5 ANIMATIONS AND REALISM

5.1.COMPUTER ANIMATION - INTRODUCTION

What is Animation?

✓ Animation is the creation of moving pictures one frame at a time.

✓ It is an art of creating, moving images with the use of computer.

✓ It creates a continuous, time varying image.

✓ "**Sequence of visual changes**" is called animation.

Animation - Use:

✓ It is used in,

1. Entertainment (2D graphics, 3D graphics)
2. Advertising
3. Art
4. Propaganda on film or video
5. Multimedia presentation
6. Simulators, etc.

Animation-Criteria

✓ Computer animation can generate following criteria's. They are,

1. Camera parameters
   - object position
   - orientation
   - Focal length, etc.
2. Changing lighting effects.
3. Time variations
4. object size
5. Transparency
6. Surface texture, etc.

5.2 DESIGN OF ANIMATION SEQUENCES

✓ In general an animation sequence is designed with the following steps. They are,

1. Storyboard Layout
2. Object definitions
3. Key frame specifications
4. Generation of in-between frames

✓ **Frame-by-frame** animation, each frame of the scene is separately generated and stored.

✓ Later, the frames can be recorded on film or they can be consecutively displayed in "**real time playback**" mode.

1. Storyboard Layout

✓ It is an outline of the action

✓ It discusses the type of animation to be produced.

✓ It defines a set of basic events that are to take place in a specific order.

✓ Sketch a storyboard in pencil, of your ideas for each slide in your presentation.

✓ Each slide must give health message to the user/audience.

✓ Each story board should be visually and spatially easy to read.

2. Object Definition

✓ It is given for each participant in the action.

✓ Object can be defined in terms of,

- Basic shapes (ex: polygon, spline, etc.,)
- Colors, etc.,

3. Key frame specifications

✓ It is a detailed drawing of the scene at a certain time in the Animation sequence.

✓ Each object is positioned according to the time for the frame.

✓ More key frames are specified for slowly varying motions.

✓ It creates transition among its target values over its duration.

4. Generation of in-between frames

✓ In between frame means "Intermediate frame between the key frames".

eg: K, K + 1, K + 2 .....

Here, K, K + 2 - key frames

K + 1 is a intermediate (or) In between frame.

Note:

✓ Film requires 24 frames per second.

✓ Graphics terminal are refreshed at the rate of 30 to 60 frames per second.

Ex:1 A film requires 1440 frames per minute, Calculate the key frames?

- Total number of Key frames - 1440 frames
- In between frames five for each pair of key frames so ⇒ 1440/5 = 288 frames.
- So, we need 288 key frames.

Other tasks:

5. Motion Verification
6. Editing
7. Production
8. Synchronization of sound track, etc.,

5.3 GENERAL COMPUTER ANIMATION FUNCTIONS

Computer Animation can be created with a computer with the help of computer animation software.

✓ Some animation software are,

1. Flash (Adobe)
2. 3D Studio Max
3. Photoshop Cs 6
4. Mari 2.0v2
5. Unity pro 4, etc.,

✓ Computer Animation software provideds following functions. They are

1. Store the object in database.
2. Manipulate the object
3. Motion generation
4. Object rendering
5. Mange the object database
6. Object shapes and associated parameters are stored & Updated in the database.

✓ Other special functions provide by animation softwares are,

7. Simulate camera movements
8. Perform Zooming
9. Panning
10. Tilting actions
11. Generate in-between frames when key frames are specified, etc.,

5.4 RASTER ANIMATIONS

✓ **"Raster animation"** helps to generate real time animations

✓ It helps to execute sequence of animation (Displaying the Sequences).

Displaying the sequences

✓ Movies work by fooling our eyes.

✓ A sequence of static images presented in a quick successive appears as continuous flow.

✓ The eye cannot register images faster than animation software.

✓ If a gap in the projection occurs, the eye seems to perform spatial interpolation over the gap

✓ To achieve smooth animation a sequence of images (frames) have to be presented an a screen with the speed of at least 30 frames per second.

✓ Animation frames can be,

1) precomputed in advance & preloaded in memory.
2) Computed in real time. etc.

Raster animation

- ✓ Raster animation is the most common animation technique.
- ✓ Frames are copied very fast from off screen memory to the frame buffer.
- ✓ Copying usually done with bit BLT type operations
- ✓ Copying can be applied to,
  1) Complete frames
  2) Only parts of the frames

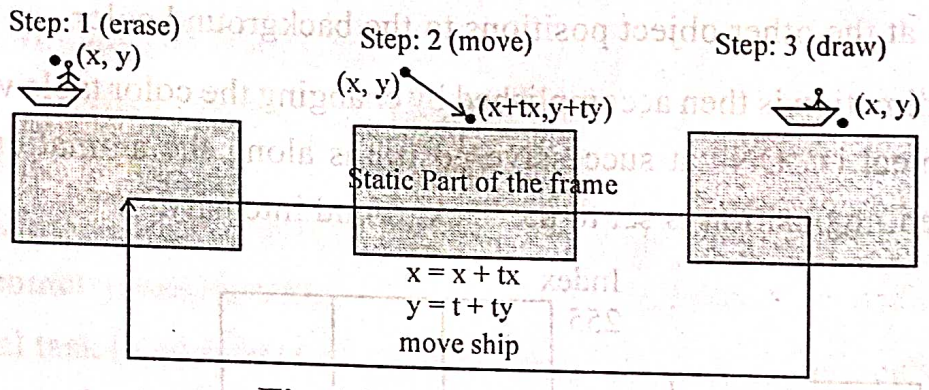

Fig.5.1 Raster Animation

Raster animation - procedures

- ✓ A part of the frame in the frame buffer needs to be erased.
- ✓ The static part of the frame is re-projected as a whole & the animated part is over projected.

1. Double buffering
2. Color table animated

Double buffering

- ✓ It is used to achieve smooth animation.
- ✓ The next frame of animation is computed to an off-screen buffer at the same time when the current frame is transferred to the frame buffer.

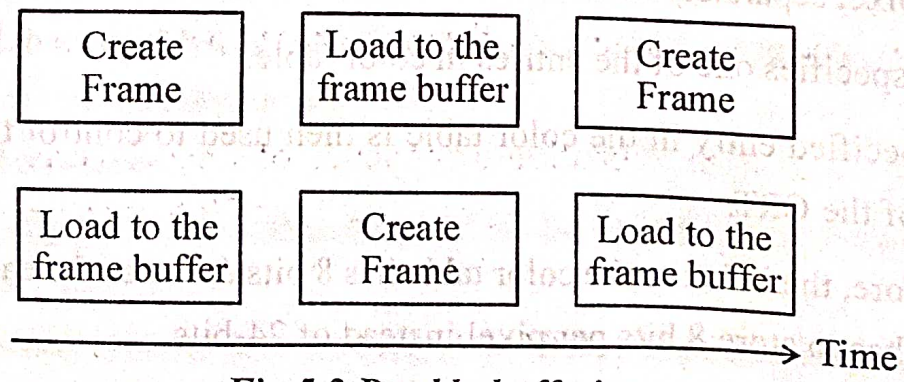

Fig.5.2 Double buffering

Color - table Animations

- ✓ Simple 2D animations can be easily implemented using Color Look Up Table (CLUT) in RGB format.
- ✓ Lookup table also called "color table".
- ✓ Set up successive blocks of pixel values to color table entries.
- ✓ We set the pixels at the first position of the object to "ON" values & we set the pixels at the other object positions to the background color.
- ✓ The animation is then accomplished by changing the color table values, so that the object is "ON" at successive positions along the animation path as the proceeding position is set to the background intensity.

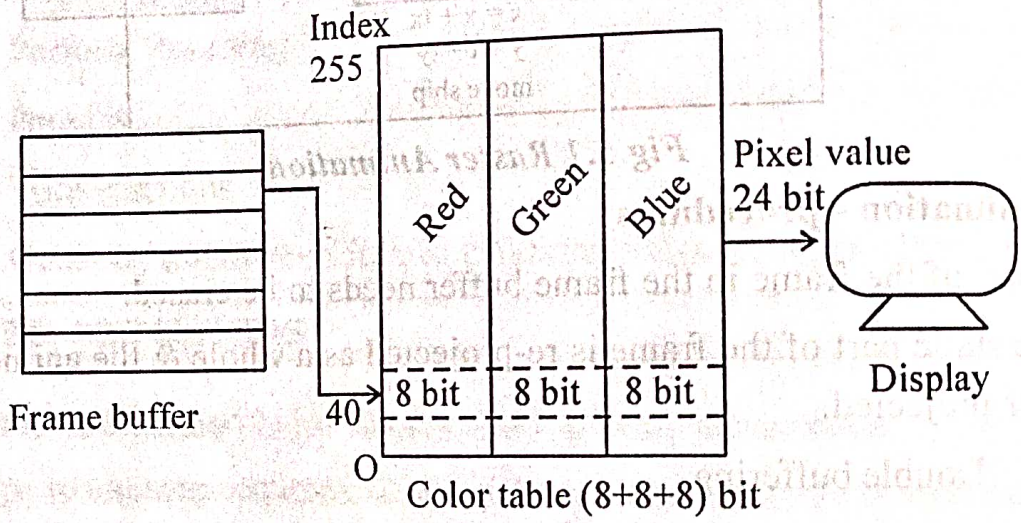

Fig.5.3 Video - color table

Here,

- ✓ Color table contains 256 entries/index.
- ✓ Each pixel separately maintain color table index from the frame buffer.
- ✓ Index specifies one of the entries in color table.
- ✓ The specified entry in the color table is then used to control the intensity or color of the CRT.
- ✓ Therefore, the index to the color table has 8 bits & hence for each pixel frame buffer has to store 8 bits per pixel instead of 24-bits.

5.5 COMPUTER ANIMATION LANGUAGES

- ✓ C, LISP, PASCAL, FORTRAN - These are the general purpose animation languages used in animation functions,
- ✓ Animation functions includes,
  1. Graphics editor
  2. Key frame Generator
  3. In-between frame generator
  4. Standard graphics routines.

Graphics editor

- ✓ It is used to design & modify object shapes using spline surfaces constructive solid geometry surfaces, etc.
- ✓ A typical task in an animation sequence is,

1. Scene description

- A typical task in an animation specification
- It includes positioning of objects, Light sources and camera parameters etc.,

2. Action Specification

- It involves the layout of motion paths for the object & camera.

Key frame systems

- ✓ Each animation languages designed & developed limited number of degree freedom to the 3D object translation.

  Eg: As an example, the single arm robot has 6 degrees of freedom, which are called sweep, shoulder, swivel, elbow extension, pitch, yaw & roll.
- ✓ We can also extend 6 degrees of freedom into 9 or 12 degrees of freedom by allowing 3D translation for the base.

Parameterized Systems

- ✓ It allows object-motion characteristics to be specified as part of the object definitions.
- ✓ The adjustable parameters control, such object characteristics as degree of freedom, motion limitations and allowable shape changes.

Scripting systems

- ✓ It allows object specifications & animation sequences to be defined with a user input script.
- ✓ From the script, a library of various objects and motions can be constructed.

5.6 KEY FRAME SYSTEMS

- ✓ Key frame is a frame where we define changes in animations.
- ✓ Every frame is a key frame when we create frame by frame animation.
- ✓ When some-one creates a 3D animation on a computer, they usually don't specify the exact position of any given object on every single frame. Because they create key frames.
- ✓ Key frames are important frames during which an object changes its size, direction, shape or other properties.
- ✓ The computer then figure out all the in-between frames & saves an extreme amount of time for the animator.
- ✓ The following illustration depict the frames drawn by user & the frames generated by computer.

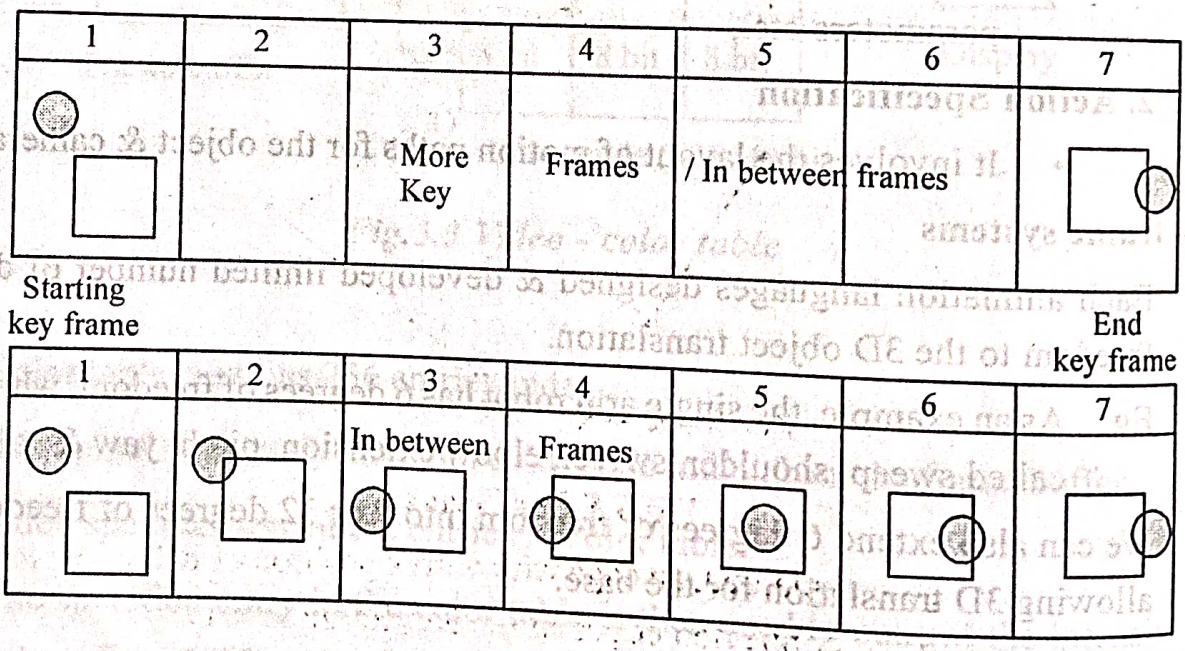

Fig.5.4 Key frame systems

- ✓ We generate each set of in-between from the specifications of two or more key frames.

- ✓ Motion paths can be given with a kinematic description as a set of spline curves, or the motion can be physically based by specifying the forces acting on the objects to be animated.
- ✓ For complex scenes, we can separate the frames into individual components or objects called "**Cels**" i.e "**Celluloid transparencies**".
- ✓ Then in-between frames of individual objects are generated using interpolation.

5.8 MOTION SPECIFICATIONS

✓ The motions of objects are specified by,

1. Direct motion specification
2. Goal - directed systems
3. Kinematics & Dynamics

5.8.1 Direct Motion Specification

✓ It helps to define a motion sequence.

✓ Direct motion specification parameters are,

1. Rotation angles
2. Translation vectors

✓ Here the geometric transformation matrices are applied to transform coordinate positions.

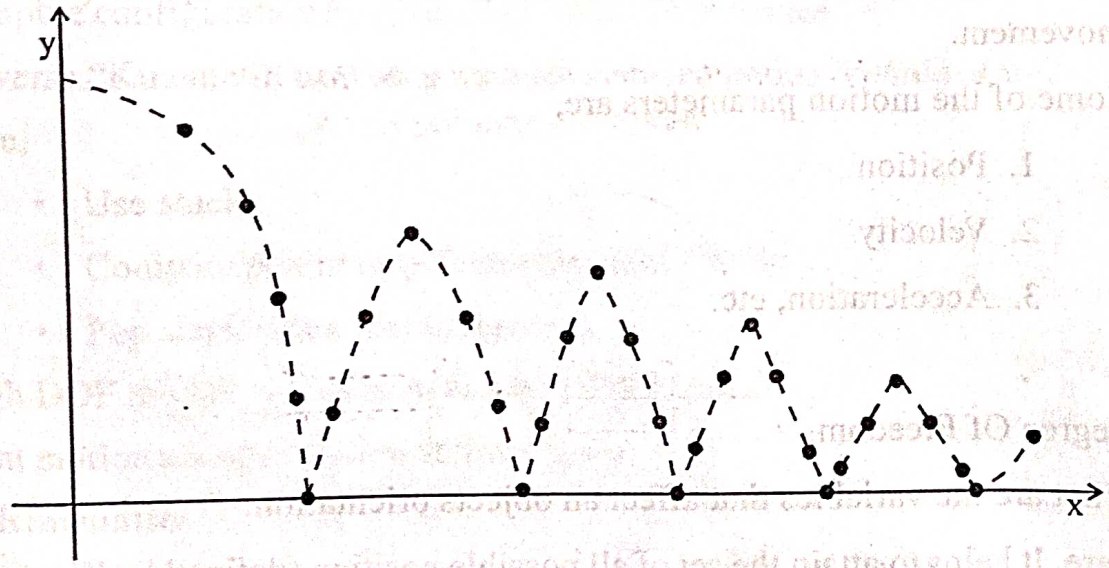

Fig.5.14 Approximating the motion of bouncing ball with a damped sin function

✓ Alternatively, we would use the **approximating equations** to specify certain kind of motions.

✓ Approximating the motion of bouncing ball with a damped sin function is,

$$y(x) = A|\sin(\omega_x + \theta_o)\, e^{-kx}$$

Here,

A → Initial Amplitude

ω → angular frequency

θ_o → phase angle

k → Damping constant

✓ This method used for simple user programmed animation sequences.

5.8.2 Goal-Directed System

✓ It specifies the motions that are to take place in general terms that abstractly describe the actions.

✓ It determined by the specific motion parameters given the goals of the animation.

ex: We would specify that we want an object to "walk" or to "run" to a particular destination.

5.8.3 Kinematics & Dynamics

1. Kinematics

✓ The study of object movements irrespective of their speed (or) style of movement.

✓ Some of the motion parameters are,

1. Position
2. Velocity
3. Acceleration, etc.

DOF

✓ Degree Of Freedom.

✓ What are the variables that affect an objects orientation.

✓ Here, It helps to attain the set of all possible position (defined by Kinematics) by an object.

eg:1 Robot arm work with six degree of freedom.

6 DOF → 2 base
1 shoulder
1 elbow
2 wrist

eg:2 Airplane flying with six DOF

6 DOF → x, y, z position roll, pitch, yaw

- ✓ If we also apply and specify accelerations (rate of change velocity), we can generate speed Up's slow down's & curved motion paths.
- ✓ Kinematic specification of a motion can be also given by simply descrypting of the motion paths.
- ✓ This is often done by spline curves.
- ✓ Kinematics modeling is based on Hierarchical parent - child spatial relationship.
- ✓ There are 2 methods of kinematics. They are
  1. Forward Kinematics
  2. Reverse Kinematics

Forward Kinematics

- ✓ Compute configuration by given individual DOF values.
- ✓ Traverse "Kinematic tree" & propagate transformation download.

  Here,
  - Use stack
  - Compare parent transformation with Childs.
  - Pop stack when leaf is reached.
- ✓ High DOF models are tedious to control this way.
- ✓ Joint motion are specified in spline curves.

Inverse Kinematics

- ✓ Compute individual DOF values that result in specified end effectors position.
- ✓ Given end effectors position, compute required joint angles.
- ✓ In simple case, analytic solution exists.

  eg: geometry, algebra etc.

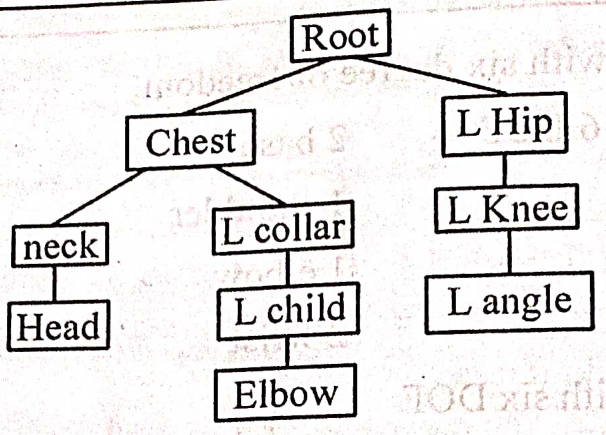

Fig.5.15 Kinematic tree

2. Dynamics

- ✓ It is a descriptions of object behaviour under the influence of forces are generally referred to as a **"physically based modeling"**.
- ✓ Object motions are obtained from physical equations.

  eg: Gravitational
  Friction
  Mechanical forces
  Electromagnetic, etc.
- ✓ It can be described by physical laws.

  eg: Newton's Law (Gravitational /Friction)
  Euler Law (Fluid flow)
  Navier stokes (Fluid flow)
  Maxwell equations(electromagnetic forces)

Newton's Law:

$$\boxed{F = \frac{d(mv)}{dt}}$$

Here,

m - mass (constant)

v - velocity vector

F - Force vector

ie, F = ma

Here, a → acceleration vector

3.23 VISIBLE SURFACE DETECTION METHODS

- ✓ It is also called **"Hidden surface elimination methods"**.
- ✓ When we view a picture containing non – transparent objects and surfaces, then we cannot see those objects from view which is behind from objects closer to eye.
- ✓ We must remove this hidden surface to get a realistic screen image.
- ✓ The identification and removal of these surface is called **"Hidden surface problem"**.

There are 2 approaches for removing hidden surface problems.

1. Object – space method,
2. Image –space method

Table 4.10 Object space Vs Image space method

Object space method	Image space method
• It is implemented in physical coordinate system.	• It is implemented in screen coordinate system.
• It deals with object definitions directly.	• It deals with their projected images.
• It compare parts of objects to each other to determine which surfaces should be labeled as visible.	• Visibility is decided point by point, each pixel position on the projection plane. And screen resolution can be a limitation.
• It decide which object is whole and visible.	• The visibility decide point by point.
• It is a continuous one.	• It is a discrete one.

• Efficient for small number of objects.	• Efficient for all the object.
• It is difficult to implement.	• Easy to implement.
eg. 1. Back face Detection 2. BSP tree method 3. Area subdivision method 4. Octree method 5. Ray casting method	eg: 1. Depth Buffer method 2. A buffer method 3. Scan line method 4. Area subdivision method

List of visible surface detection methods

1. Back face detection Algorithm.
2. Scan line method.
3. Depth buffer algorithm (or) Z buffer algorithm.
4. Depth sort algorithm (or) Painters algorithm.
5. A buffer algorithm.
6. BSP Tree method.
7. Octree method.
8. Area subdivision Algorithm (or) Warnock algorithm.
9. Ray casting method.

3.23.1 Back- Face Detection Algorithm

- A fast & simple object space method for identifying the back- faces of a polyhedron is based on the "**inside – outside**" tests.
- A point (x, y, z) is " inside" a polygon surface with plane parameters A, B, C & D, if when an inside point is the line of sight to the surface, the polygon must be a back face.
- i.e, We are inside that face and cannot see the front of it from our viewing position.
- We can simplify this test by considering the normal vector 'N' to a polygon surface, which has Cartesian components (A, B, C). In general, If "V" is a vector in the viewing direction from the eye (or 'camera') position, then this polygon is a back face if,

$$\boxed{V \cdot N > 0}$$

- Further more, if object descriptions are converted to projection coordinates and your viewing direction is parallel to the viewing Z – axis, then

$$V = (0,0,V_z)$$

and

$$V \cdot N = V_z \cdot C$$

- So that we only need to consider the sign of 'C' the component of the normal vector 'N'. In a right – handed viewing system with viewing direction along the negative Zv axis, the polygon is a back face if $C<0$, we cannot see any face whose normal has Z component $C = 0$, since your viewing direction is towards that polygon.

- Thus, in general, we label any polygon as a back face if its normal vector has a Z component value, $C<=0$.

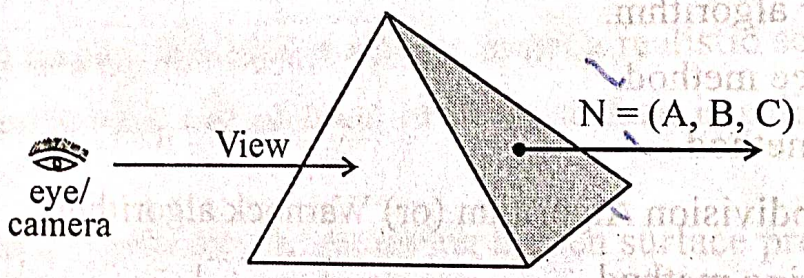

Fig.3.92

- Similar methods can be used in packages that employ a left – handed viewing system.
- In these packages, plane parameters A, B, C & D can be calculated from polygon vertex coordinates specified in a clockwise direction.
- Unlike the counter clockwise direction used in a right handed system.

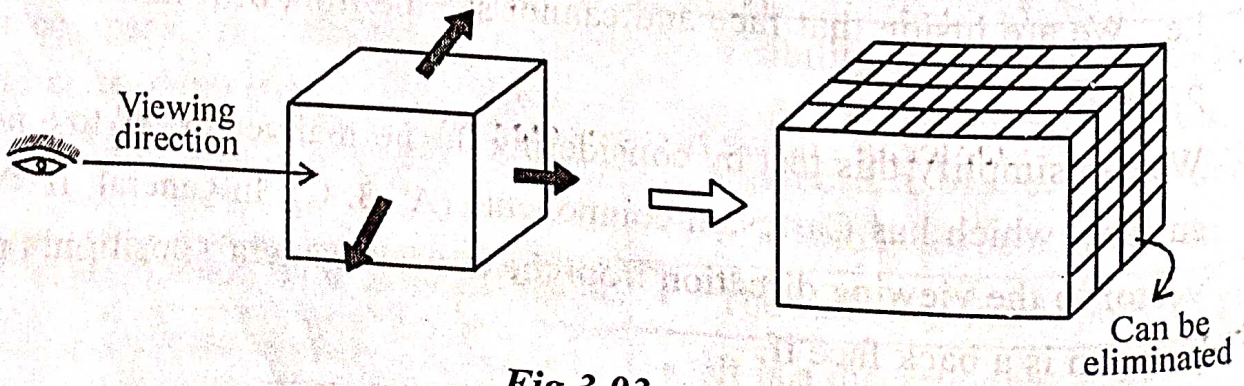

Fig.3.93

- Also, back faces have normal vectors that point away from the viewing position & identified by $C >= 0$ when viewing direction is along the positive z_r axis.
- By examining parameter 'C' for the different planes defining an object, we can immediately identify all the back faces.

3.23.2 Depth Sort Algorithm

- It is also called **"Painters algorithm"**.
- It is developed by **Newell & Sancha.**
- It using both image & object space operations.
- The depth – sorting method performs 2 basic function. (i.e,)

1.	The surface are sorted in order of decreasing path.
2.	The surface are scan – converted in order, stating with surface of greatest path.

- The scan conversion of the surface is performed in image space.
- This method for solving the hidden – surface problem in often referred to as **"Painters algorithm"**.
- The following figure shows the effect of depth sorting (eg : Oil painting).

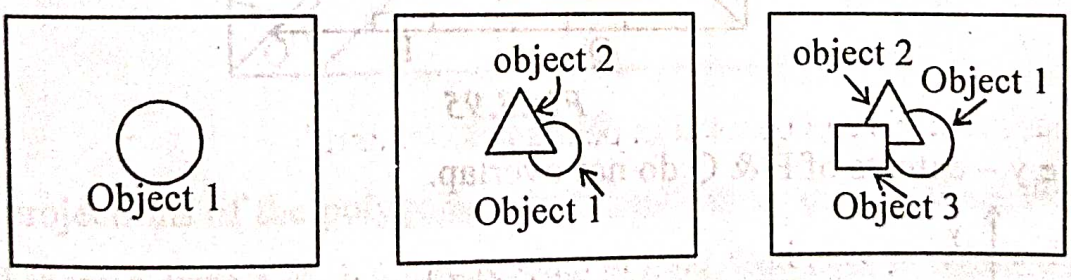

Fig.3.94 Painters algorithm

In the Oil painting,

- The new paint covers the old paint, so that only the newest layer of point is visible.
- Using the similar technique we first sort the polygon according to their distance from view point.
- The intensity values for the farthest polygon are then entered into the frame buffer.

- ✓ Taking each polygon is succeeding polygon in turn (in decreasing depth order) polygon intensities are painted on the frame buffer over the intensities of the previous processed polygons.
- ✓ This process is continued as long as no overlaps are occur.
- ✓ In depth overlap is detected by any point in the sorted list, we have to make some additional comparisons to determine whether any of the polygons should by recorded.
- ✓ We can check whether any polygon Q does not obscure (Hidden) polygon p by performing following steps.

1. The Z- extents of P & Q do not overlap.

i.e $\boxed{Z_{Qmax} < Z_{pmin}}$

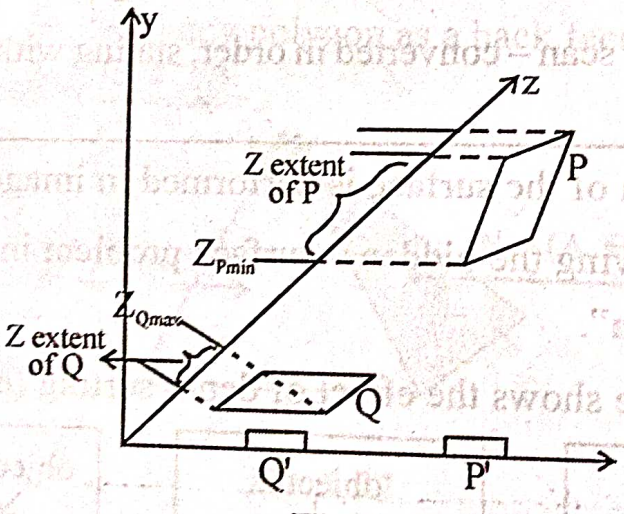

Fig.3.95

2. The y – extents of P & Q do not overlap.

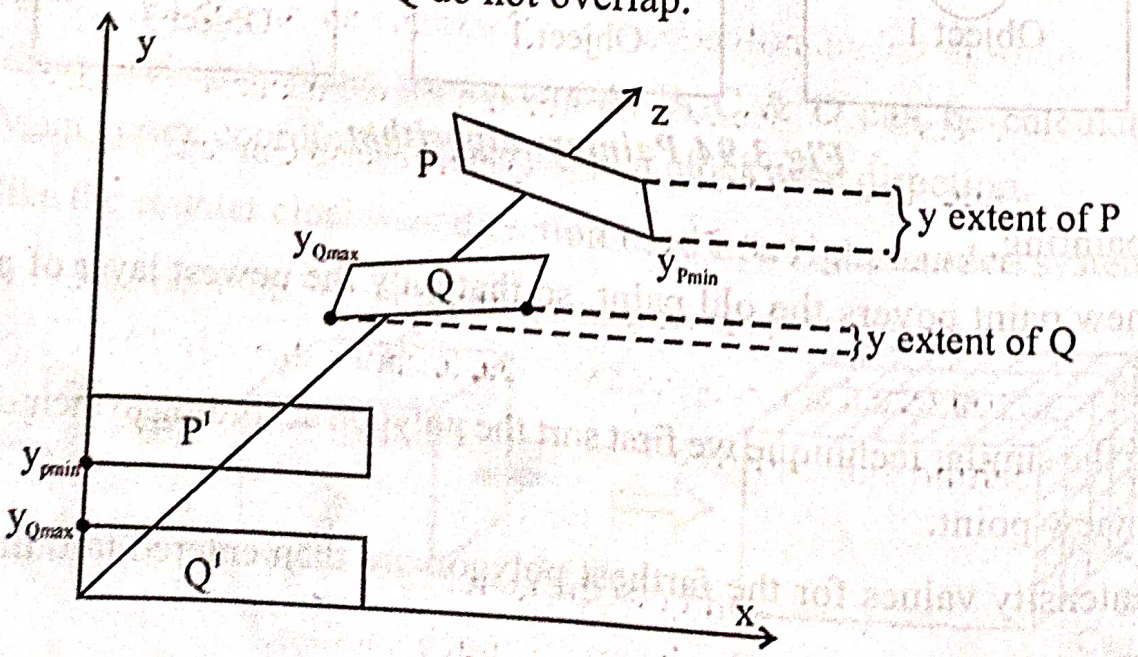

Fig.3.96

3. The extents of P & Q do not overlap.
4. Polygon P lying entirely on the opposite side of Q's plane from view port.

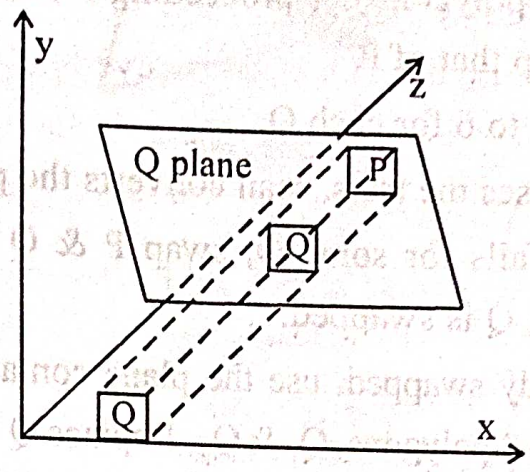

Fig.3.97

5. Polygon Q lying entirely on the same side of P's plane as the view port.

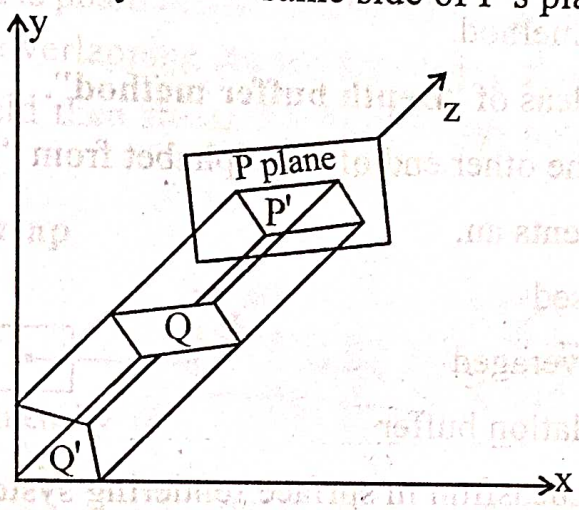

Fig.3.98

6. The projections of the polygons P & Q onto the xy screen donot overlap.

✓ It all these 6 tests fall, we assume for the moment that P actually obscures (Hidden) Q and therefore test whether Q can be scan converted before P. Here, we have to repeat test 4 & 5 for Q.

✓ If these test's also fail when we can say that there is no order in which P & Q can be scan converted correctly and we have to split either P or Q into 2 polygons.

✓ The idea behind the splitting that spitted polygons may not obscure other polygons.

✓ Theoretically, this partitioning could generate $O(n^2)$ individual polygons, but in practice, the number of polygons is much smaller,

Painters Algorithm

1. Sort all polygons in order of decreasing depth.
2. Determine all polygons Q (proceeding P) in the polygon list whose Z extents overlap that of P.
3. Perform test 2 to 6 for each Q.
4. If every Q passes the tests, scan converts the polygon P.
5. If every test fails for some Q, swap P & Q in the list and make the indication that Q is swapped.
6. If Q has already swapped, use the plane containing polygon P to divide polygon Q into 2 polygons, Q_1 & Q_2. Replace Q with Q_1 & Q_2 repeat step 3.

3.23.3 A – Buffer Method

✓ It is a image space method.

✓ It is an extension ideas of "**Depth buffer method**".

✓ "**A – buffer**" – at the other end of the alphabet from "**Z- buffer**".

✓ "A – buffer" represents an,

- Antialiased
- Area – averaged
- Accumulation buffer

✓ It is developed by Lucasfilm in surface rendering system.

Drawback for z- buffer algorithm

1. It is time consuming process.
2. Only find, one visible surface at each pixel position.
3. It cannot accumulate intensity values for more than one surface.

✓ A-buffer method expands the depth buffer, so that each position in the buffer can reference a **linked list** of surfaces.

✓ More than one surface intensity can be taken into consideration each pixel position, and object edges can be antialiased.

✓ Each position in the A – buffer has 2 fields.

✓ **Depth field** – It store a positive or negative real number.

✓ **Intensity field** – It store surface intensity information (or) a pointer value.

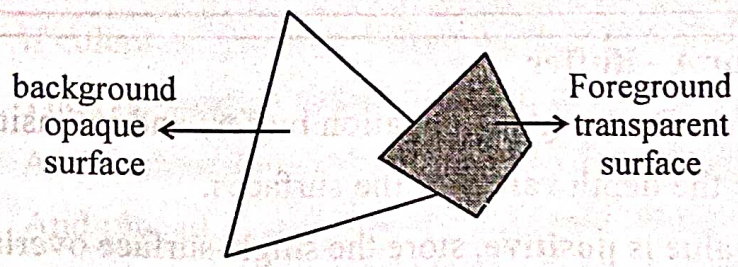

Fig.3.99

Single surface overlap

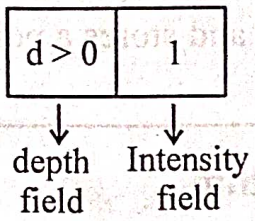

Fig.3.100 Single surface overlap

- ✓ In the depth field is positive, the number stored at that position in the depth of a single surface overlapping the corresponding pixel area.
- ✓ The intensity field then stores the RGB components of the surface color at that point.

Multiple surface overlap

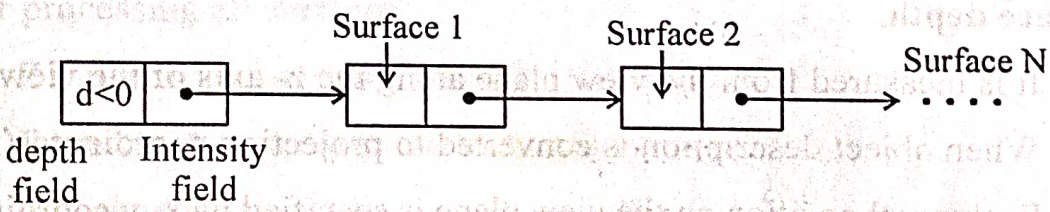

Fig.3.101 Multiple surface overlap

- ✓ If the depth field is Negative, limited list indicates multiple contributors to the pixel intensity.
- ✓ The intensity field can stores a pointer to a linked list of surface data.
- ✓ Data for each surface in the linked list includes,

1. RGB intensity components
2. Opacity (transparency) parameters
3. Depth
4. Area coverage percentage
5. Surface identifier
6. Surface rendering parameters
7. Pointer to text surface, etc.

Algorithm for A –Buffer

1. Initialize A- Buffer (Accumulation buffer) and intensity values.
2. Calculate the depth value for the surfaces.
3. If depth value is **positive,** store the single surface overlapping pixel area and stores a RGB components.
4. If depth value is **negative,** store the multiple surface overlapping pixel area in the linked list and stores a point values in the intensity field.
5. Stop.

3.23.4 Depth Buffer Algorithm

- ✓ It is also called "**Z – buffer**" algorithm.
- ✓ It is a "Image space approach".
- ✓ It is developed by Catmull.
- ✓ "This algorithm **compares surface depths** at each pixel position on the projection plane".

Surface depth

- ✓ It is measured from the view plane along the z- axis of the viewing system.
- ✓ When object description is converted to projection coordinates(x, y, z).
- ✓ Each pixel position on the view plane is specified by x, y coordinate, and 'Z' value given the depth information.
- ✓ Object depth can be compared by 'Z' values.
- ✓ The implementation requires another memory called "Z buffer" along with the frame buffer memory required for raster display devices.
- ✓ "Z buffer" is used to store depth values for each (x, y) position as surfaces are processed, and frame buffer store the intensity values for each position.
- ✓ At the beginning of Z – buffer is initialized to zero, representing the z value at the back clipping plane. Frame buffer is initialized to the background color.
- ✓ Each surface listed in the display files is then process one scan line at a time.
- ✓ Calculating the depth (Z- value) of each (x, y) pixel position.
- ✓ The calculated depth value is compared to the value previously stored in the Z- buffer at the position.

If calculation depth value > stored Z buffer value
Then new depth value is stored
And surface intensity at that position is determined
And placed in the same x, y location in the frame buffer.

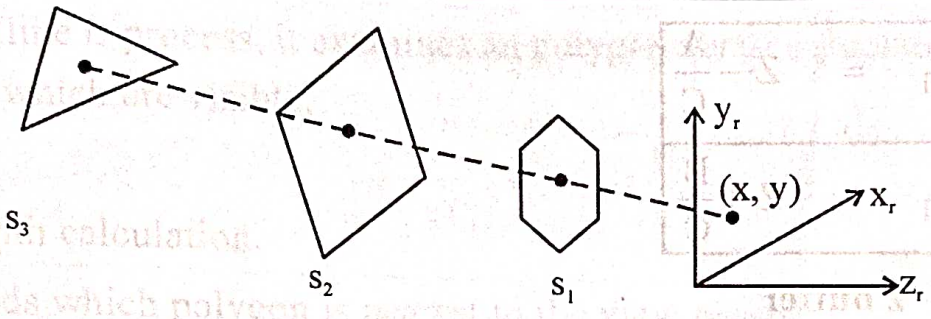

Fig.3.102

- ✓ In the above figure,

  s_1, s_2, s_3 denoted surfaces.
- ✓ Surface s_1 has the smallest depth of view position (x, y), hence highest Z-value.
- ✓ Surface s_3 has the lowest depth at view position (x, y).
- ✓ After processing all surfaces,
  - Z buffer contains depth value for the visible surfaces.
  - Frame buffer contains intensity values for those surfaces.
  - To calculate Z values,

$$\boxed{A_x + B_y + C_z + D = 0}$$

- ✓ Here, (x, y, z) are point in the plane.

  A, B, C, D – spatial properties (constant values)
- ✓ Threfore,

$$\boxed{Z = \frac{-Ax - By - D}{C}}$$

- ✓ Note, if at (x, y) the above equation evalutes to Z_1, then at $(x + \Delta x, y)$, the value of Z is,

$$\boxed{Z_1 = \frac{A}{C}(\Delta x)}$$

- ✓ Only one subtraction is needed to calculate $Z(x+1, y)$, given (z, (x,y)).

- ✓ Since the Quotient $\frac{A}{C}$ is constant and $\Delta x = 1$.
- ✓ A similar incremental calculation can be performed to determine the first value of z on the next scan line, decrementing by $\frac{B}{C}$ for each Δy.

Z_1	=	$Z - \frac{A}{C}$
Z_1	=	$Z + \frac{B}{C}$

Algorithm – z buffer

1. Initialize buffer position.

   Z- buffer (x, y) = 0

   Framebuffer (x, y) = $I_{background}$
2. Conversion process

   For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
3. Calculate Z – value for (x, y) plane position on the polygon.

   If Z > Z- buffer (x, y)

   Then set Z- buffer = Z

   Framebuffer (x, y)= $I_{surface}$ (x, y)
4. Stop

Note

- ✓ $I_{background}$ – Background intensity.
- ✓ $I_{surface}$ –It is the projected intensity value for the surface at pixel position (x, y)

Z – buffer advantages

- ✓ Easy to implement for larger surfaces.
- ✓ Overcome speed problem with hardware implementation.
- ✓ It is efficient for many objects.

Z- buffer disadvantages

- ✓ It required additional buffer (So need large memory).
- ✓ It is a time consuming process (because it is based pixel comparison technique).

3.23.6 BSP –Tree method

- ✓ BSP stands for "**Binary space partitioning**" tree.
- ✓ In this method determining object visibility by painting surfaces onto the screen from back to front, as in the painters algorithm.

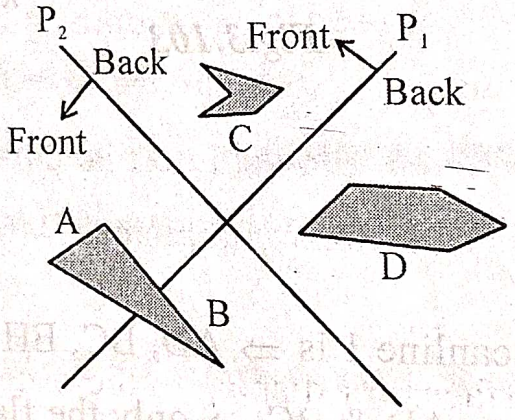

Fig.3.104

- ✓ It is particularly useful for when the view reference point changes (But, objects in the scene are fixed position).
- ✓ Applying a BSP tree to visibility testing involves identifying surfaces that are "inside" & "outside" the partitioning plane of each step of the space subdivision, relative to the viewing position.
- ✓ Here, P_1 & P_2 plane for partitioning.
- ✓ A, B, C, D are objects.

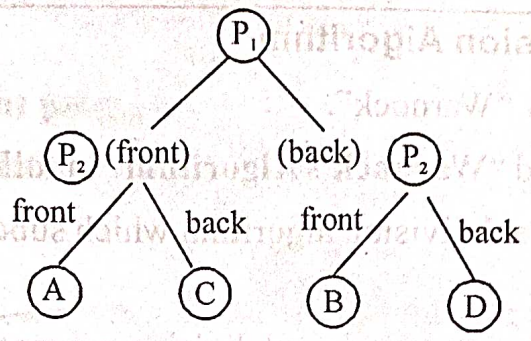

Fig.3.105 BSP – Tree representation

- ✓ With plane P_1, we first partition the space into 2 set of objects.
- ✓ One set of objects is behind or in back of, plane P_1 relative to the viewing direction, and the other set in front of P_1.
- ✓ Since one object is intersected by plane P_1, we divide that object into 2 separate objects, labeled A & B.
- ✓ Object A & C are in front of P_1 & objects B& D are behind P_1.
- ✓ Next partition the space again with plane P_2 & construct the binary tree representation.
- ✓ Objects are represented as **terminal nodes.**
- ✓ Polygon equation are to be used to partitioning planes for identifying "inside" (or) "outside" polygons.
- ✓ Intersecting polygon are partitioned into two parts.

  if BSP tree is selection / completed,

  then

  Selecting surfaces for display in the order of back to front.

BSP Tree – Algorithm

1. Partitioning the object in a view plane to the fixed positions.
2. Identify the surfaces that are 'inside' or 'outside' the partition plane.
3. Construct a Binary tree space representation.
4. Select and complete the BSP Tree.
5. Selecting the surface for display in the order of back to front.
6. Stop.

3.23.7 Area Subdivision Algorithm

- ✓ It is developed by "Warnock".
- ✓ So, it is also called "**Warnock's Algorithm**" In follow object space approach.
- ✓ He develops area subdivision algorithm which subdivides each area into four equal squares.
- ✓ At each stage in the recursive – subdivision process, the relationship between projection of each polygon & the area of interest is checked for four possible relationships.

1. Surrounding polygon

- ✓ One that completely encloses area of in interest.

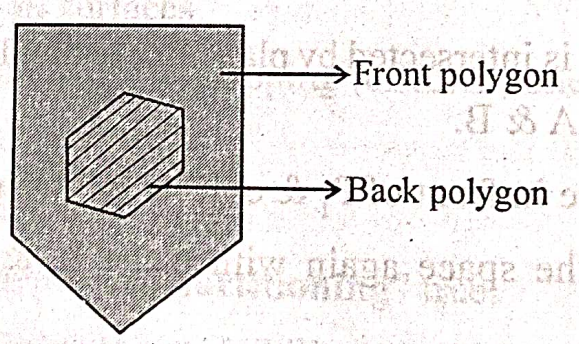

Fig.3.106

2. Overlapping (or) intersecting polygon

- ✓ One that is partially inside and partially outside the area.

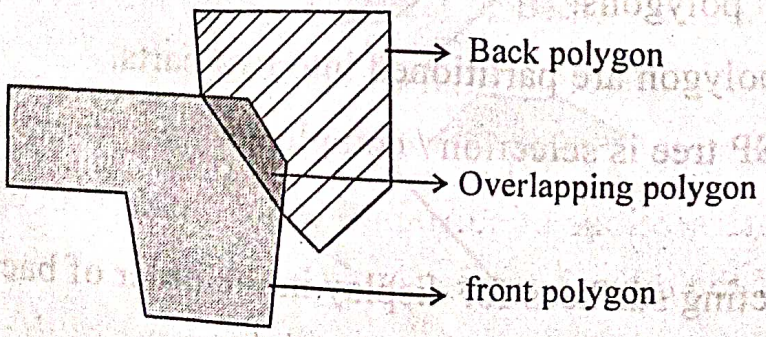

Fig.3.107

3. **Inside (or) contained polygon**

✓ One that is completely inside the area.

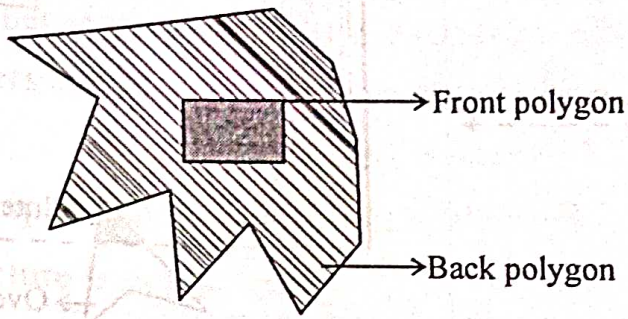

Fig.3.108

4. **Outside (or) Disjoint polygon**

✓ One that is completely outside the area.

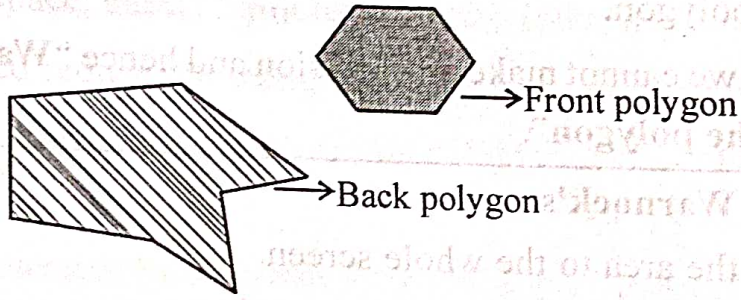

Fig.3.109

✓ Alter checking four relationships, we can handle each relationship as follows.

1. If all the polygons disjoint from the area, then the background colors is displayed in the area.
2. If there is only one intersecting or only one intersecting polygon (or) contained polygon, then the area is first filled with the background color, and then the part of the polygon contained in the area is filled with color of polygon.
3. If there is a single surrounding polygon, but no intersecting or contained polygons, then the area is filled with the color of the surrounding polygon.
4. If there are more than one polygon intersecting, contained in or surrounding the area then we have to do some more processing.

✓ The four intersections of surrounding polygon are all closer to the view point than any of the other intersections.

✓ Therefore, the entire area is filled with the color of the surrounding polygon.

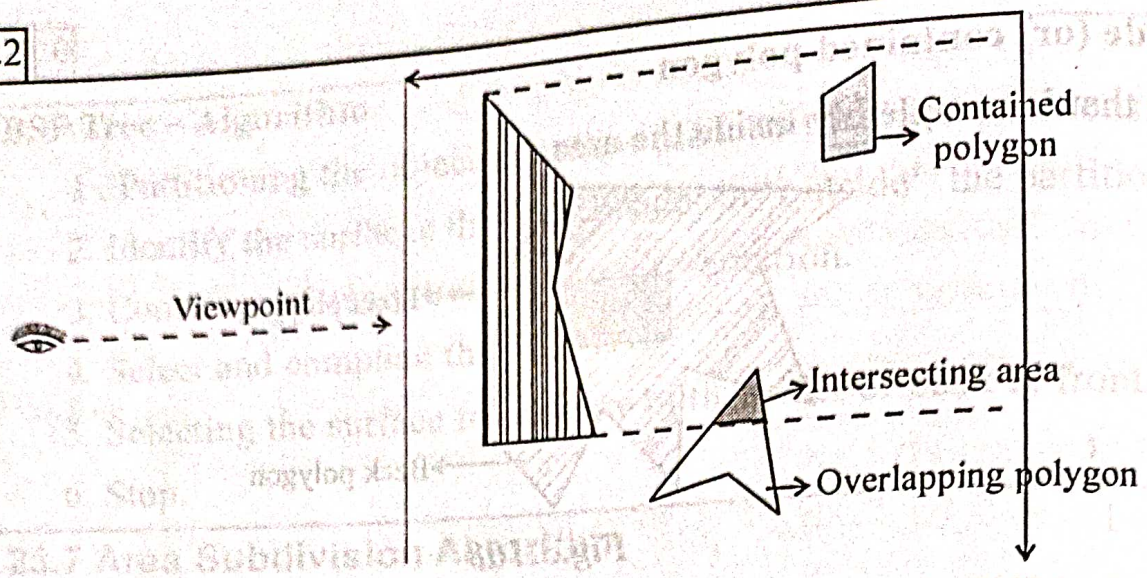

Fig.3.110

- ✓ In some cases, surrounding polygon is not completely in front of the intersecting polygon.
- ✓ In such cases we cannot make any decision and hence **"Warnock's algorithm subdivides the polygon"**.

Algorithm – Warnock's

1. Initialize the area to the whole screen.
2. Create the list of polygon and sorting them.
3. Find the relationship of each polygon.
4. Perform the visibility decision test.
   a) If all the polygon are disjoint from the area, then fill background color.
   b) If there is only are intersection or only are contained polygon then first fill entire area with background color and then fill the part of the polygon contained in the area with the color of polygon.
   c) If there is a single surrounding polygon, but no intersecting or contained polygon, then fill the area with the surrounding polygon.
   d) If surrounding polygon is closer to the view point than all other polygon, so that all other polygons are hidden by it, fill the area with the color of the surrounding polygon.
   e) If the area pixel is (x, y), and neither a, b, c nor d applies, computes the z coordinate at pixel (x, y) of all polygons in the list. The pixel is then set to colors of the polygon which is closer to the viewpoint.
5. If none of the above tests are true than subdivide the area and go to step 2.

Advantages

1. No need for extra memory.
2. Support 'n' number of polygon.
3. Parallel computers can be used to speed up process.