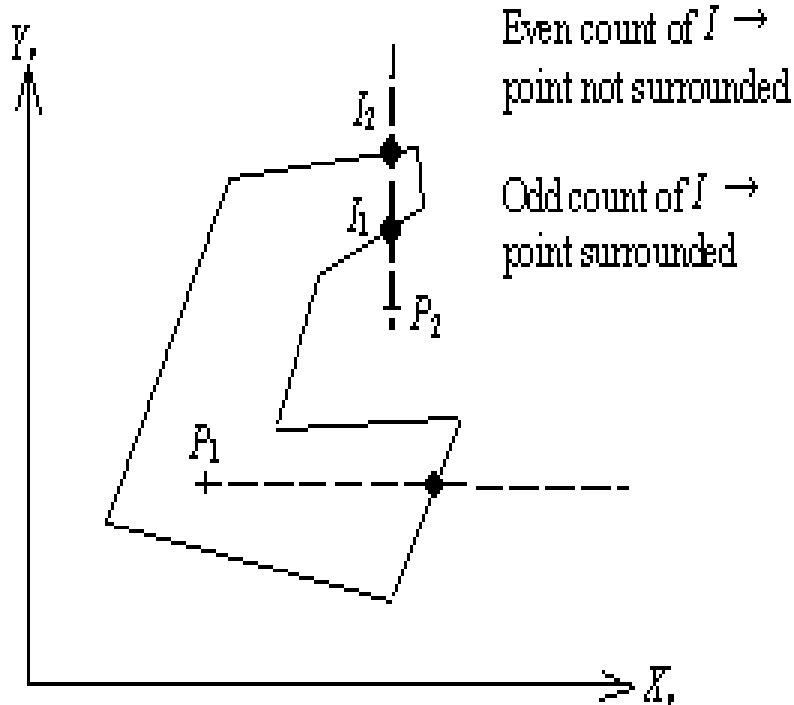
(c) Boxes and polygons overlap



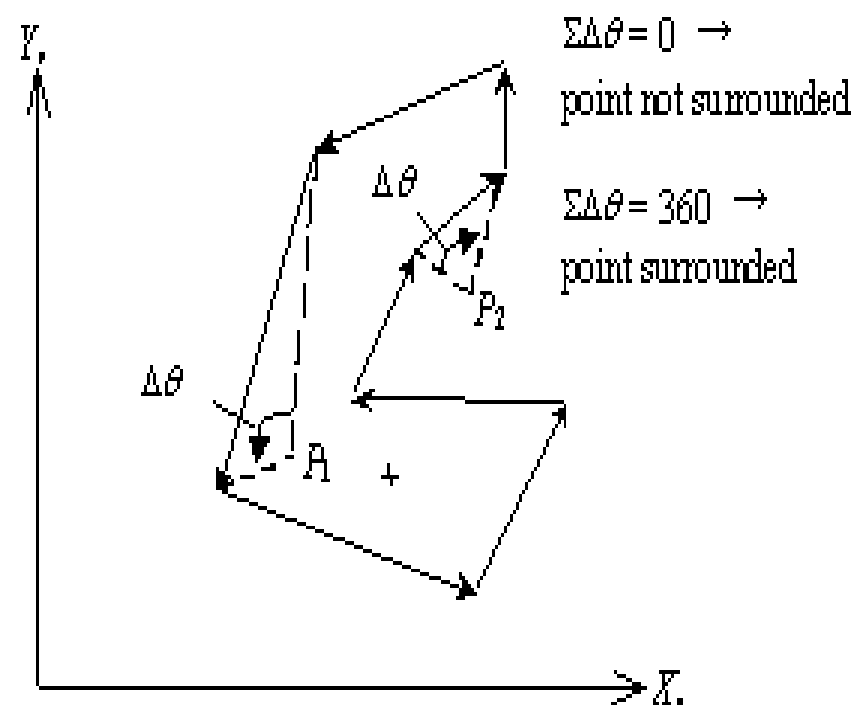
(d) Minimax test of individual edges

X_v

CONTAINMENT TEST



(a) Intersection method



(b) Angle method

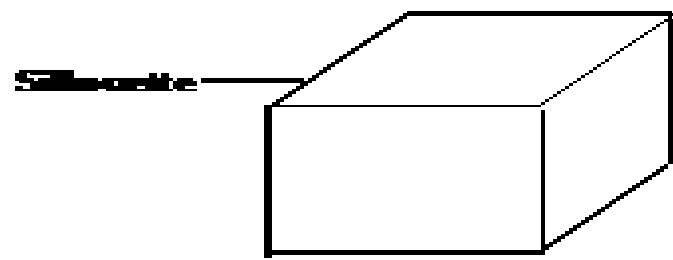
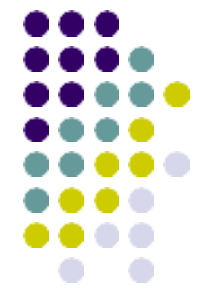


- **The containment test checks whether a given point lies inside a given polygon or polyhedron.** There are three methods to compute containment or surroundedness.
- For a convex polygon, one can substitute the X and Y coordinates of the point into the line equation of each edge. If **all substitutions result in the same sign, the point is on the same side of each edge and is therefore surrounded.**
- For non-convex polygons, two other methods can be used. In the first method, we draw a line from the point under testing to infinity as shown in Figure 2a. The **semi-infinite line is intersected with the polygon edges.** If the **intersection count is even**, the point is **outside the polygon** (in Figure 2a). If it is **odd, the point is inside.**
- If one of the polygon edges lies on the semi-infinite line, a singular case arises which needs special treatment to guarantee the consistency of the results.
- The second method for non-convex polygons (Figure 2b) **computes the sum of the angles subtended by each of the oriented edges** as seen from the test point. If the **sum is zero**, the point is **outside the polygon.** If the sum is -360 or $+360$ the point is inside. The minus sign reflects whether the vertices of the polygon are ordered in a clockwise direction instead of counter clockwise.

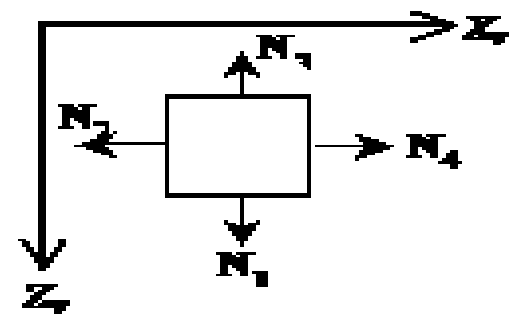
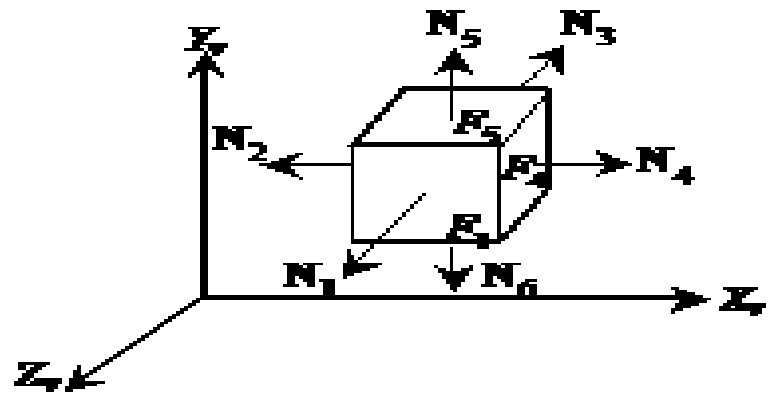
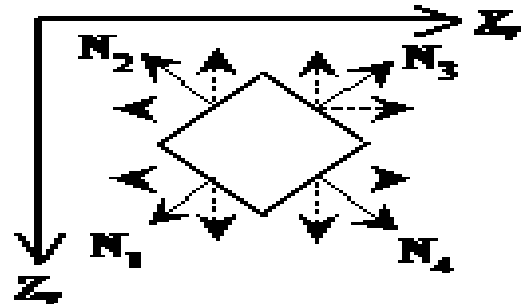
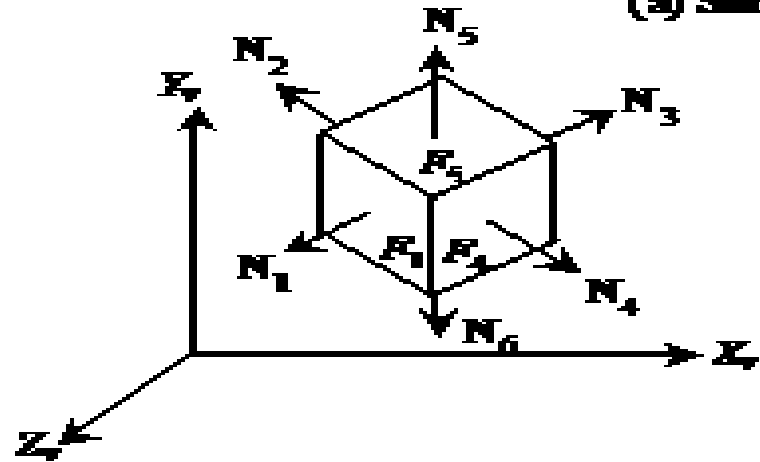


Computing silhouettes

- A set of edges that separates visible faces from invisible faces of an object with respect to a given viewing direction is called silhouette edges (or silhouettes).
- The signs of the components of normal vectors of the object faces can be utilized to determine the silhouette.
- An edge that is part of the silhouette is characterized as the intersection of one visible face and one invisible face.
- An edge that is the intersection of two visible faces is visible, but does not contribute to the silhouette.
- The intersection of two invisible faces produces an invisible edge.



(a) Silhouette edges

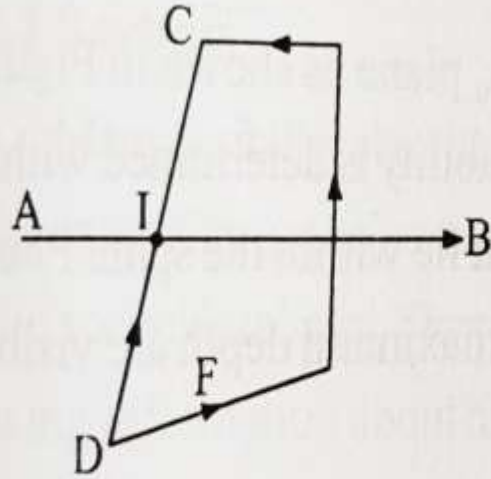


(b) Determining silhouette edges

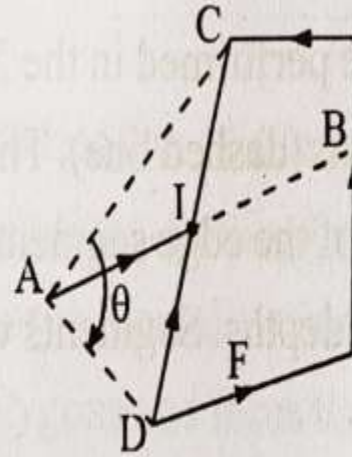


Edge intersection

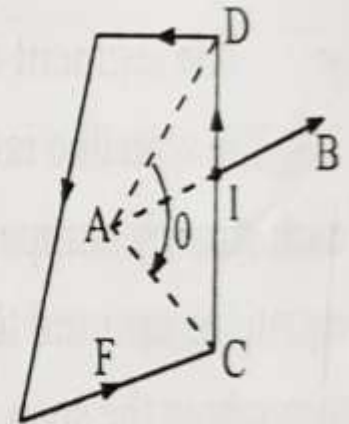
- The hidden algorithm initially calculates the edges intersections in 2D.
- To find out partially visible lines.
- The two edges intersect at a point where $y_2 - y_1 = 0$.
- Then segment comparisons are used to further determine visibility



(a) Fully visible edge AB



(b) CD marks the disappearance of partially hidden edge AB

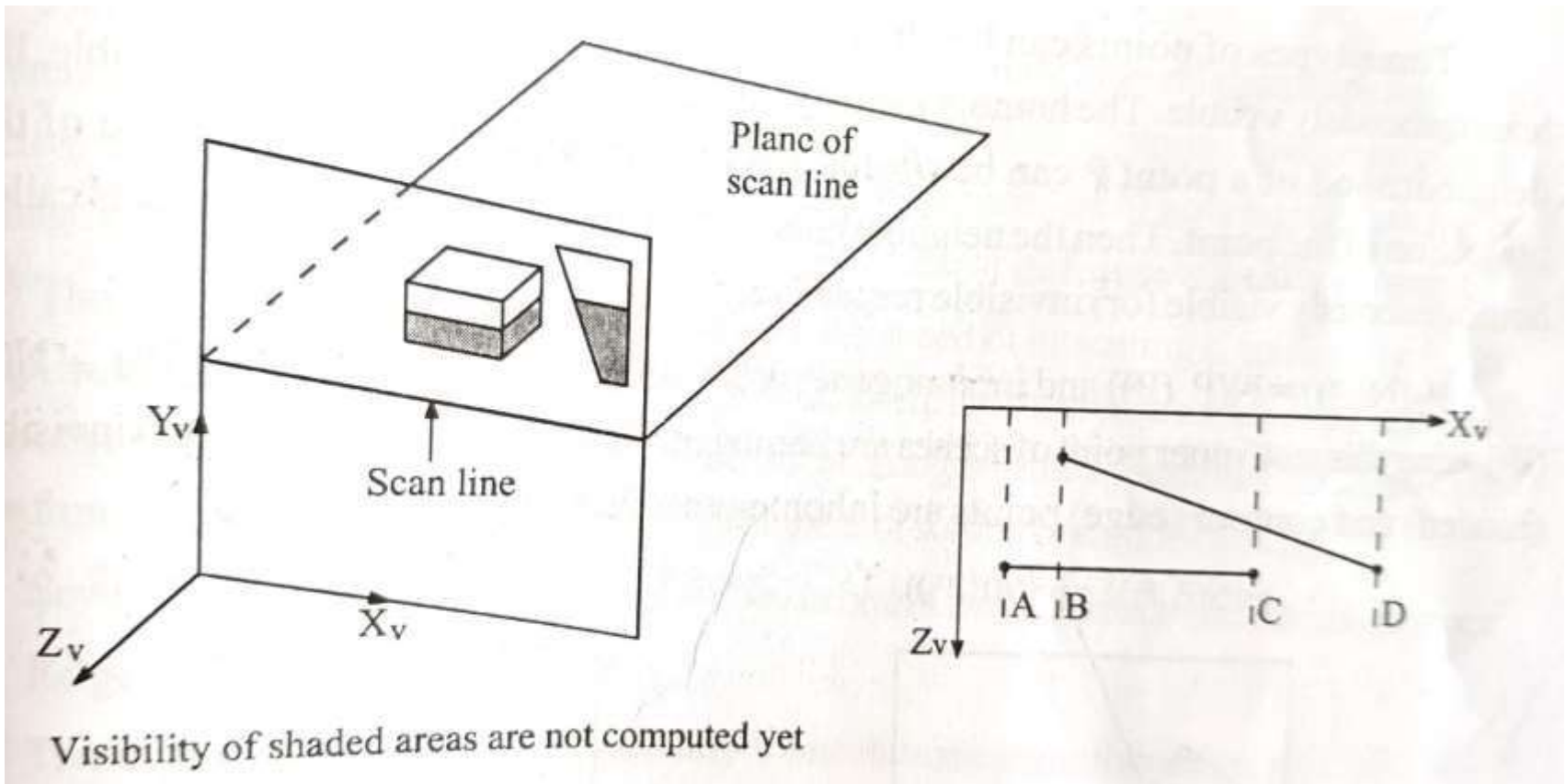


(c) CD marks the appearance of partially hidden edge AB

Segment comparison



- The image is computed scan line by line that is in segments and displayed in the same order.
- The scan line is divided into spans(dashed lines).
- The visibility is determined within each span by comparing the depths of the edge segments that lie in the span.
- Segments with maximum depth are visible throughout the span.



Homogeneity test

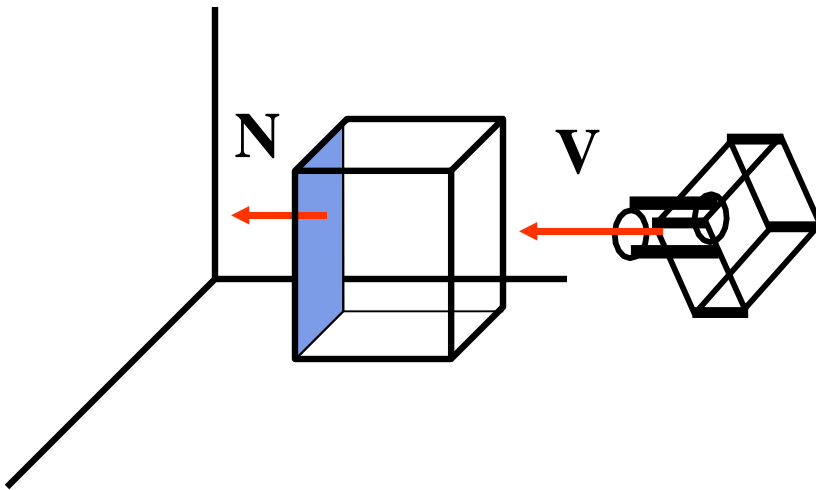


- Points are compared for visibility.
- Homogeneously visible
 - Neighborhood of point P can be projected objectively onto neighborhood of the projection of a point.
- Homogeneously invisible
 - Neighborhood of point P cannot be projected objectively onto neighborhood of the projection of a point.
- In-homogeneously visible
 - $P_r(N(P))=N(P_r(P))$
 - $P_r(N(P))\neq N(P_r(P))$ inhomogeneously invisible.

Surface Back-face elimination



We cannot see the back-face of solid objects:
Hence, these can be ignored

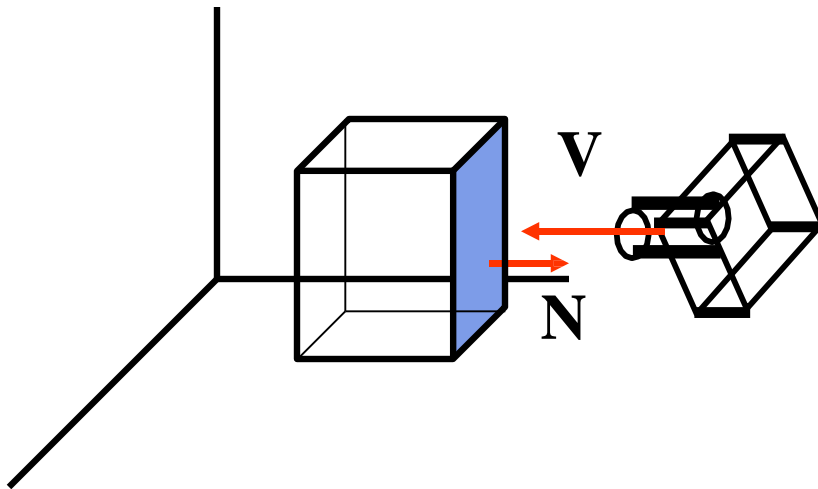


$\mathbf{V} \cdot \mathbf{N} > 0$: back face



Back-face elimination

We cannot see the back-face of solid objects:
Hence, these can be ignored

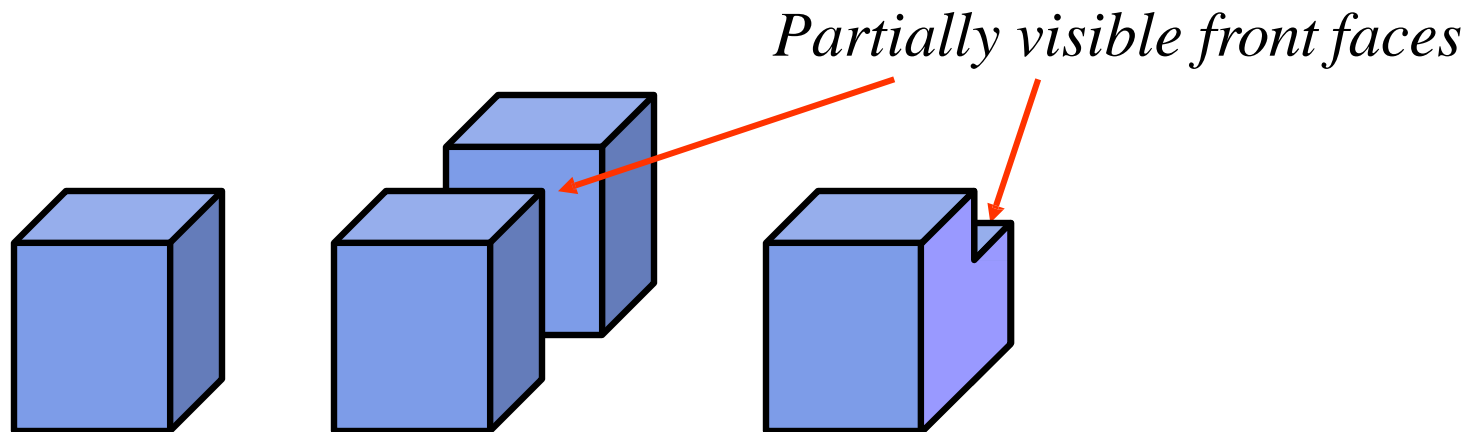


$\mathbf{V} \cdot \mathbf{N} < 0$: front face



Back-face elimination

- Object-space method
- Works fine for convex polyhedra: $\pm 50\%$ removed
- Concave or overlapping polyhedra: require additional processing
- Interior of objects can not be viewed





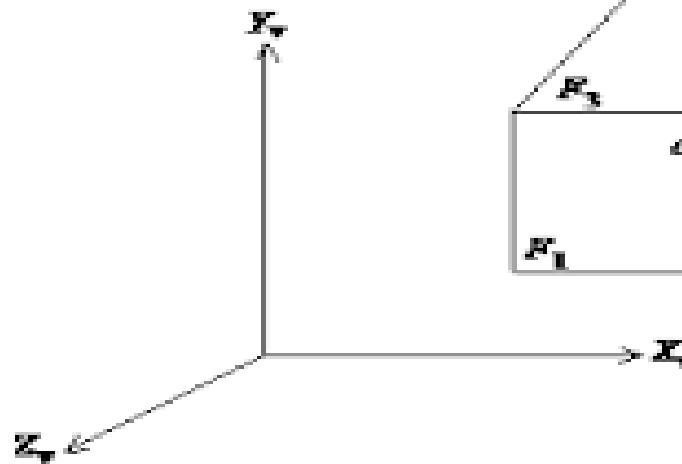
- **Hidden line removal algorithm**
 - Depth Algorithm or Z algorithm or Priority algorithm
 - Area oriented algorithms
 - Overlay algorithm-Curved surface
 - Roberts algorithm
- **Hidden surface removal algorithm**
 - Depth buffer algorithm or z-buffer algorithm
 - Area coherence algorithm or Warnock's algorithm
 - Scan-line algorithm or Watkin's algorithm
- **Hidden solid removal algorithm**
 - Ray tracing algorithm

Depth or priority algorithm



- This algorithm is also known as the depth or z algorithm. The algorithm is based on sorting all the faces (polygons) in the scene according to the largest z coordinate value of each.
- This step is sometimes known as assignment of priorities. If a face intersects more than one face, other visibility tests besides the z depth are needed to resolve any ambiguities.
- Its basis is on the view according to the biggest Z co-ordinate value.
- If face intersects more than one face, other visibility test beside z-depth is required to solve any issue.

The priority algorithm



Face list	Priority list
F ₁	1
F ₂	1
F ₃	1
F ₄	2
F ₅	
F ₆	

Iteration 1

Face list	Priority list
F ₂	1
F ₃	1
F ₄	2
F ₅	
F ₆	
F ₁	

Iteration 2

Face list	Priority list
F ₃	1
F ₄	2
F ₅	
F ₆	
F ₁	
F ₂	

Iteration 3

Face list	Priority list
F ₄	2
F ₅	2
F ₆	2
F ₁	1
F ₂	1
F ₃	1

Iteration 4

(b) Assignment of priorities

Depth or priority algorithm

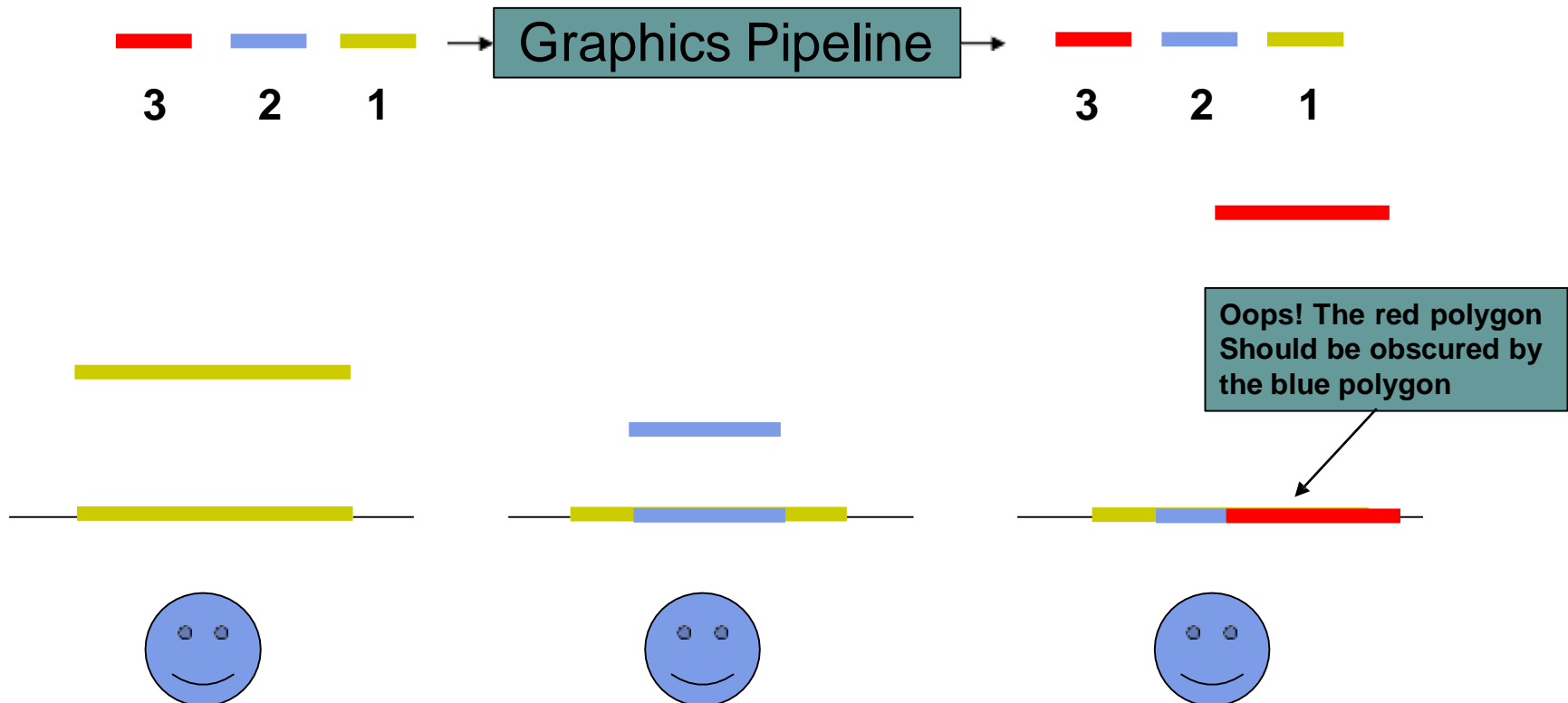


- **Painter's algorithm**
 - As we utilize the procedure the painter's way of creating the background first and then the overlaying layer and then the outermost layer with reducing depth.
- When we view from z and x axis there is no overlapping view.



Painter's Algorithm

- Assumption: Later projected polygons overwrite earlier projected polygons



Painter's Algorithm

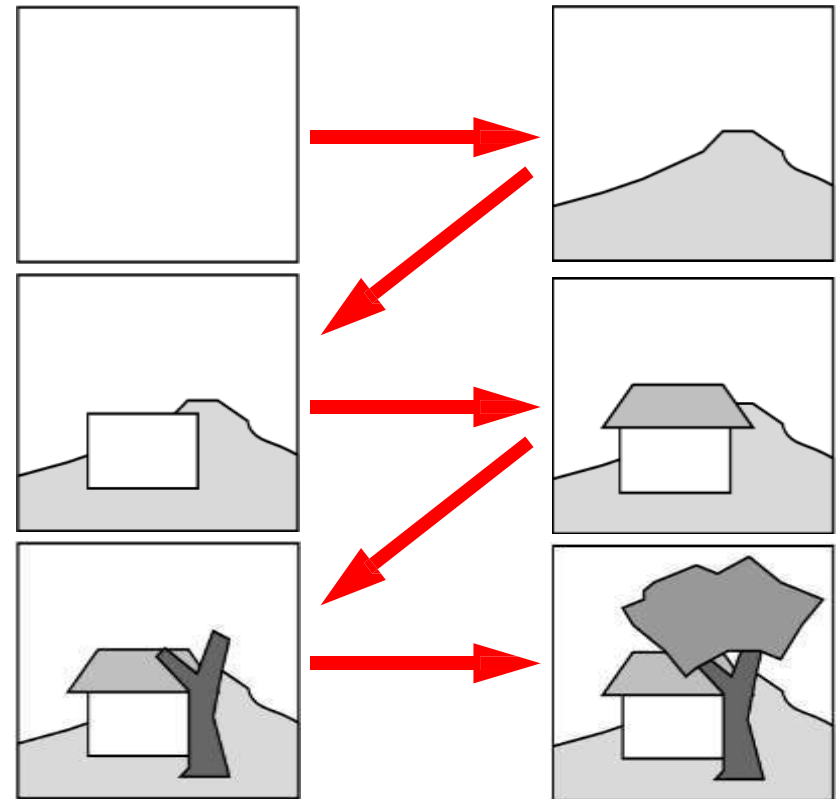


- Main Idea

- A painter creates a picture by drawing background scene elements before foreground ones

- Requirements

- Draw polygons in back-to-front order
- Need to **sort** the polygons by depth order to get a correct image

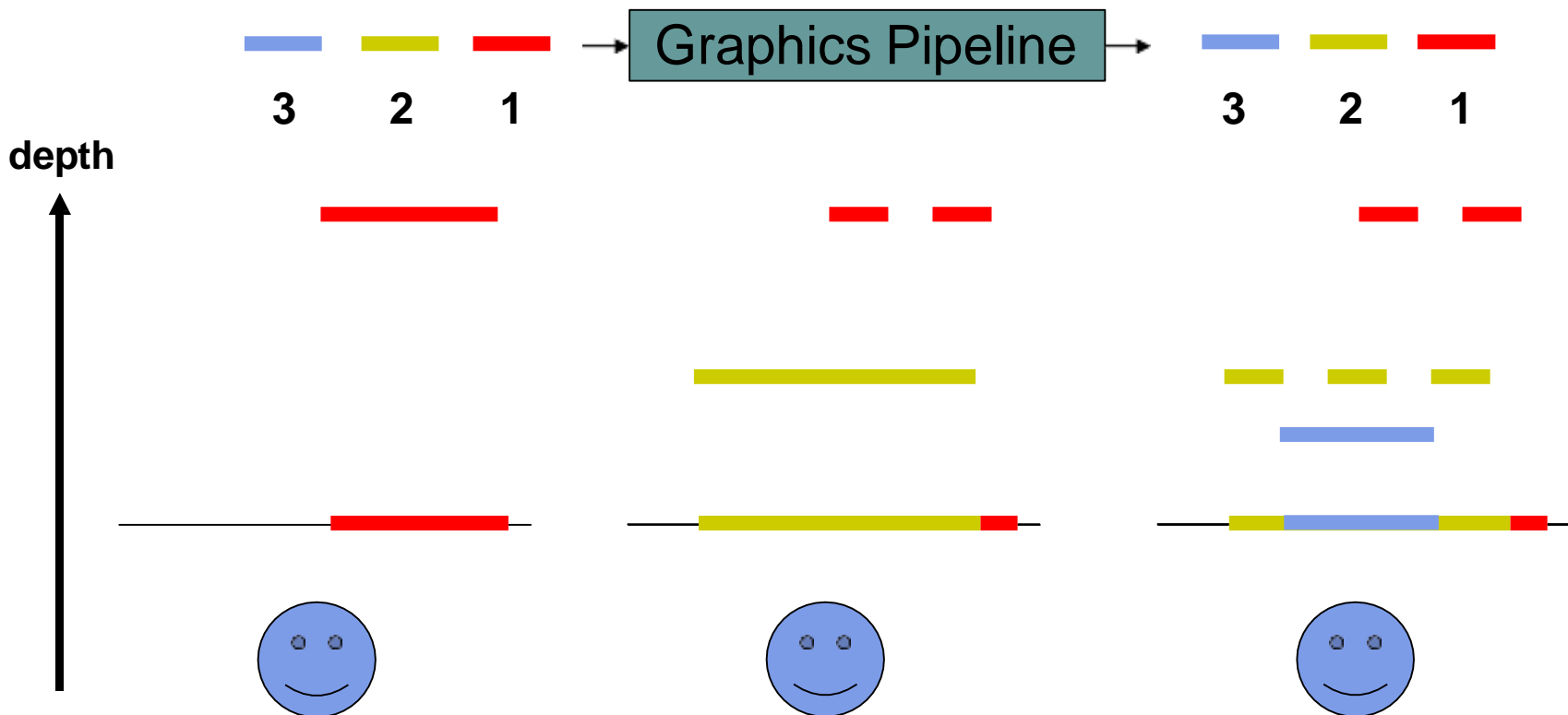


from Shirley



Painter's Algorithm

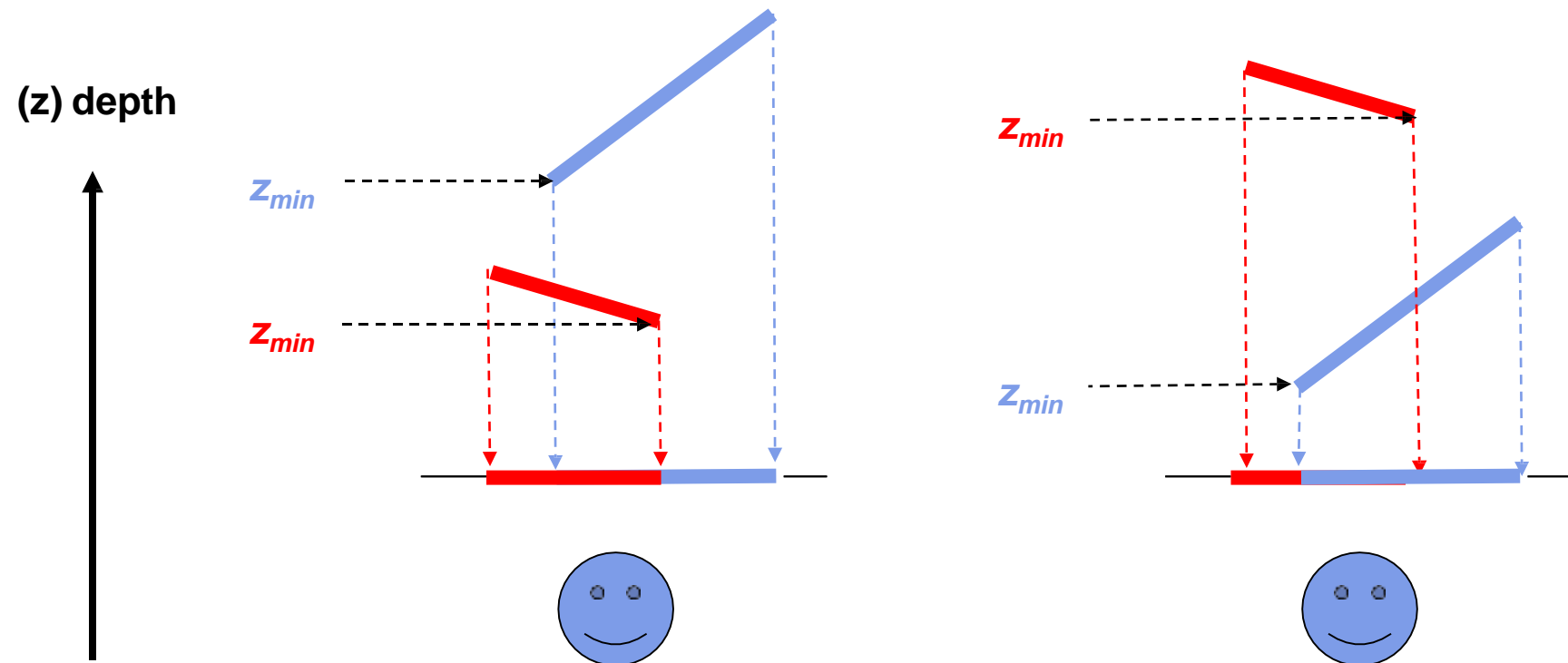
- Sort by the depth of each polygon





Painter's Algorithm

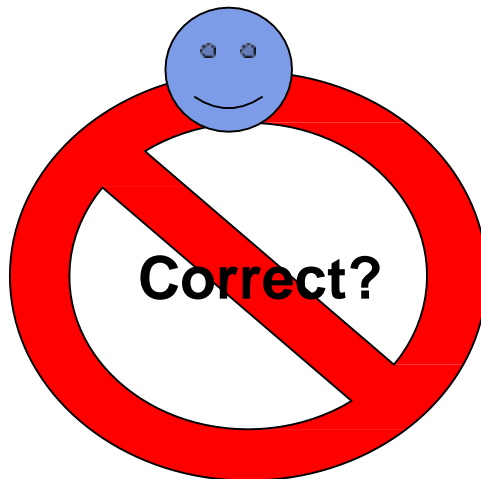
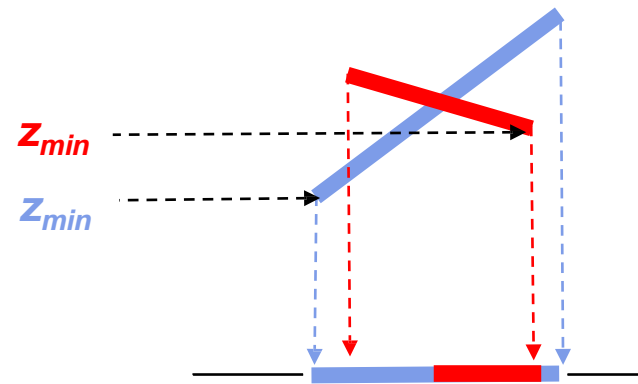
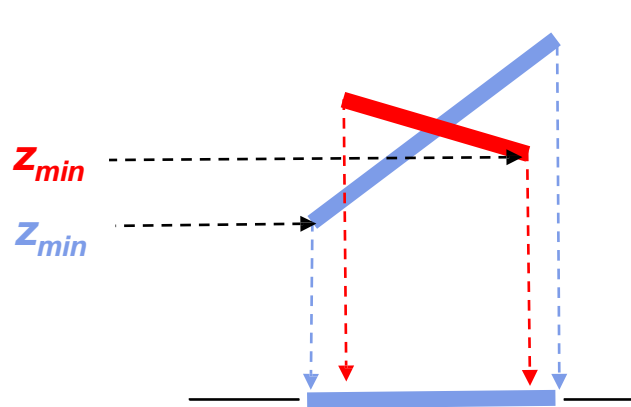
- Compute z_{min} ranges for each polygon
- Project polygons with furthest z_{min} first





Painter's Algorithm

- Problem: Can you get a total sorting?



Area oriented algorithm



It is based on subdivision of given data set in a stepwise fashion until all visible areas are determined and displayed.

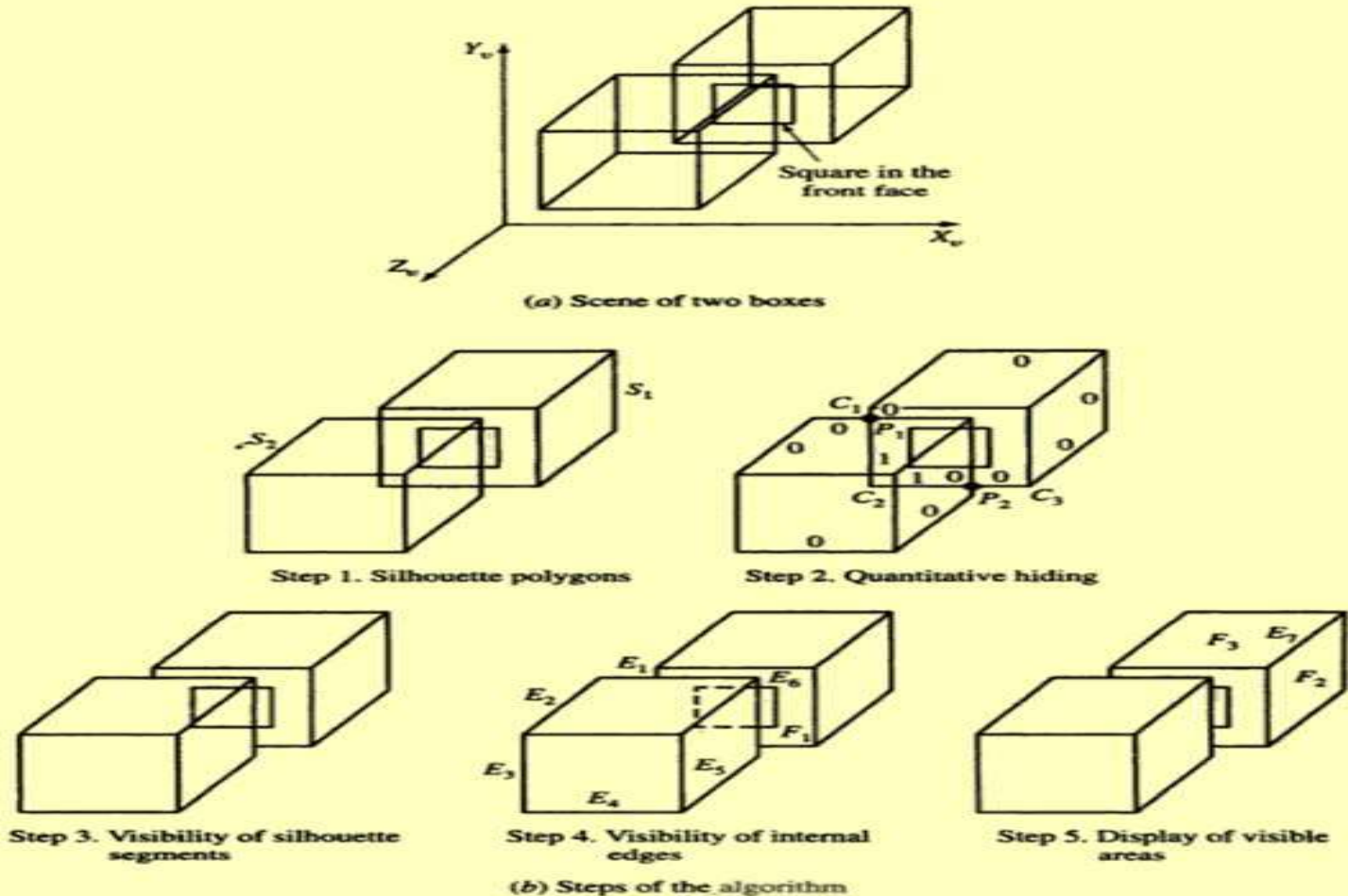


Figure 13.14 Area-oriented algorithm.



- **Identify silhouette polygons** – silhouette edges are recognized and connection of silhouette edges to form closed polygons by sorting all edges for equal end points
- **Assign quantitative hiding(QH) values to silhouette polygon.**
 - This is achieved by intersecting the polygons. The intersection points define points where QH may change. Find QH value using depth test. 0 is visible and 1 is invisible.
- **Determine the visible silhouette segments.**
 - If closed silhouette polygon is completely invisible it need not be considered any further. If it is visible the segments with least QH values are considered.



- **Intersect the visible silhouette segments with partially visible faces.**
 - To find out the partially visible and fully visible faces.
- **Display the interior of the visible or partially visible polygons.**
 - By using stack and simply enumerates all faces lying inside a silhouette polygon.
 - The stack is started with a visible face and a loop begins popping of face F2

Overlay algorithm



- The curved surfaces are approximated as planar surfaces.
- The u-v grid is used to create grid surface which consists of regions having straight edges.
- The curves in each region are approximated as line segment.
- The 1st step is to use surface equation and grid linear edges are created.

Hidden line removal for curved surface



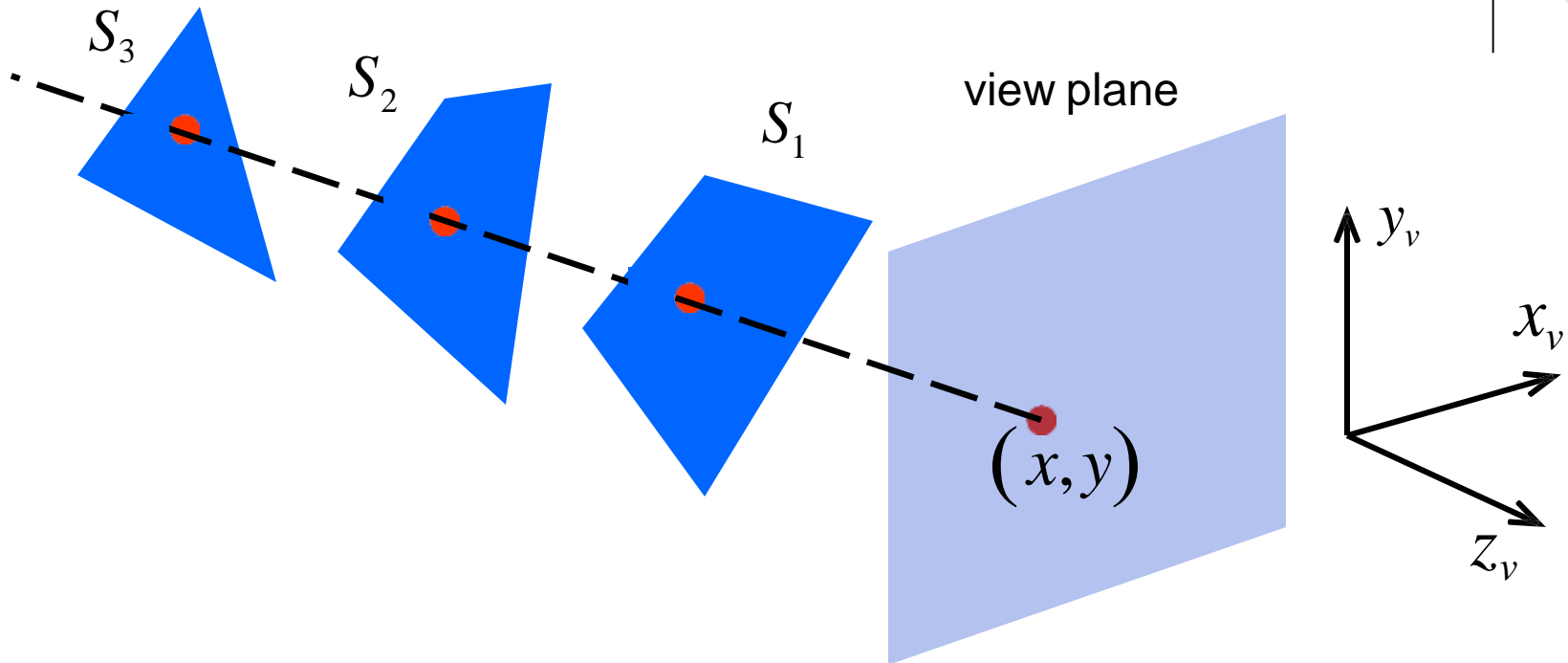
- To compute the exact visibility we introduce a notion of **visibility curves** obtained by projection of silhouette and boundary curves and decomposing the surface into nonoverlapping regions.
- The nonoverlapping and visible portions of the surface are represented as trimmed surfaces and we present a representation based on polygon trapezoidation algorithms.
- The curved surface is converted in polygon mesh and calculated for visibility.

Hidden surface removal algorithm



- The elimination of parts of a solid objects that are covered by others is called hidden surface removal.
- *Depth buffer or Z-buffer Algorithm*
- *Area coherence or Warnock's algorithm*
- *Scan-line algorithm or Watkin's algorithm*

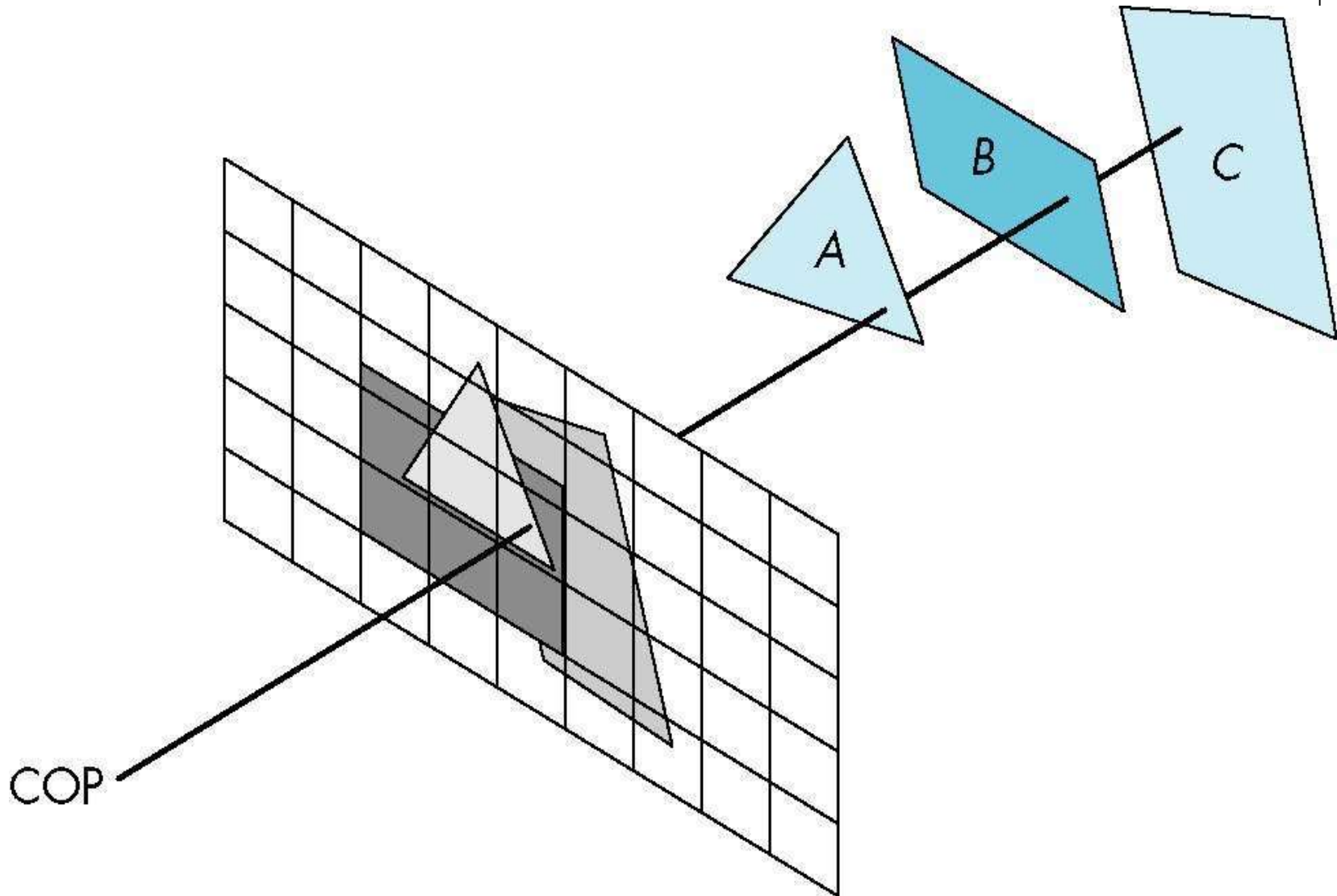
Depth-Buffer Methods



Three surfaces overlapping pixel position (x, y) on the view plane.

The visible surface, S_1 , has the smallest depth value.

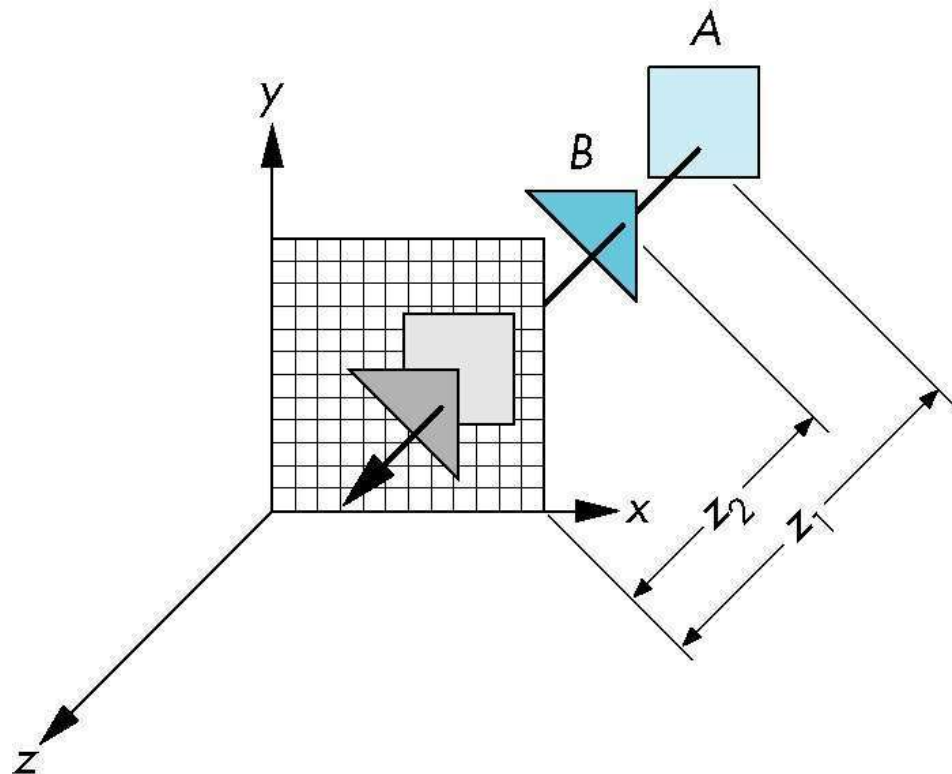
Depth buffer or z-buffer algorithm



Z-Buffer Algorithm



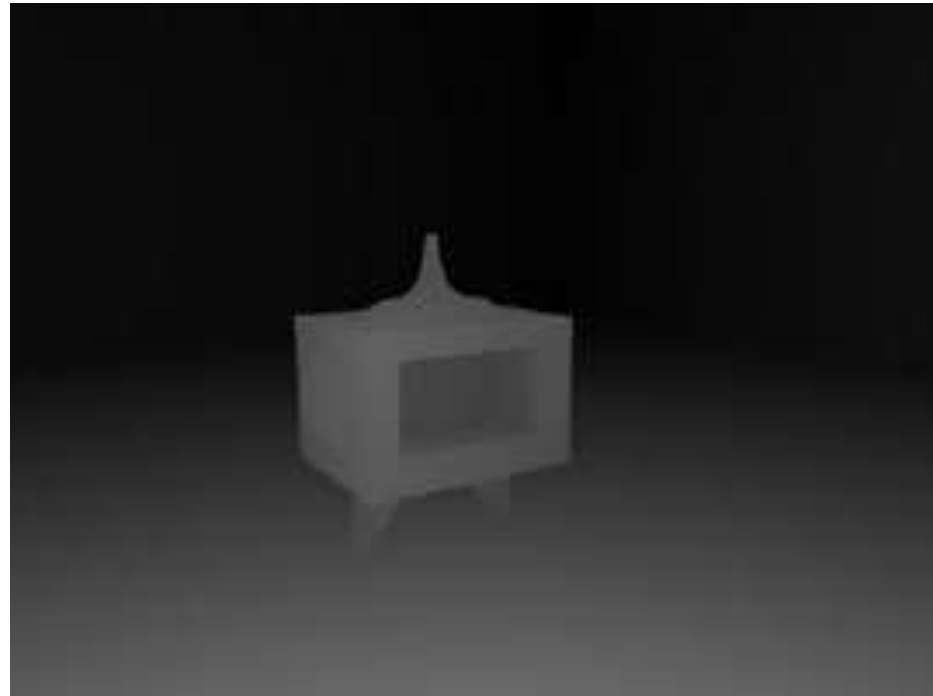
- As we render each polygon, compare the depth of each pixel to depth in z buffer
- If less, place shade of pixel in color buffer and update z buffer



Z-buffer: A Secondary Buffer



Color buffer



Depth buffer



- ◆ Two buffer areas are required
 - Depth buffer
 - Store depth values for each (x, y) position
 - All positions are initialized to minimum depth
 - Usually 0 – most distant depth from the viewplane
 - Refresh buffer
 - Stores the intensity values for each position
 - All positions are initialized to the background intensity



Z-BUFFER ALGORITHM:

- Its an extension of Frame Buffer
- Display is always stored on Frame Buffer
- Frame Buffer stores information of each and every pixel on the screen
- Bits (0, 1) decide that the pixel will be ON or OFF
- Z- Buffer apart from Frame buffer stores the depth of pixel
- After analyzing the data of the overlapping polygons, pixel closer to the eye will be updated
- Resolution of X,Y => Array[X,Y]

Given set of polygon in image space

Z-Buffer Algorithm:



1. Set the frame buffer & Z-Buffer to a background value

(Z-BUFFER= Z_{\min}) where, Z_{\min} is value

To display polygon decide \Rightarrow color, intensity and depth

2. Scan convert each polygon

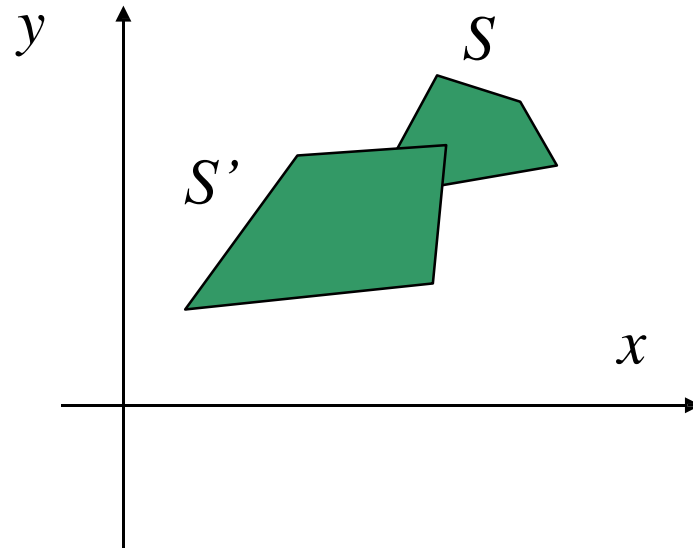
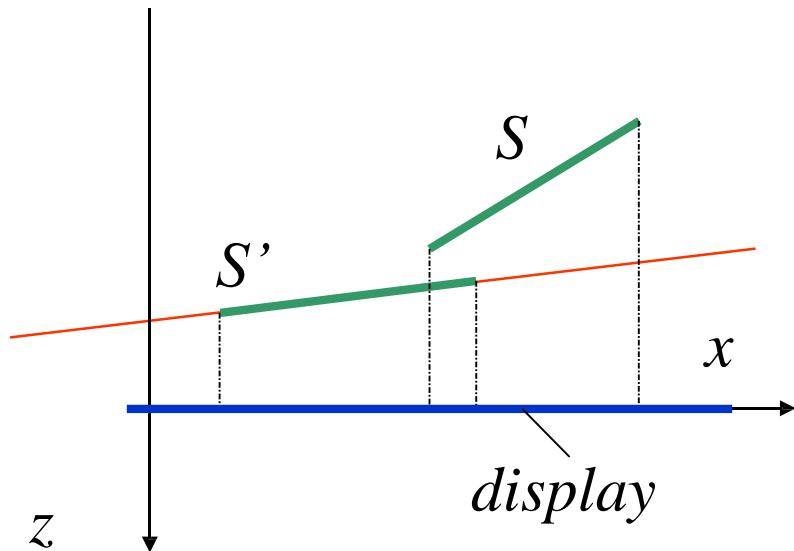
- i. e, for each pixel, find depth at that point If $Z(X, Y) > Z\text{-BUFFER}(X, Y)$

Update $Z\text{-BUFFER}(X, Y) = Z(X, Y)$

& FRAME BUFFER

This process will repeat for each pixel

- By this way we can remove hidden lines and display those polygons which are closer to eye
- $X*Y$ space required to maintain Z-Buffer=> $X*Y$ times will be scanned
- Expensive in terms of time and space as space is very large



Area coherence or warnock's algorithm

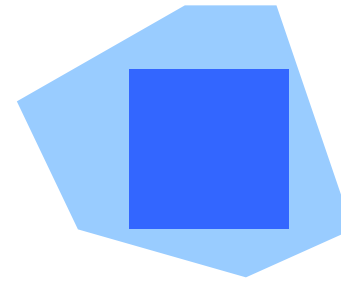
Area Subdivision



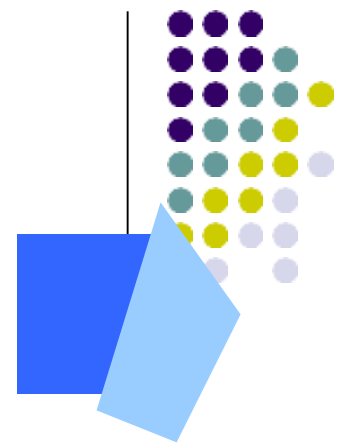
- Exploits *area coherence*: Small areas of an image are likely to be covered by only one polygon
- Three easy cases for determining what's in front in a given region:
 1. a polygon is completely in front of everything else in that region
 2. no surfaces project to the region
 3. only one surface is completely inside the region, overlaps the region, or surrounds the region

Identifying Tests

- Four possible relationships
 - Surrounding surface
 - Completely enclose the area
 - Overlapping surface
 - Partly inside and partly outside the area
 - Inside surface
 - Outside surface
- No further subdivisions are needed if one of the following conditions is true
 - All surface are outside surfaces with respect to the area
 - Only one inside, overlapping, or surrounding surface is in the area
 - A surrounding surface obscures all other surfaces within the area boundaries → from depth sorting, plane equation



Surrounding
Surface



Overlapping
Surface



Inside
Surface



Outside
Surface

Warnock's Area Subdivision

(Image Precision)



- Start with whole image
- If one of the easy cases is satisfied (previous slide), draw what's in front
- Otherwise, subdivide the region and recurse
- If region is single pixel, choose surface with smallest depth
- Advantages:
 - No over-rendering
 - Anti-aliases well - just recurse deeper to get sub-pixel information
- Disadvantage:
 - Tests are quite complex and slow

Characteristics

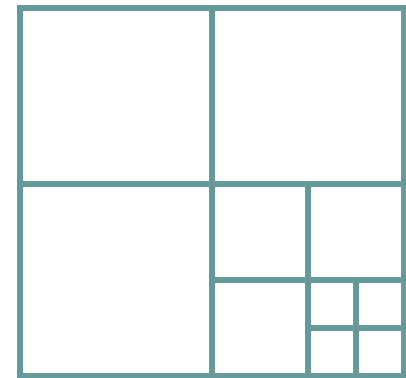


- Takes advantage of area coherence
 - Locating view areas that represent part of a single surface
 - Successively dividing the total viewing area into smaller rectangles
 - Until each small area is the projection of part of a single visible surface or no surface
 - Require tests
 - Identify the area as part of a single surface
 - Tell us that the area is too complex to analyze easily
- Similar to constructing a *quadtree*



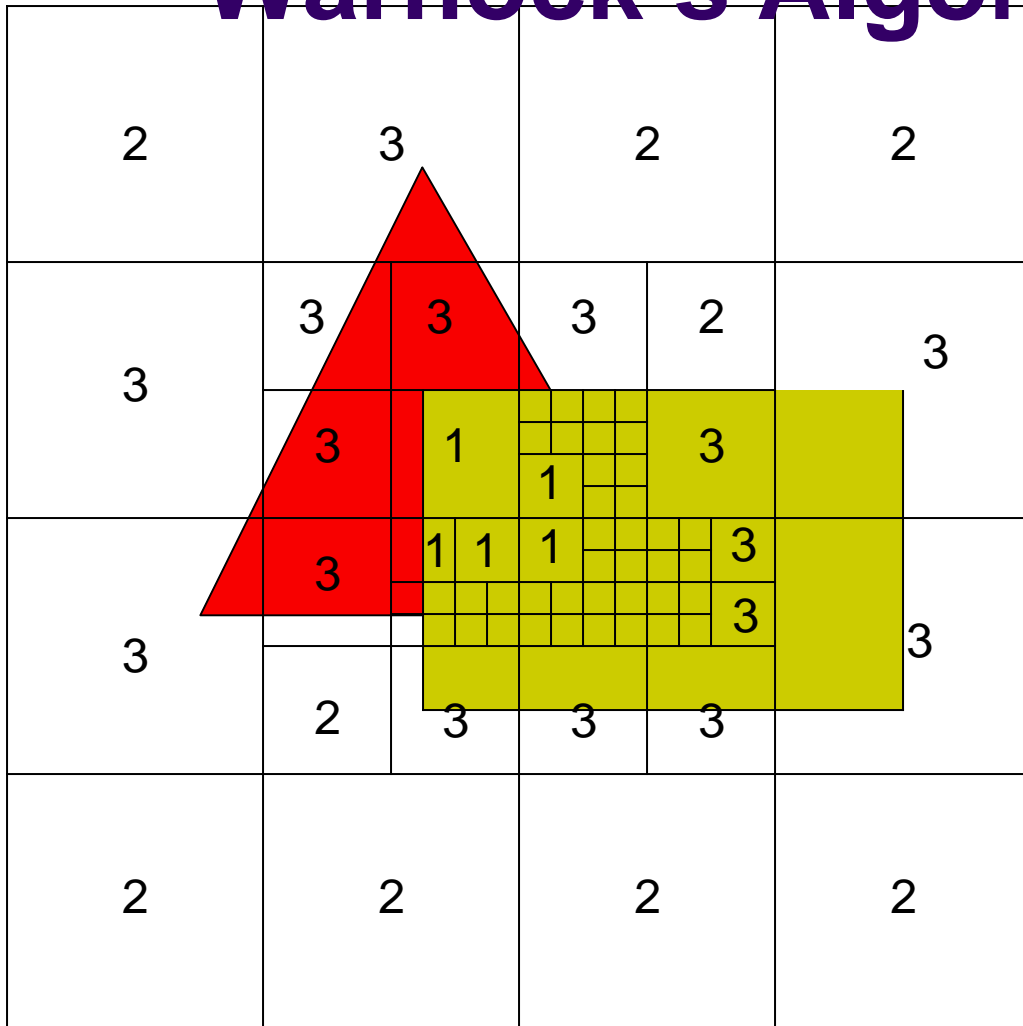
Process

- Staring with the total view
 - Apply the identifying tests
 - If the tests indicate that the view is sufficiently complex
 - Subdivide
 - Apply the tests to each of the smaller areas
 - Until belonging to a single surface
 - Until the size of a single pixel
- Example
 - With a resolution 1024×1024
 - 10 times before reduced to a point





Warnock's Algorithm

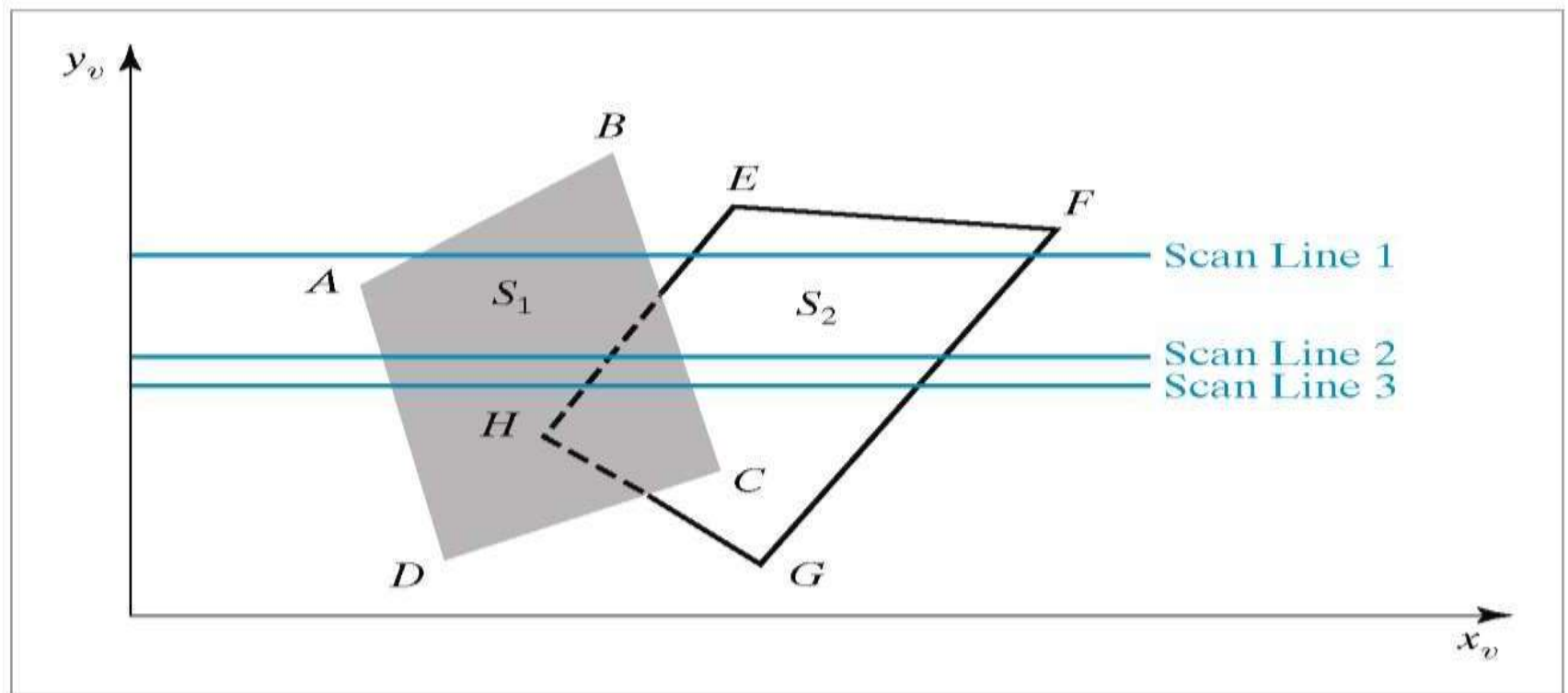


- Regions labeled with case used to classify them:
 - 1) One polygon in front
 - 2) Empty
 - 3) One polygon inside, surrounding or intersecting
- Small regions not labeled

SCAN LINE Z-BUFFER ALGORITHM:



- An image space method for identifying visible surfaces
- Computes and compares depth values along the various scan-lines for a scene



Scan-Line Method Basic Example

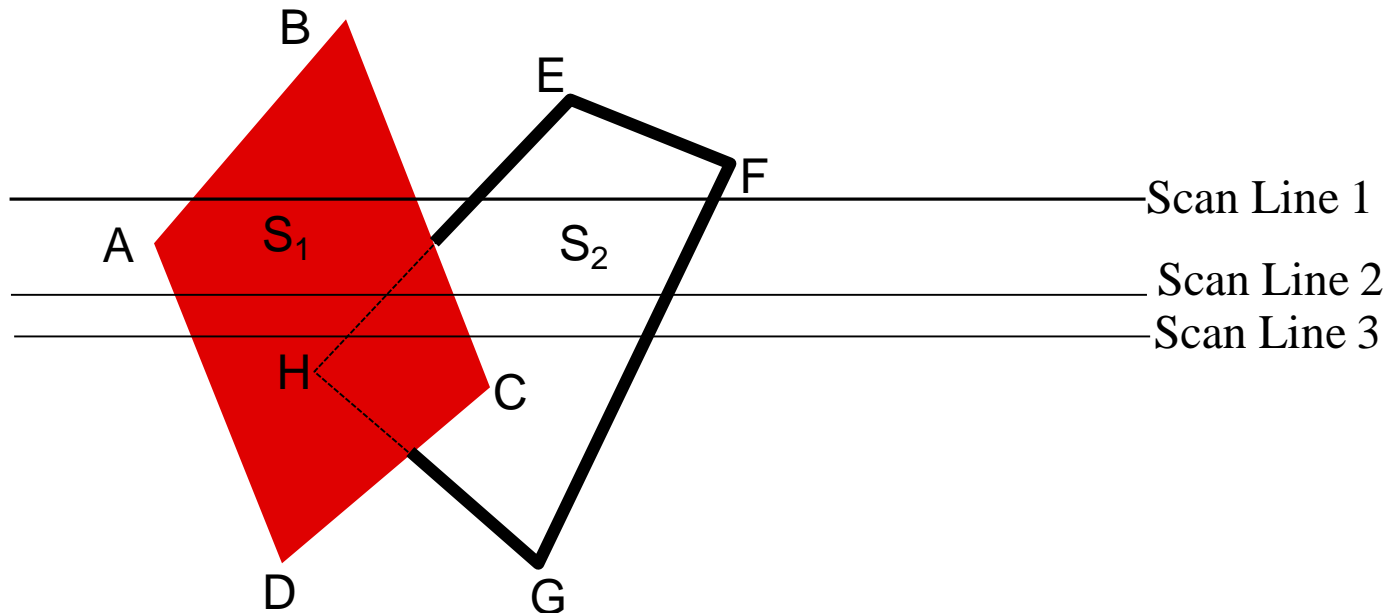


- Scan Line 1:

- (A,B) to (B,C) only inside S_1 , so color from S_1
- (E,H) to (F,G) only inside S_2 , so color from S_2

- Scan Line 2:

- (A,D) to (E,H) only inside S_1 , so color from S_1
- (E,H) to (B,C) inside S_1 and S_2 , so compute & test depth
In this example we color from S_1
- (B,C) to (F,G) only inside S_2 , so color from S_2



- Scanning takes place row by row
- To facilitate the search for surfaces crossing a given scan-line an active list of edges is formed for each scan-line as it is processed
- The active list stores only those edges that cross the scan-line in order of increasing x
- Pixel positions across each scan-line are processed from left to right
- We only need to perform depth calculations
- In Scan Line, Z-Buffer(X) whereas earlier it was $X*Y$



Use Z-Buffer for only 1 scan line / 1 row of pixel



- During scan conversion in the Active Edge List(AEL) \Rightarrow Calculate $Z(X,Y)$

i.e, -Pixel info between 1 & 2 active edges will only be stored

-Next pixel will be stored as $z=z_1+\Delta z$

- Δz will be constant but can change anywhere in case of slope

- If $Z(X,Y) > Z \text{ BUFFER}(X) \Rightarrow$ Update
- If 2 polygons are present-along with Active Edge List, Active Polygon List will also be included
- Active Polygon List \Rightarrow List of polygons intersecting a scan line



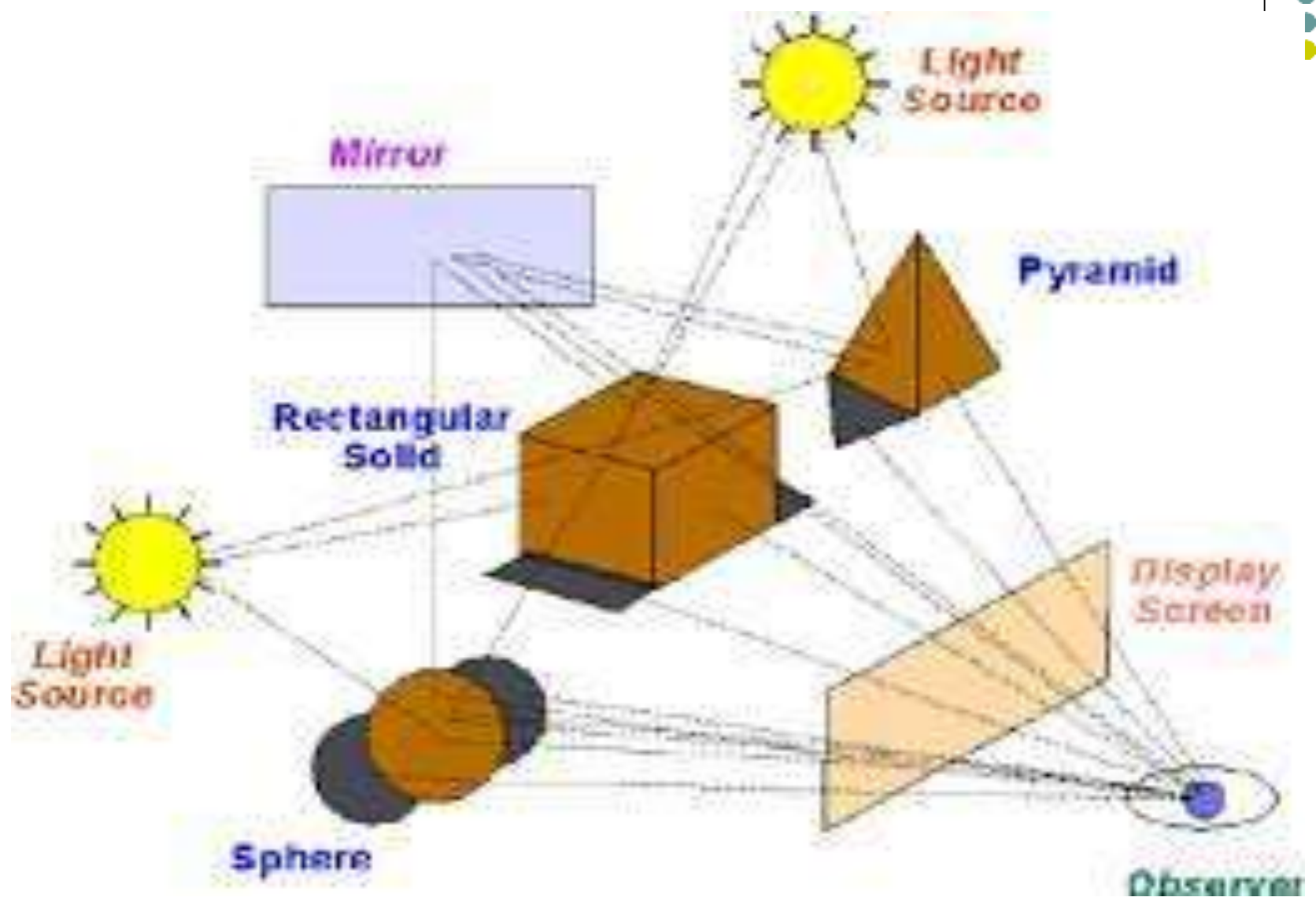
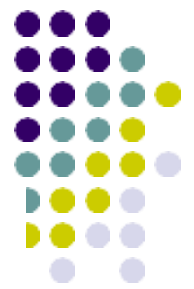
Hidden solid removal

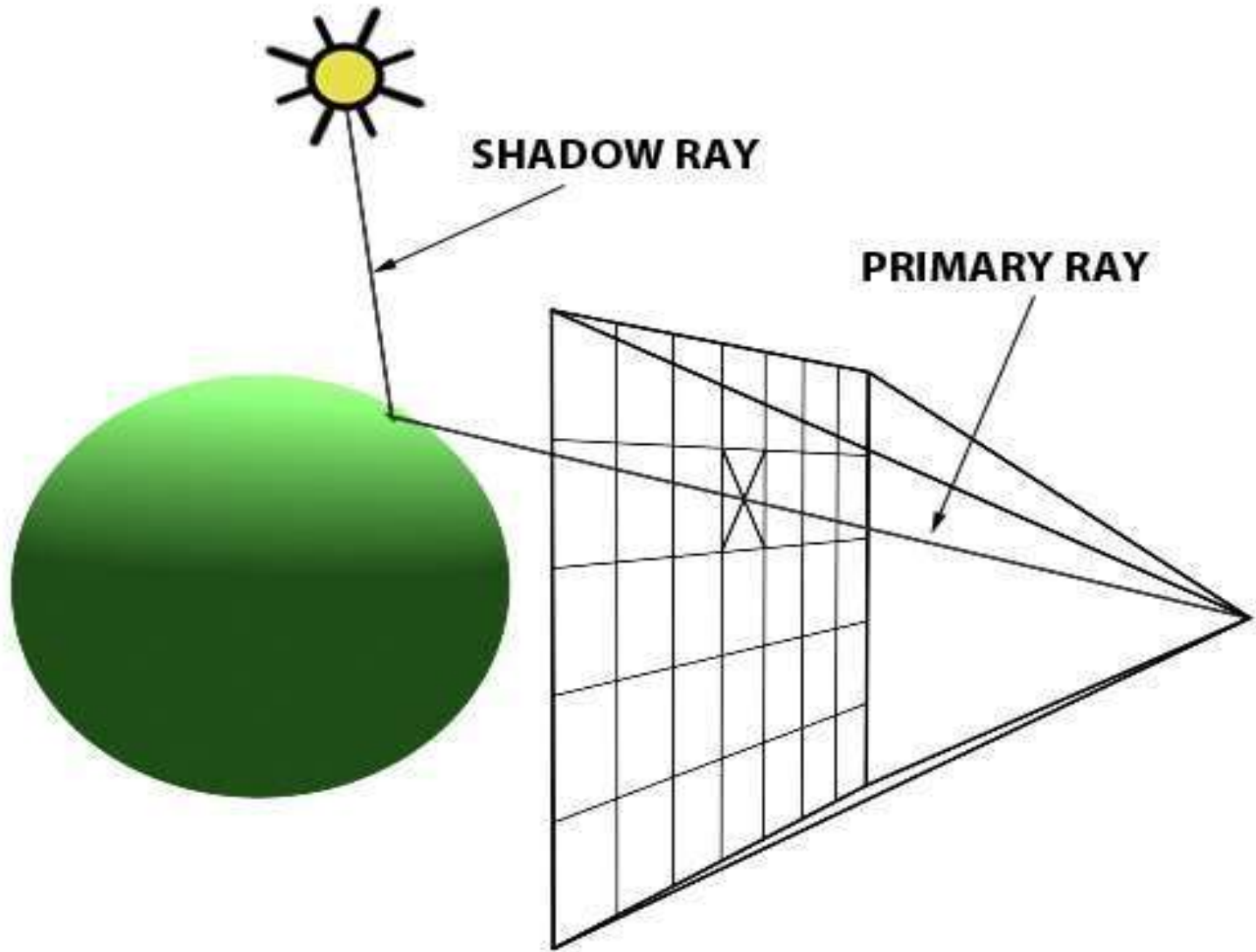
- The hidden solid removal of B-rep model are algorithms such as z-buffer.
- Convert CSG to B-rep.
- Render it with standard hidden surface removal techniques.
- **RAY TRACING**
- The complex 3D solid/solid intersection problem is converted into a 1D ray/solid intersection calculation

Ray tracing



- If we shoot a ray from the viewpoint through the pixel, the first object which hits is the one that is visible at the pixel.
- It can be used for both flat and curved surface.
- Shoot the ray from the eyepiece one per pixel
- Find the closest object blocking the path of the ray.
- Since it has infinite rays, the light rays are traced backwards, a ray from the viewpoint is traced through a pixel until it reaches a surface.







Ray casting

- If resolution is x,y then there are xy pixels so xy light rays are traced.
- Each ray is tested for intersections with each object in the picture including the non clipping plane.
- The intersection closest to the viewpoint is determined since rays intersect many objects.




- The main advantage is that it can create extremely realistic rendering of pictures by incorporating laws of optics for reflection and transmitting light rays
- The major disadvantage is the performance since it starts the process a new and treat each eye ray separately.

UNIT IV


ASSEMBLY OF PARTS




- Assembly Modelling
 - Interferences of Positions And Orientation
 - Tolerance Analysis
 - Mass property Calculations
 - Mechanism Simulation
 - Interference Checking
- 

Assembly Modeling


Assembly modelling is a technology and method used by computer-aided design and product visualization computer software systems to handle multiple files that represent components within a product

- Constructing an assembly begins with bringing in a base component, selected because of its central role. The base component is fixed.
 - •Each component brought in needs to be oriented and located relative to the base or other components in the assembly.
 - •Geometric relations (constraints) are used between components to position them properly for analysis and presentation
- 


Assembly Modeling Approaches

- Bottom – Up approach
 - Top – Down approach
- 

Bottom-Up approach

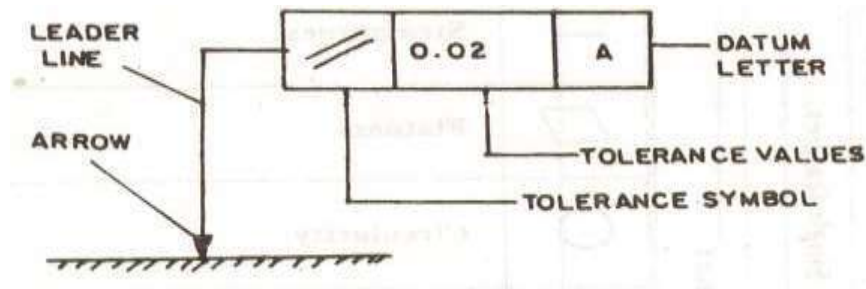
- **Bottom-Up approach – this is a logical, traditional, and most common approach. The individual parts are created independently, inserted into the assembly, and located and oriented (using the mating conditions) as required by the design. The first part inserted is known as the base and is fixed.**
 - Allows the designer to use part drawings that already exist (off the shelf).
 - Multiple copies (instances) of parts can be inserted into the assembly.
 - Any changes in the original part is reflected on all instances in the assembly
 - Provides the designer with more control over individual parts
- 


Top-Down approach

- **Top-Down approach – In this approach, the assembly file is created first with an assembly layout sketch. The parts are made in the assembly file or the concept drawing of the parts are inserted and finalized in the assembly file. In other words, the final geometry of the parts have not been defined before bringing them into the assembly file.**
 - The approach is ideal for large assemblies consisting of thousands of parts.
 - The approach is used to deal with large designs including multiple design teams.
 - It lends itself well to the conceptual design phase.
- 

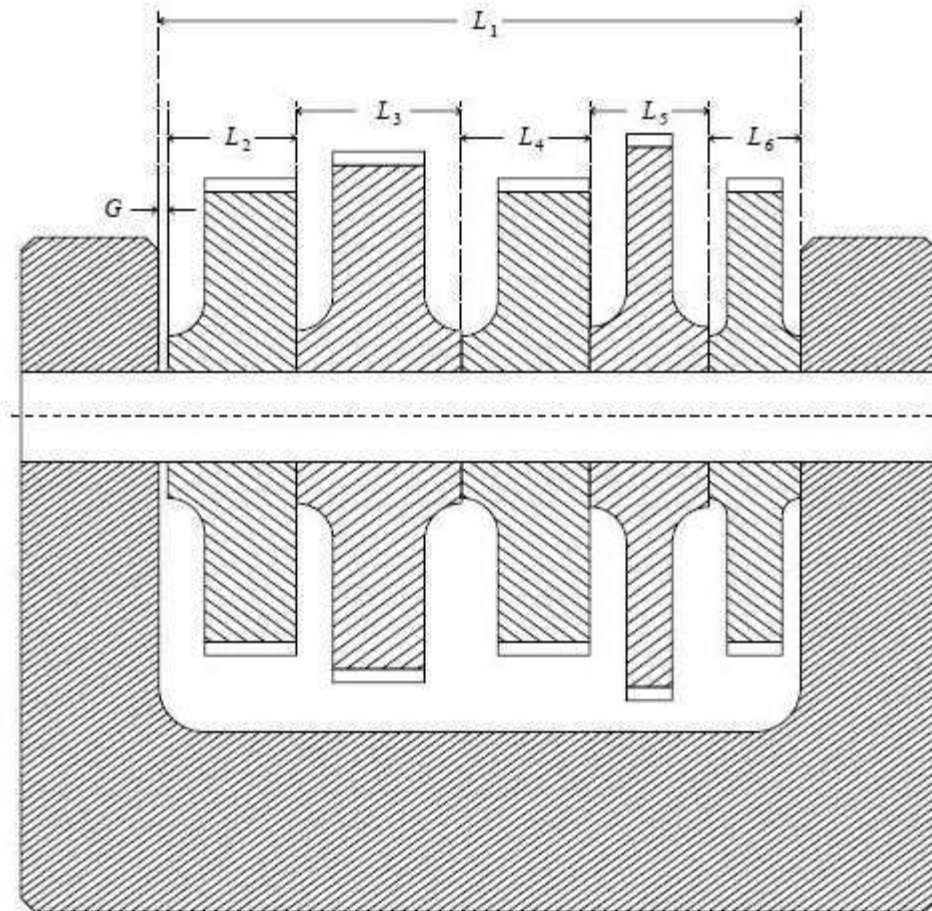
Tolerance

A tolerance is the total permissible variation from the specified basic size of the part



- Cost generally increases with smaller tolerance
 - Small tolerances cause an exponential increase in cost
 - Parts with small tolerances often require special methods of manufacturing.
 - Parts with small tolerances often require greater inspection and call for the rejection of parts
 - Greater Quality Inspection leads Greater cost.
 - Do not specify a smaller tolerance than is necessary
- 

Tolerance stack up analysis

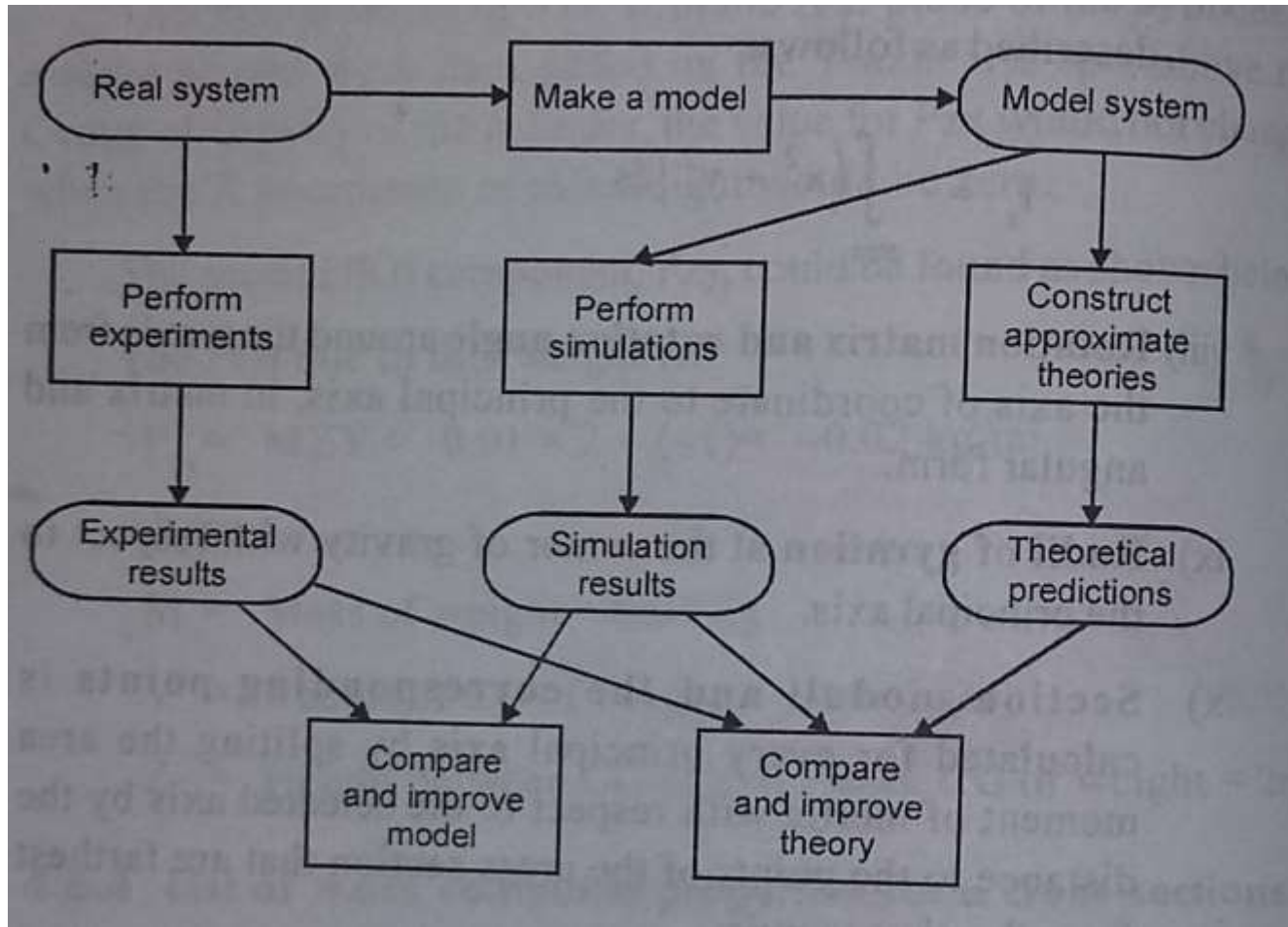


$$\begin{aligned} G &= L_1 - (L_2 + L_3 + L_4 + L_5 + L_6) \\ &= L_1 - L_2 - L_3 - L_4 - L_5 - L_6 \end{aligned}$$

IF $G \leq 0$, There is no tolerance

IF $G > 0$, Tolerance exist


Simulation mechanism



Virtual Simulation Input Hardware

- Body tracking
 - Physical controllers
 - Voice/Sound recognition
- 

Applications of Simulation

- Design for complex aircraft system
 - Aircraft simulation to train pilots
 - Simulators for the design of robots and its algorithms
 - Testing of safety mechanisms in new vehicle models.
- 

THANK YOU

UNIT – V

CAD Standards

Need for Graphics Standards

- The real issue with choosing the standards is portability and device independence.
- Complex CAD/CAM systems.
- Shape, Non shape, design and Manufacturing data.
- Need to integrate and automate design and manufacturing process to obtain maximum benefits from CAD/CAM.
- Direct Translators and neutral formats.

CAD Standards

- Graphics Standards
 - GKS,PHIGS,NAPLPS,GKS 3D, IGES
- Standards for Exchange images
 - Open GL
- Data Exchange Standards
 - IGES,STEP,DXF,STL,CALS,PDES,VRML,CGM
- Communication Standards
 - LAN,WAN,CGI,VDI,

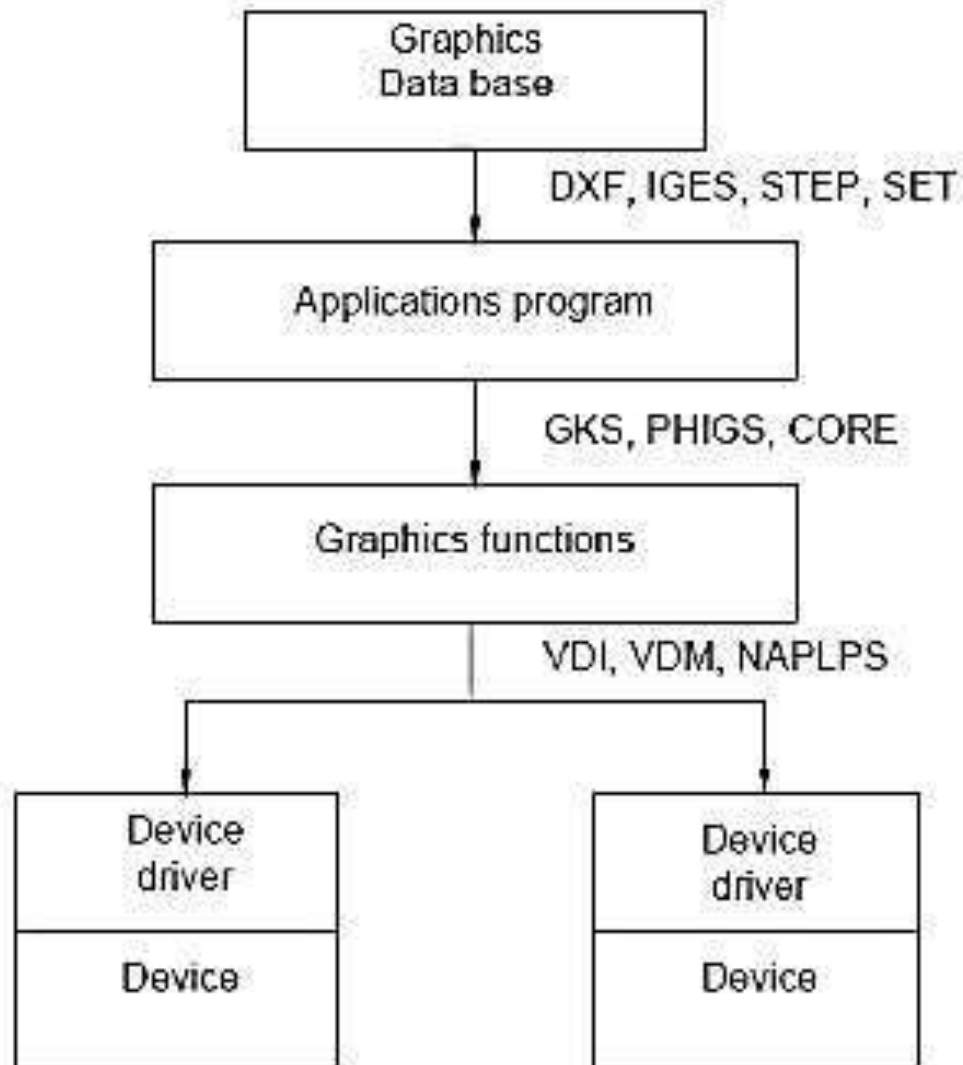
Various Interface Standards Exchange Format

- GKS (Graphical Kernel System)
- PHIGS (Programmer's Hierarchical Interface for Graphics)
- GKS -3D
- IGES (Initial Graphics Exchange Specification)
- DXF (Drawing eXchange Format)
- STEP (STandard for the Exchange of Product Model Data)
- CALS (Continuous Acquisition and Life cycle Support)
- ACIS (*.sat) [Alan, Charles, Ian's System]
- OpenGL – Open Graphics Laboratory
- DMIS (Dimensional Measurement Interface Specification)
- VDI (Virtual Device Interface)
- CGI (Computer Graphics Interface)
- VDM (Virtual Device Metafile)
- CGM (Computer Graphics Metafile)
- GKSM (GKS Metafile)
- PDES (Product Data Exchange Standards)
- VRML (Virtual Reality Modelling Language)

CAD Function

- Graphics Output Primitives
 - Line, polygon, sphere, ...
- Attributes
 - Color, line width, texture, ...
- Geometric transformations
 - Modeling, Viewing
- Shading and illumination
- Input functions

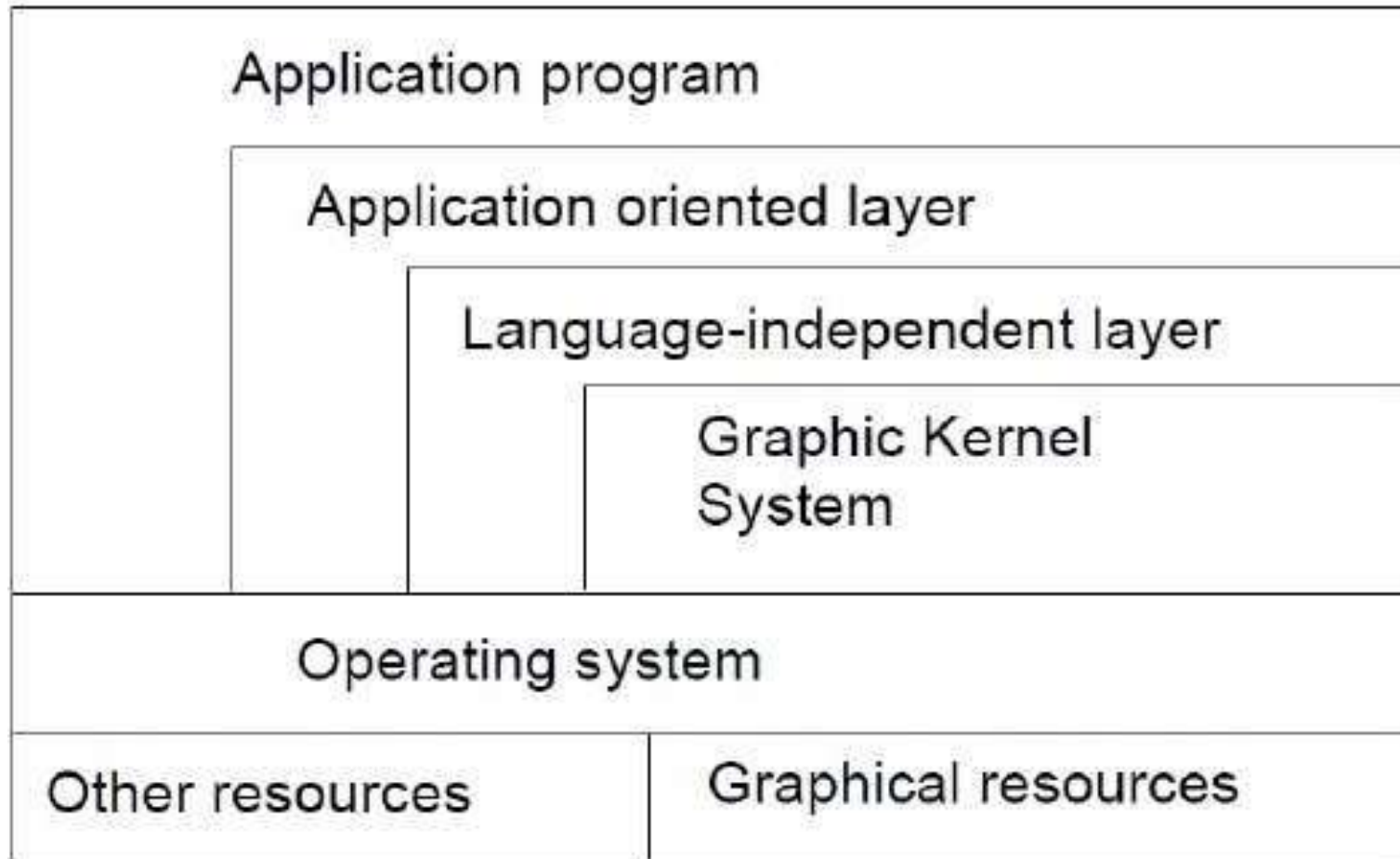
Graphics Standards



Graphical Kernel System (GKS)

- The main objectives that were put forward for GKS are :
 - To provide the complete range of graphical facilities in 2D, including the interactive capabilities.
 - To control all types of graphical devices such as plotter and display devices in consistent manner.
 - To be small enough for variety of programs.

Layer Model of GKS

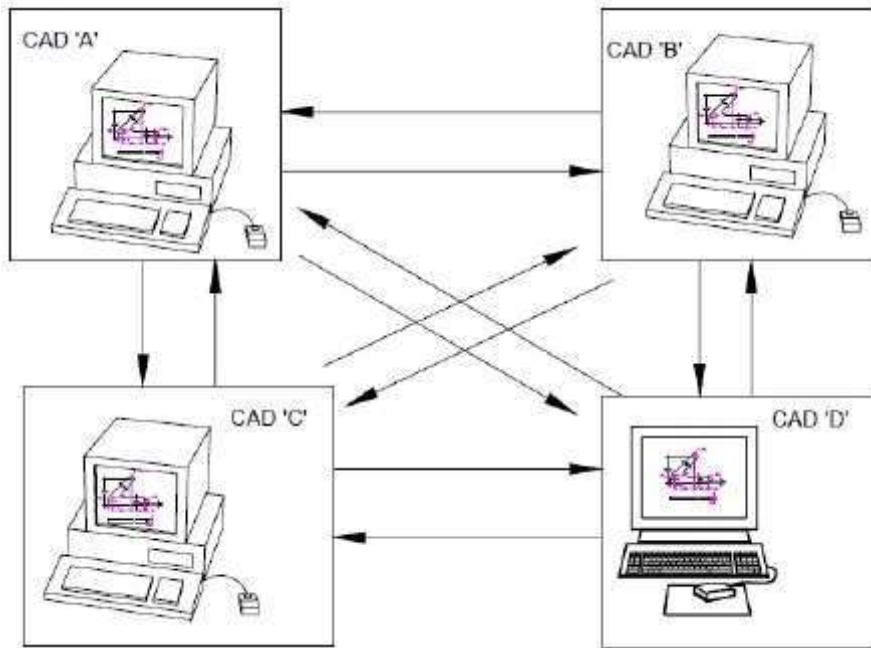


Types of Translators

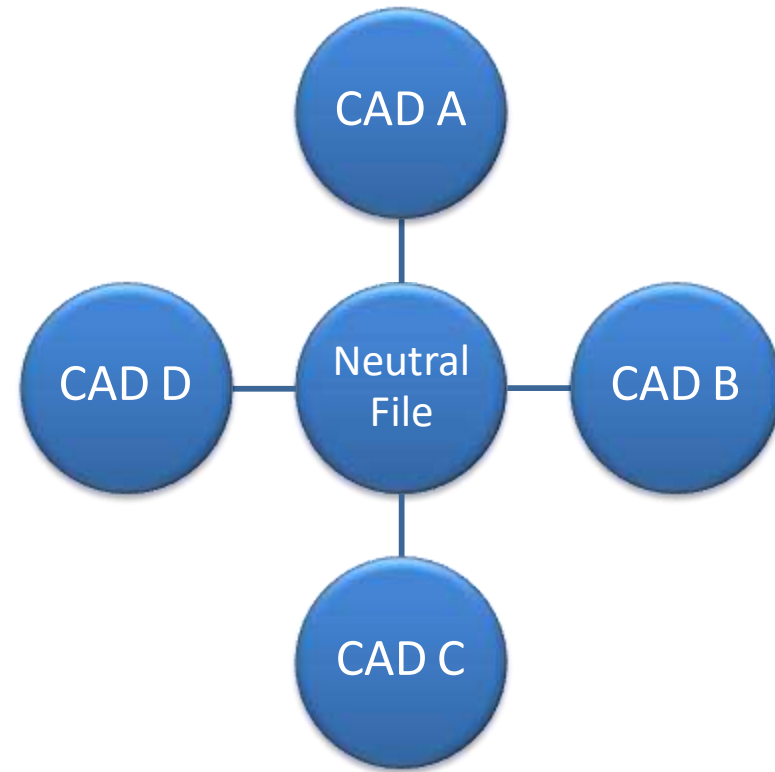
- **Direct Translators.**
 - Translating the CAD data directly from one CAD/CAM system to another, in a single step.
- **Indirect Translators.**
 - Translate the CAD data from one native format to another neutral file which is independent of any existing CAD/CAM system, that all systems can interpret and understand.

Types of Translators

Direct Translator



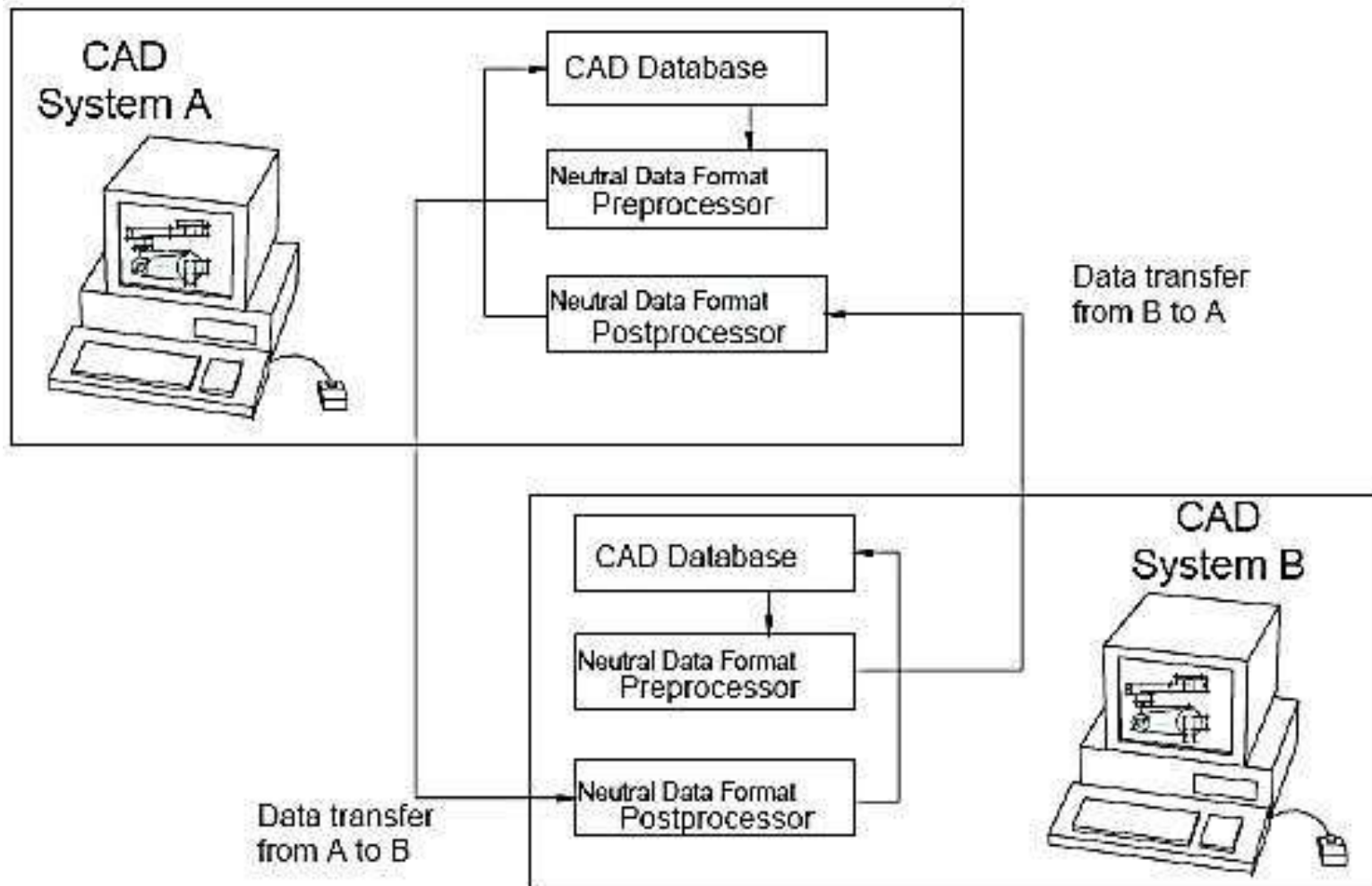
Indirect Translator



Initial Graphics Exchange Specification (IGES)

- IGES is the most comprehensive standard and is designed to transmit the entire product definition including that of manufacturing and any other associated information.
- IGES has three data types.
 - Geometric (Product Shape, Curve, Surface & Solid)
 - Annotations (Dimensions , Notes, Labels, GD&T)
 - Structure (Views, Visuals, Properties , Macros)

IGES Data interchange Format



IGES - File Structure

Flag Section

- Optional
- ASCII/Binary/Compressed ASCII

Start Section

- Man Readable Prologue
- Initialize IGES file

Global Section

- Details of Product
- Drafting Standards Used

Directory Section

- Index for File
- Color, Line Type etc

Parameter Entry Section

- Geometry, Annotation & Structure

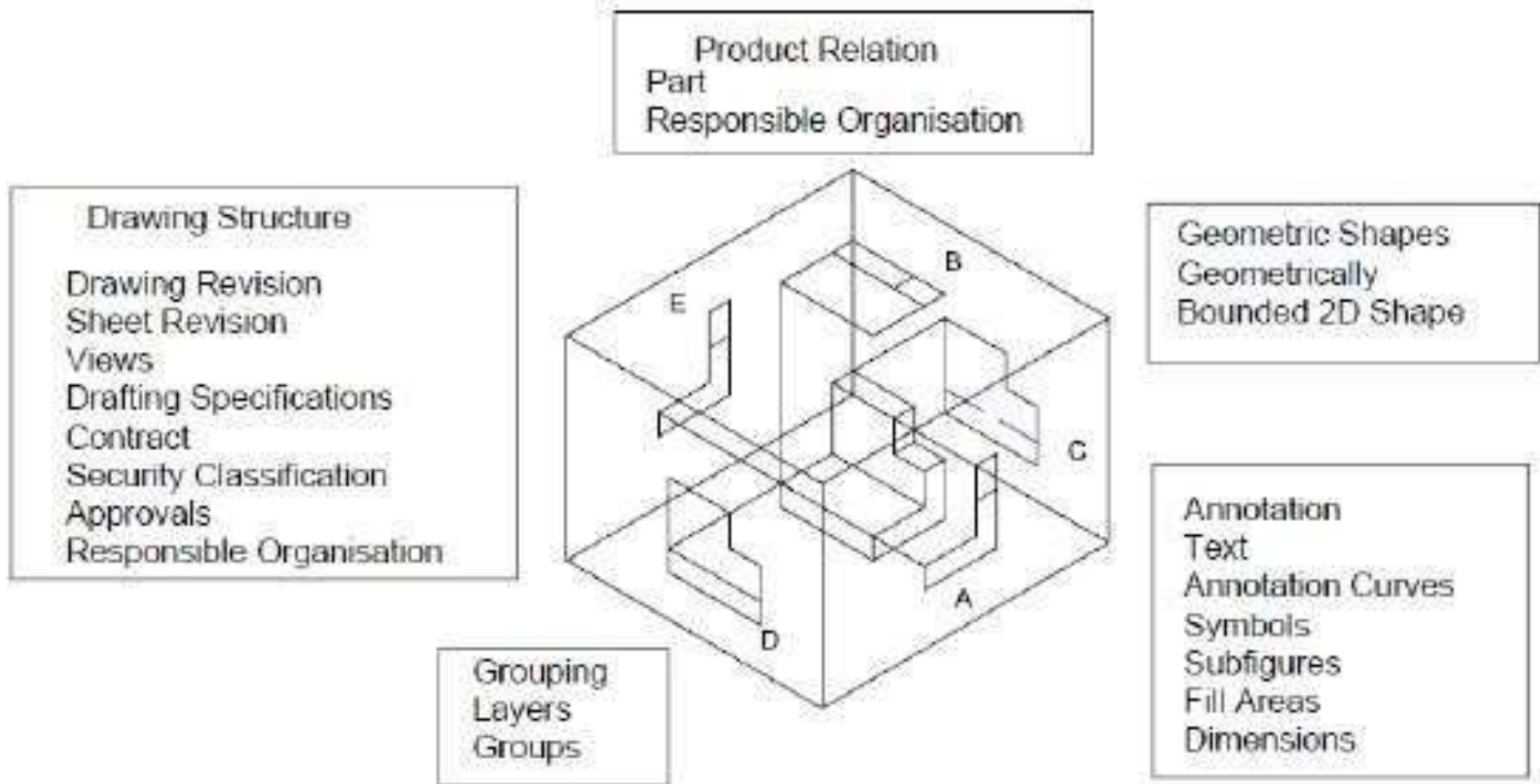
Terminate Section

- Subtotals of Record in every sections

Standards for the Exchange of Product model data (STEP)

- The standard method of representing the information necessary to completely define a product throughout its entire life.
- i.e from the conceptual figure to the end of the product in useful life.
- Standard methods for exchanging the data electronically between two different systems.

Standards for the Exchange of Product model data (STEP)



STEP Example – Sheet metal Die Planning & Design

Shape Definition
Representation
Advanced BREP Solids
Elementary BREP Solids
Facetted BREP Solids
Manifold Surfaces
with Topology
Surfaces and Wireframe
without Topology
Constructive Solid
Geometry
Part Shape Relationship
to Die Shape
Physical Model
Identification
Tolerances

AP 207: Sheet Metal
Die Planning
and Design

Items
Parts, Dies
or Materials
Versions
Classifications
Standard or designed

Part Process Plans
Template
Operations
Constraints

Work
Internal or External
Start or Change

Drawing Exchange Format (DXF)

- DXF format has been developed and supported by Autodesk for use with the AutoCAD drawing files.
- Depending on the software creating the DXF file, it can either be in an ASCII or a binary format.

Dimensional Measurement Interface Specification (DMIS)

- A new standard communication being established by CAM-I for Manufacturing.
- Used to identify and develop industrial automation standards.

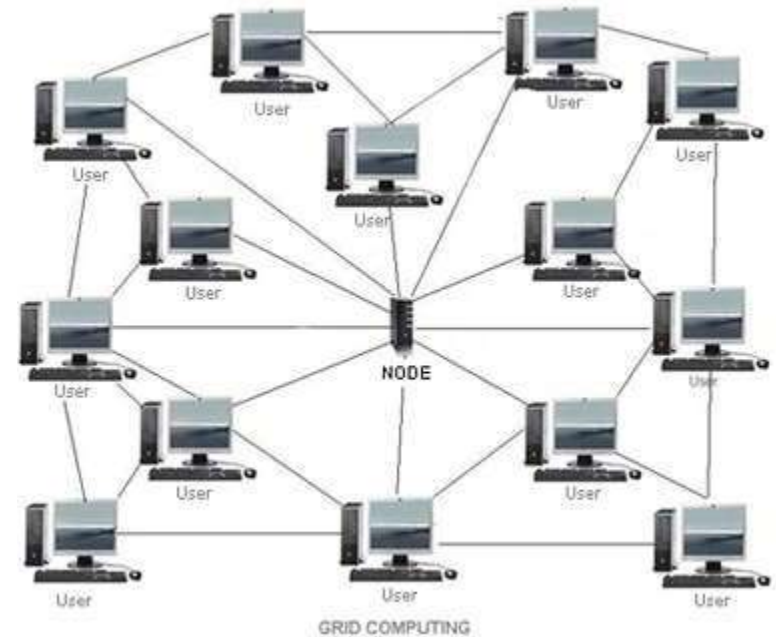
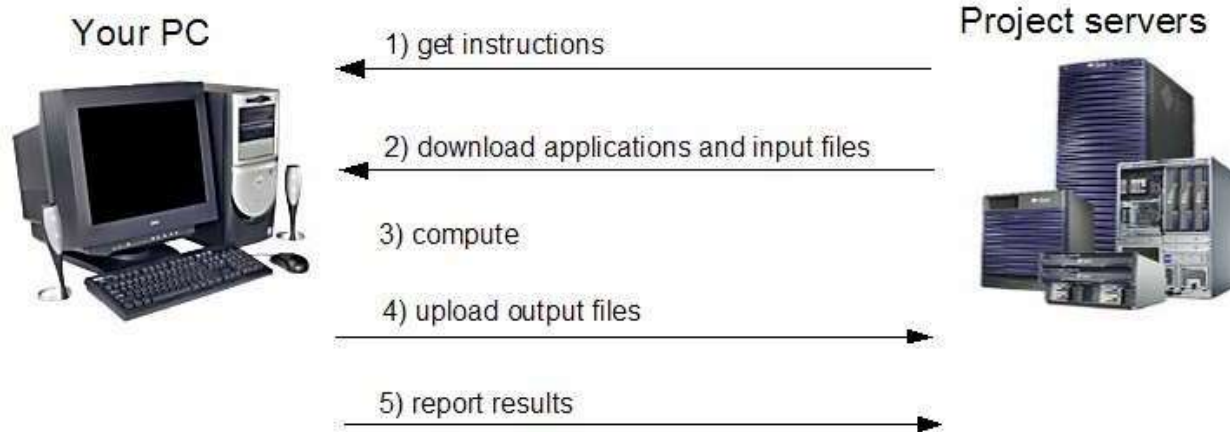
Communication Standards

- **Collaborative Design :**
 - It is an interactive process of real time communication between the members of a design team who are physically in different locations.
 - Exchanging documents between designers on time.

Concepts of Collaborative Design

- **Distributed Computing**
- **Intranets**
- **Internet**
- **Virtual Reality Modelling Language (VRML)**

Distributed Computing



Virtual Reality Modelling Language (VRML)

- VRML is a standard file format for representing 3-dimensional (3D) interactive vector graphics, designed particularly with the World Wide Web.
- Using VRML, you can build a sequence of visual images into Web settings with which a user can interact by viewing, moving, rotating, and otherwise interacting with an apparently 3-D scene.
- To view a VRML file, you need a VRML viewer or browser, which can be a plug-in for a Web browser you already have .

Thank You