COMPUTER GRAPHICS
UNIT-IV

3D TRANSFORMATION VIEWING AND CLIPPING

R.MANIMEGALAI
DEPARTMENT OF COMPUTER SCIENCE
PERIYAR GOVT ARTS COLLEGE
CUDDALORE.

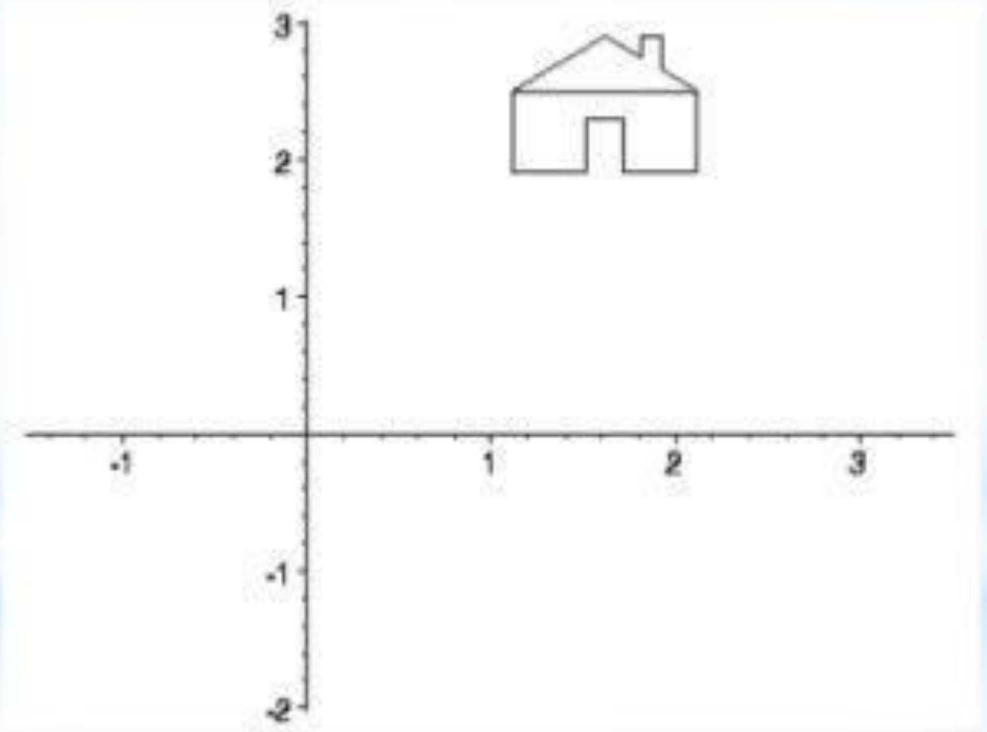
OUR PRESENTATION ON

3D TRANSFORMATION

CONTENTS

TRANSFORMATION

- Transformations are a fundamental part of the computer graphics. Transformations are the movement of the object in Cartesian plane .



TYPES OF TRANSFORMATION

- There are two types of transformation in computer graphics.

  1) 2D transformation
  2) 3D transformation

- Types of 2D and 3D transformation

  1) Translation
  2) Rotation
  3) Scaling
  4) Shearing
  5) Mirror reflection

WHY WE USE TRANSFORMATION

- Transformation are used to position objects , to shape object , to change viewing positions , and even how something is viewed.
- In simple words transformation is used for

  1) Modeling

  2) viewing

3D TRANSFORMATION

- When the transformation takes place on a 3D plane .it is called 3D transformation.
- Generalize from 2D by including **z** coordinate

Straight forward for translation and scale, rotation more difficult

Homogeneous coordinates: 4 components

Transformation matrices: 4×4 elements

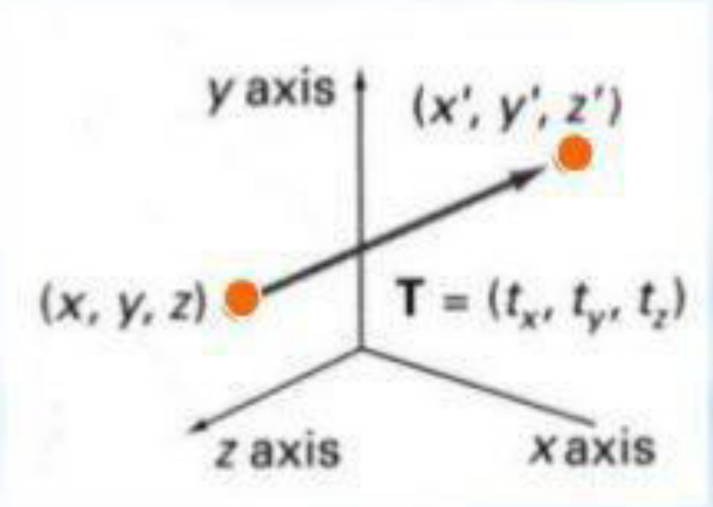
$$\begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D TRANSLATION

- Moving of object is called translation.
- In 3 dimensional homogeneous coordinate representation , a point is transformed from position P = (x, y , z) to P'=(x', y', z')
- This can be written as:-

Using **P' = T . P**

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

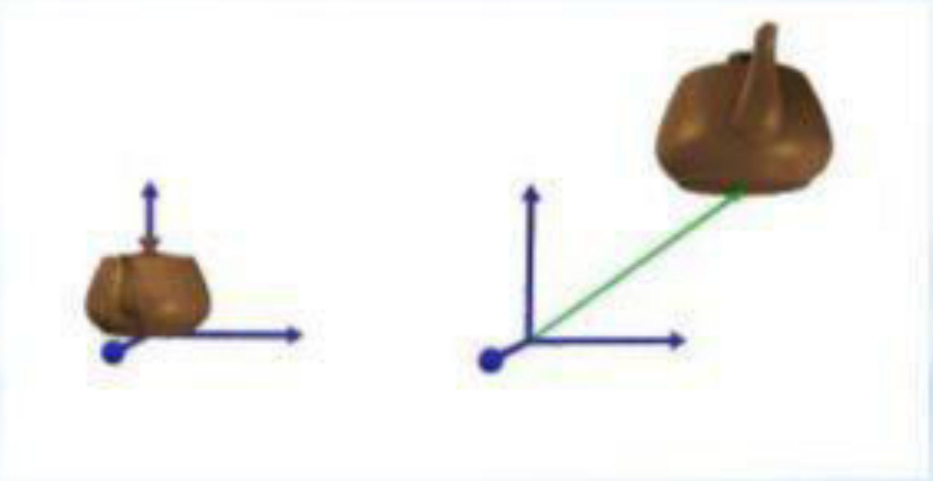


3D TRANSLATION

- The matrix representation is equivalent to the three equation.

  $x' = x + t_x$, $y' = y + t_y$, $z' = z + t_z$

Where parameter t_x, t_y, t_z are specifying translation distance for the coordinate direction x , y , z are assigned any real value.

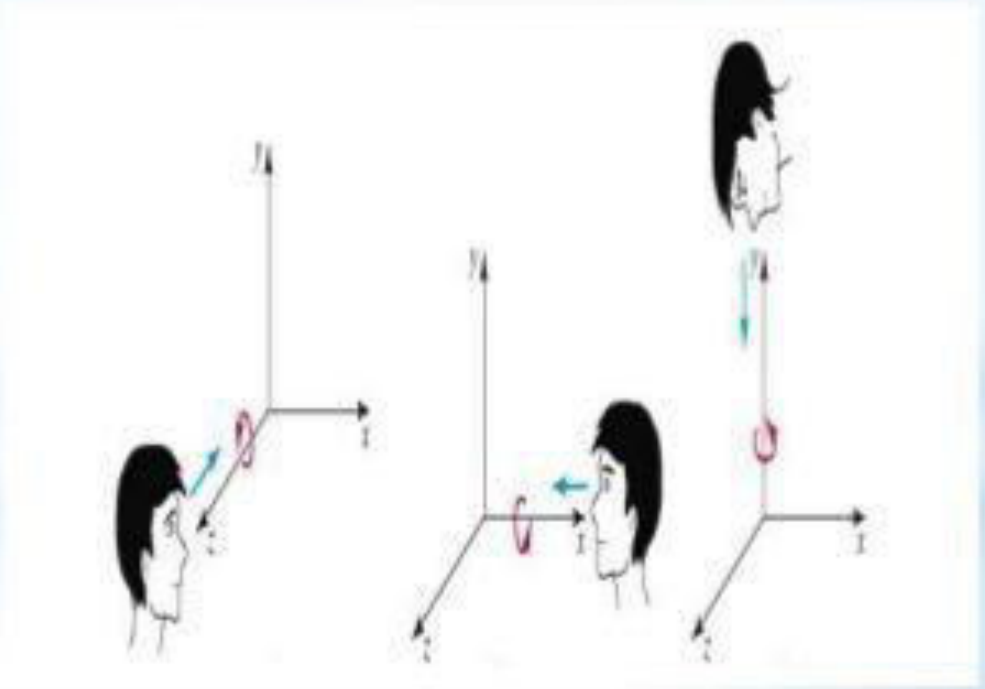


3D ROTATION

Where an object is to be rotated about an axis that is parallel to one of the coordinate axis, we can obtain the desired rotation with the following transformation sequence.

Coordinate axis rotation

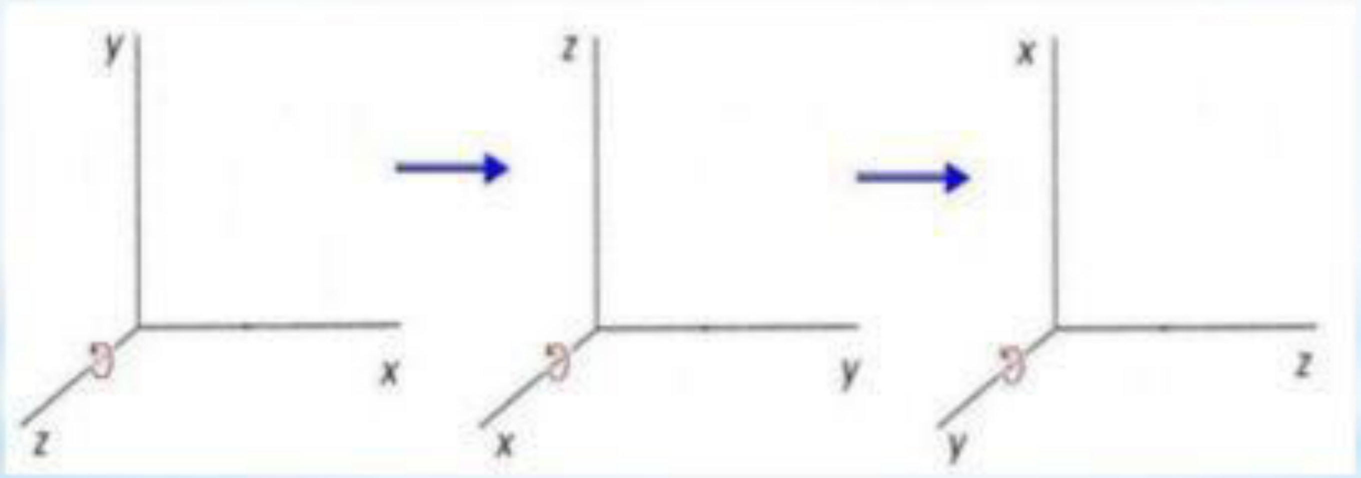
Z- axis Rotation(Roll)
Y-axis Rotation(Yaw)
X-axis Rotation(Pitch)



COORDINATE AXIS ROTATION

- Obtain rotations around other axes through cyclic permutation of coordinate parameters:

$$x \to y \to z \to x$$



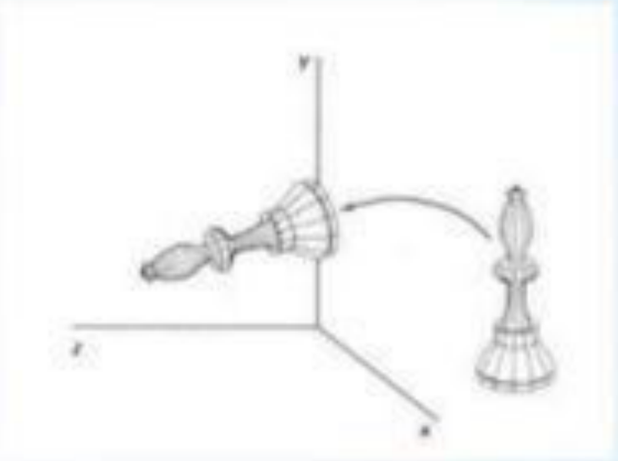
X-AXIS ROTATION

The equation for X-axis rotation

$x' = x$

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

$z' = y \sin\theta + z \cos\theta$



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

Y-AXIS ROTATION

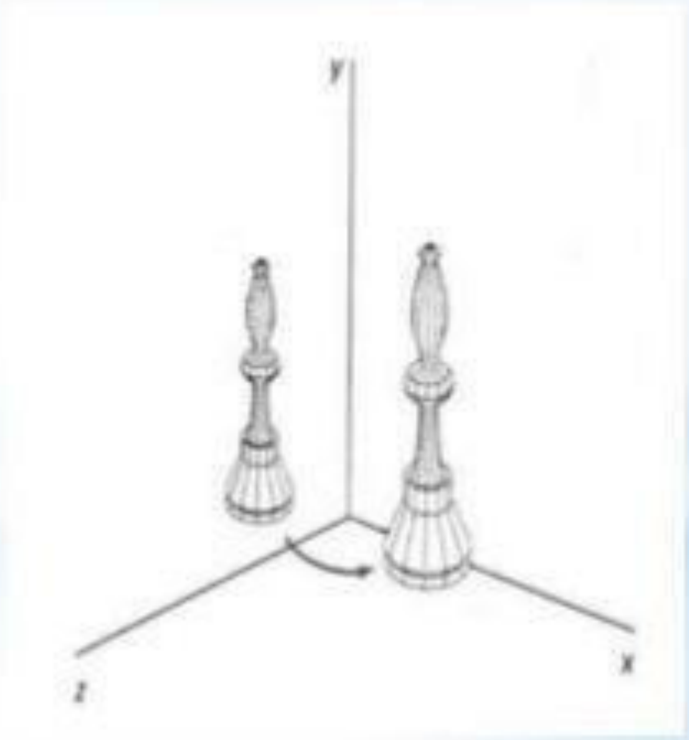
The equation for Y-axis rotaion

x' = x cosθ + z sinθ

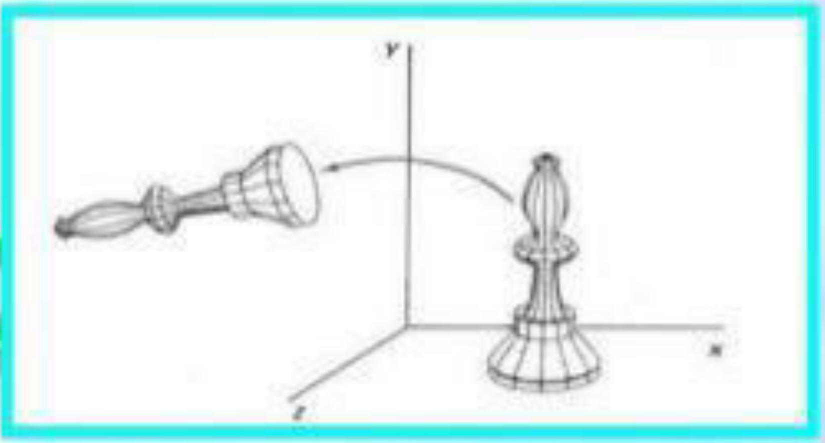
y' = y

z' = z cosθ - x sinθ

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



Z-AXIS RO



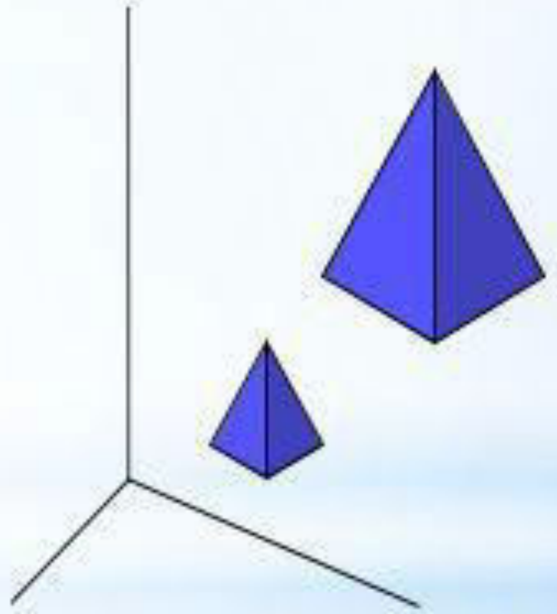
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$z' = z$

3D SCALING

- Changes the size of the object and repositions the object relative to the coordinate origin.

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



3D SCALING

- The equations for scaling

$$S_{sx,sy,sz} \;\square\; \begin{matrix} x' = x \,.\, sx \\ y' = y \,.\, sy \\ z' = z \,.\, sz \end{matrix}$$

3D REFLECTION

- Reflection in computer graphics is used to emulate reflective objects like mirrors and shiny surfaces
- Reflection may be an x-axis

y-axis , z-axis. and also in
the planes xy-plane,yz-plane , and
zx-plane.

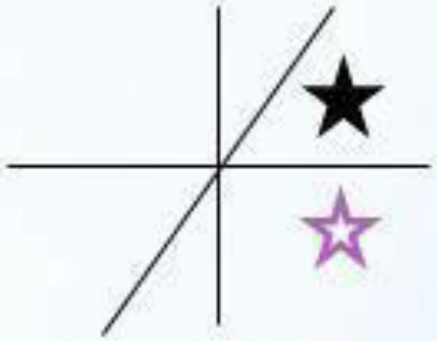
Reflection relative to a given
Axis are equivalent to 180
Degree rotations

3D REFLECTION

- Reflection about x-axis:-

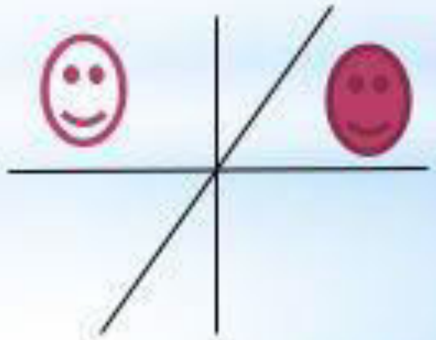
x'=x y'=-y z'=-z

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$



Reflection about y-axis:-

y'=y x'=-x z'=-z

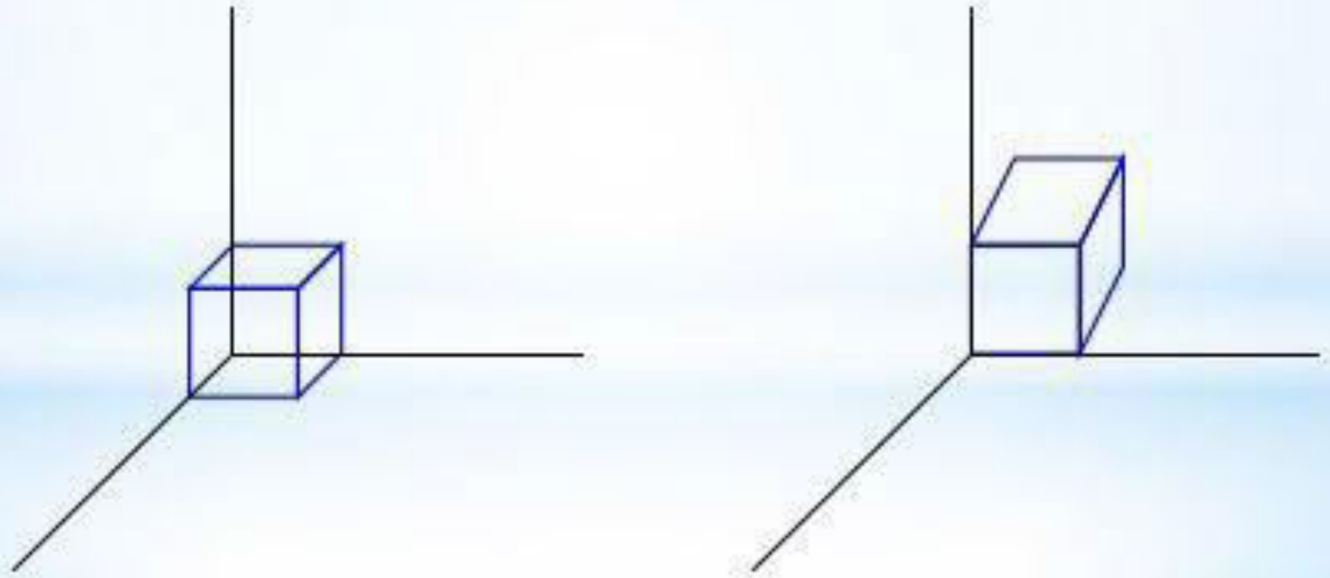


3D SHEARING

- **Modify object shapes**
- **Useful for perspective projections**
- When an object is viewed from different directions and at different distances, the appearance of the object will be different. Such view is called perspective view. Perspective projections mimic what the human eyes see.

3D SHEARING

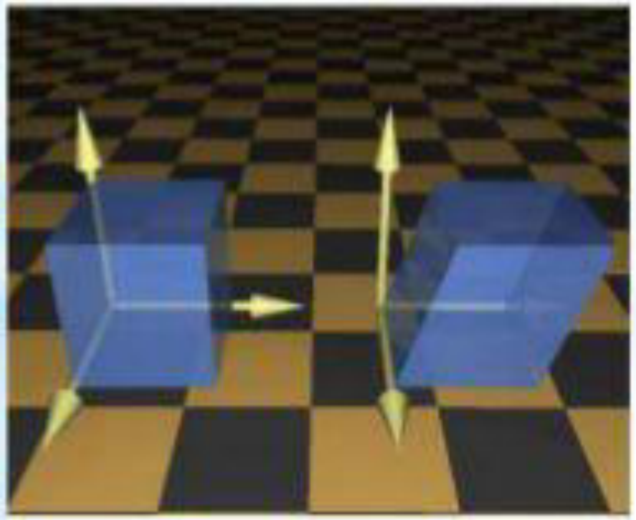
E.g. draw a cube (3D) on a screen (2D) Alter the values for **x** and **y** by an amount proportional to the distance from z_{ref}



3D SHEARING

- Matrix for 3d shearing
- Where a and b can
  Be assigned any real
  Value.

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



PROJECTION

Transform 3D objects on to a 2D plane using ***projections***

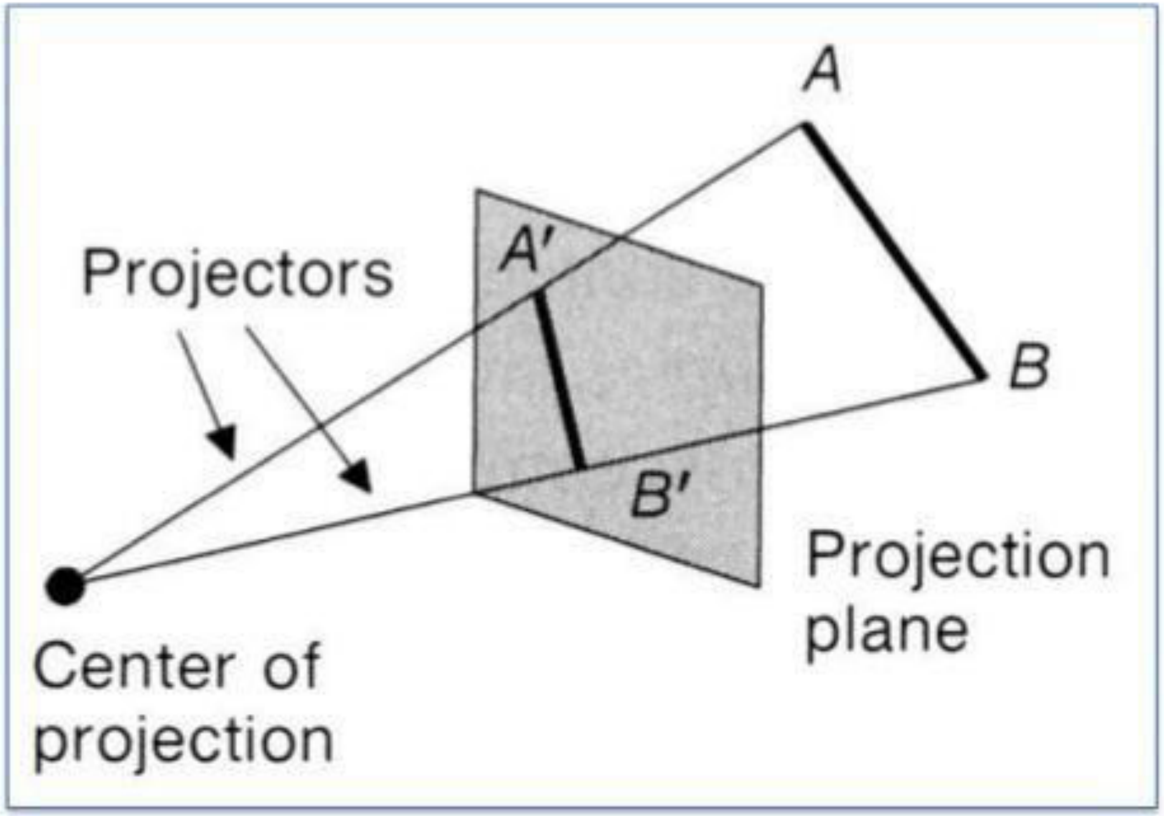
2 types of projections

Perspective

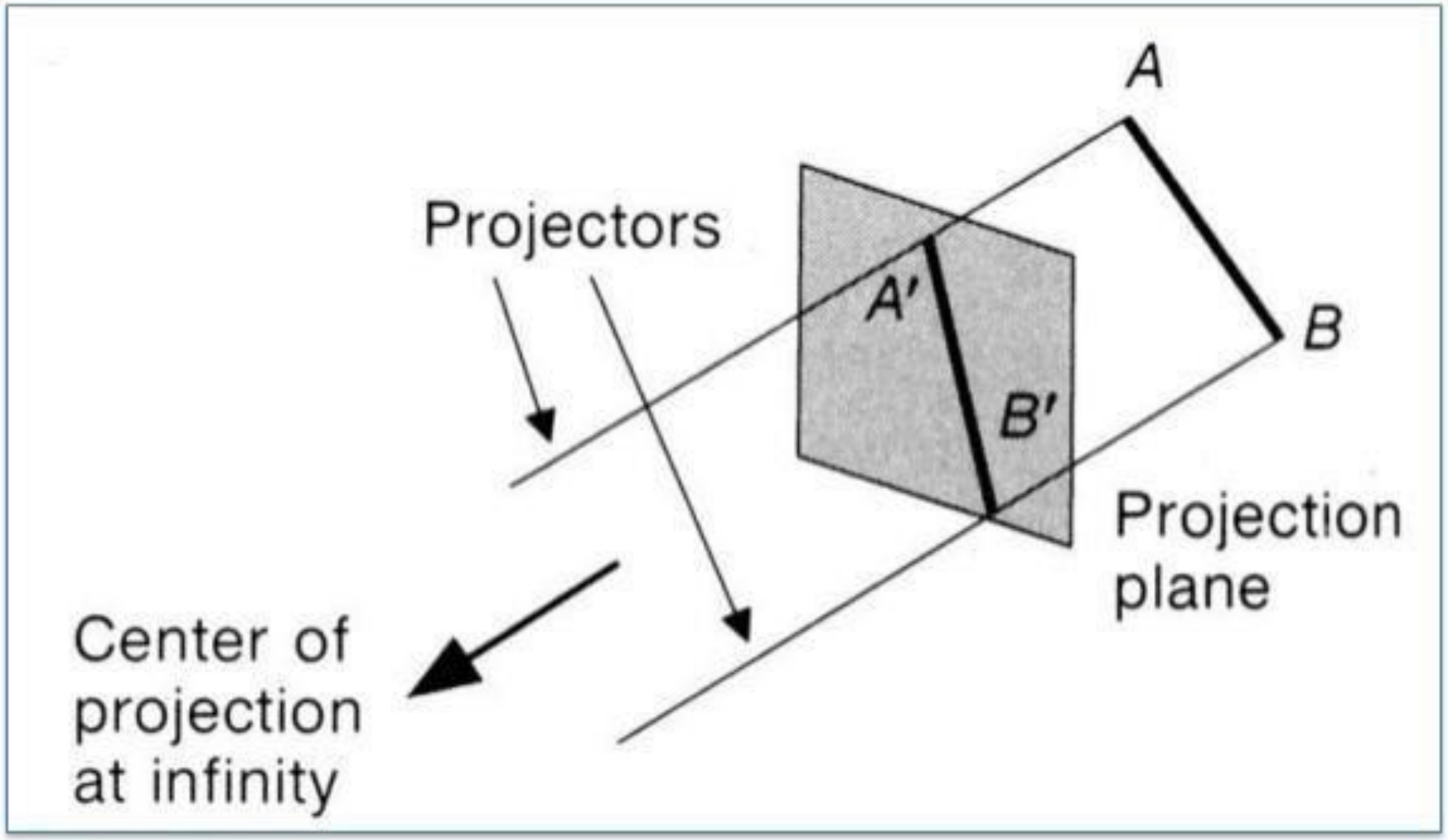
Parallel

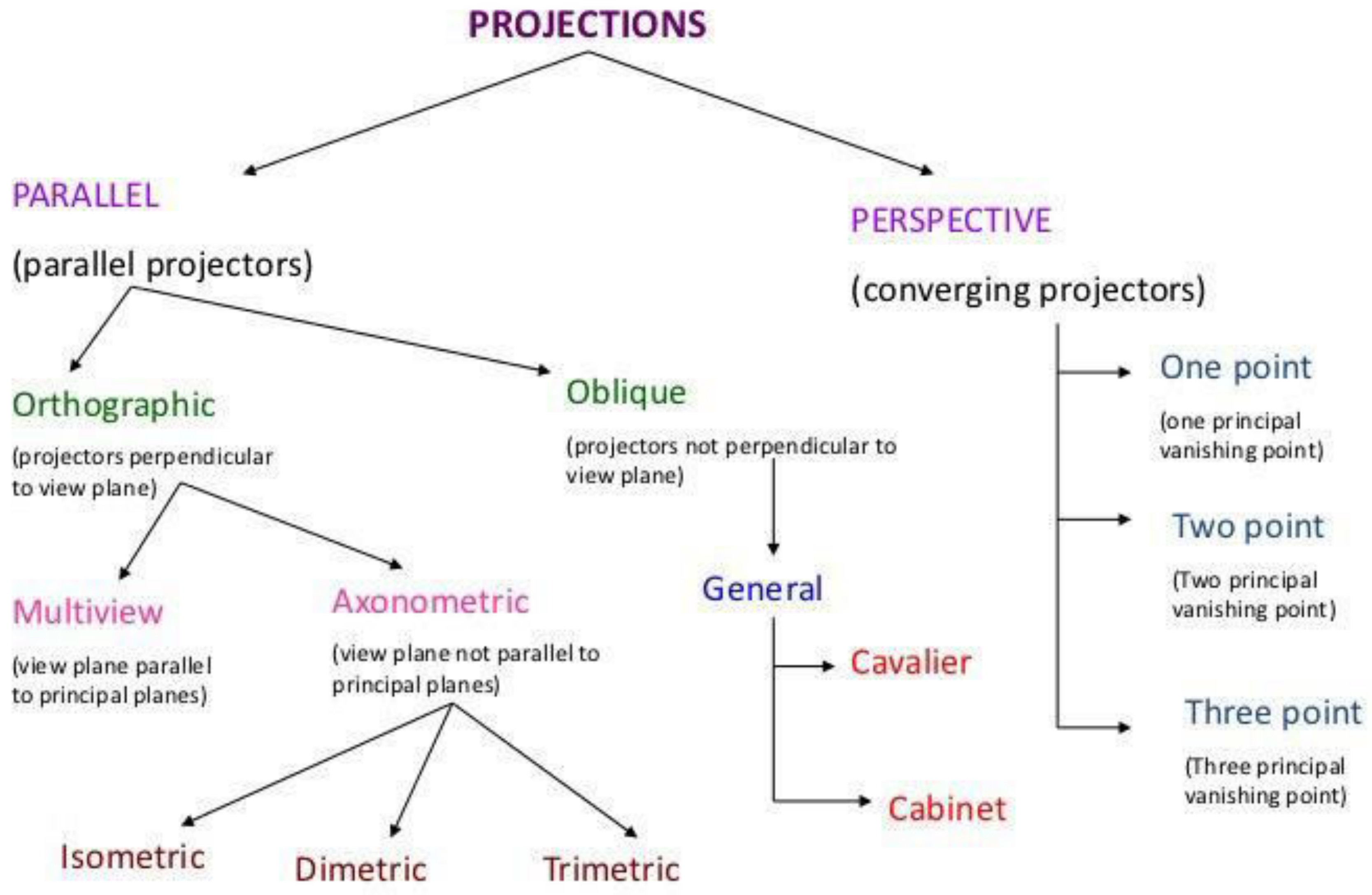
In **parallel projection**, coordinate positions are transformed to the view plane along parallel lines.
In **perspective projection**, object position are transformed to the view plane along lines that converge to a point called **projection reference point (center of projection)**

Perspective Projection



Parallel Projection




PROJECTIONS
PARALLEL
(parallel projectors)
PERSPECTIVE
(converging projectors)
Orthographic
(projectors perpendicular to view plane)
Oblique
(projectors not perpendicular to view plane)
One point
(one principal vanishing point)
Two point
(Two principal vanishing point)
Three point
(Three principal vanishing point)
Multiview
(view plane parallel to principal planes)
Axonometric
(view plane not parallel to principal planes)
General
Cavalier
Cabinet
Isometric
Dimetric
Trimetric

Perspective v Parallel

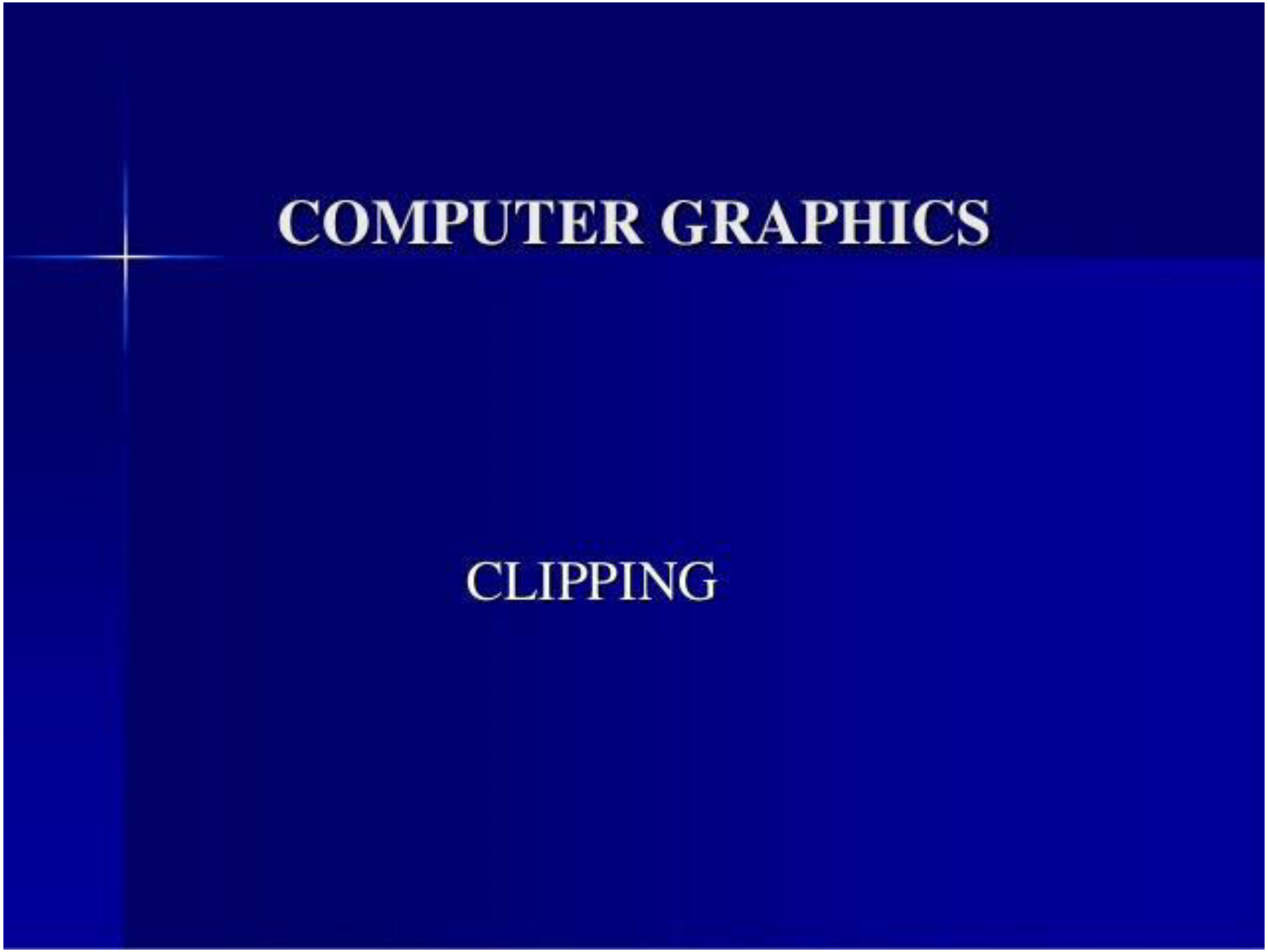
- **Perspective:**
  - visual effect is similar to human visual system...
  - has 'perspective foreshortening'
    - size of object varies inversely with distance from the center of projection. Projection of a distant object are smaller than the projection of objects of the same size that are closer to the projection plane.
- **Parallel:**

  It preserves relative proportion of object.
  - less realistic view because of no foreshortening
  - however, parallel lines remain parallel.

Perspective Projections

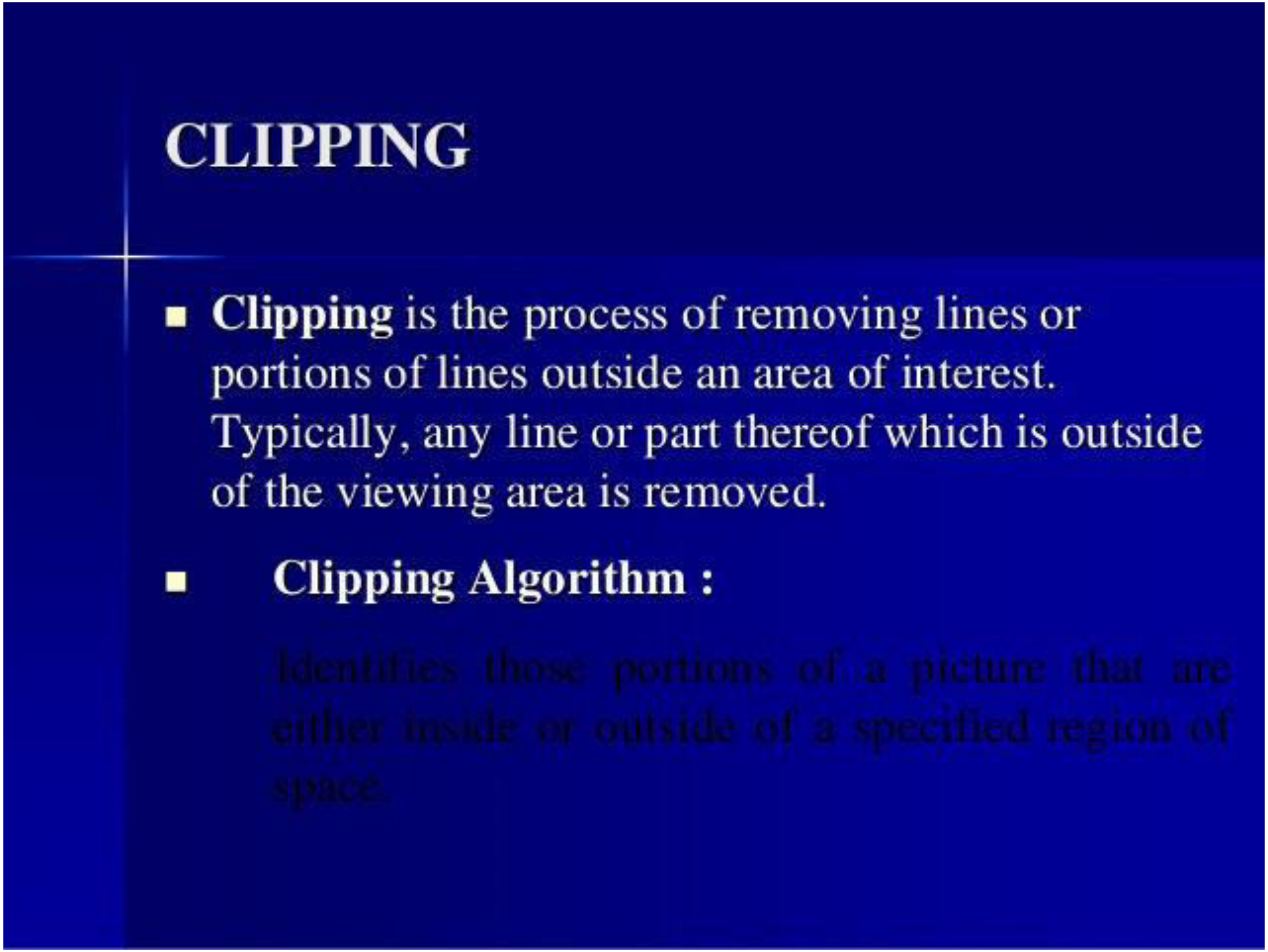
- Characteristics:
- Center of Projection (CP) is a finite distance from object
- Projectors are rays (i.e., non-parallel)
- *Vanishing points*
- Objects appear smaller as distance from CP (eye of observer) increases
- Difficult to determine exact size and shape of object
- Most realistic, difficult to execute

- When a 3D object is projected onto view plane using perspective transformation equations, any set of parallel lines in the object that are *not* parallel to the projection plane, converge at a vanishing point.
  - There are an infinite number of vanishing points, depending on how many set of parallel lines there are in the scene.
- If a set of lines are parallel to one of the three principle axes, the vanishing point is called an *principal vanishing point*.
  - There are at most 3 such points, corresponding to the number of axes cut by the projection plane.



COMPUTER GRAPHICS

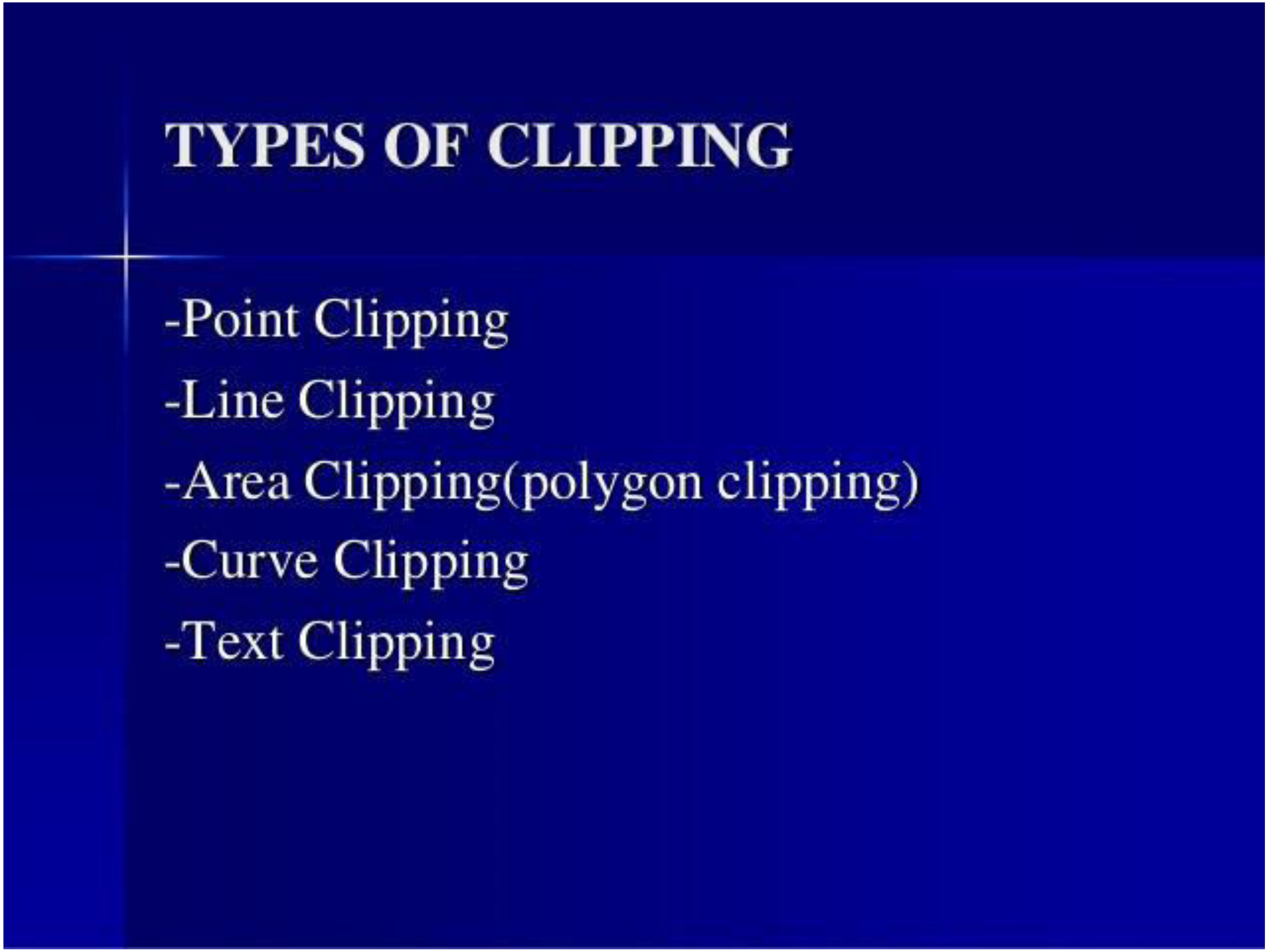
CLIPPING



CLIPPING

- **Clipping** is the process of removing lines or portions of lines outside an area of interest. Typically, any line or part thereof which is outside of the viewing area is removed.
- **Clipping Algorithm :**

  Identifies those portions of a picture that are either inside or outside of a specified region of space.



TYPES OF CLIPPING

-Point Clipping

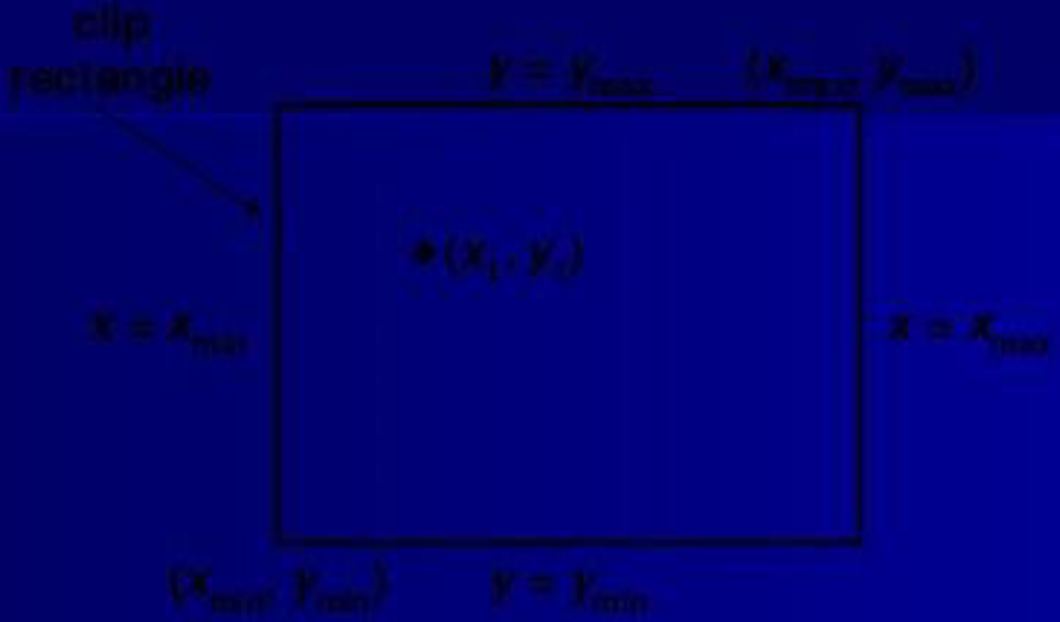
-Line Clipping

-Area Clipping(polygon clipping)

-Curve Clipping

-Text Clipping

Point Clipping



- For a point (x,y) to be inside the clip rectangle:

Point Clipping

Clip window: rectangle, edges of the clip window $(xw_{min}, xw_{max}, yw_{min}, yw_{max})$

If $xw_{min} \leq \mathbf{x} \leq xw_{max}$

$yw_{min} \leq y \leq yw_{max}$

Then

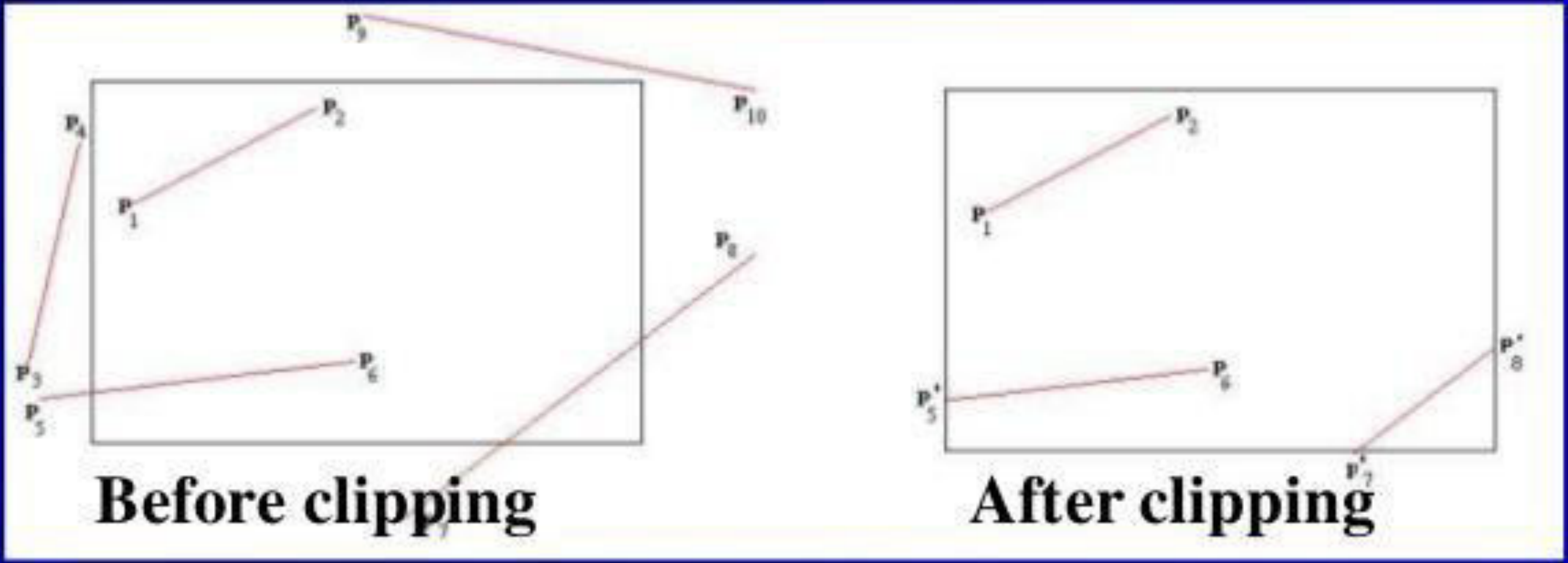
The point $\mathbf{p = (x, y)}$ is saved

for display

Otherwise: the point is clipped (not saved for display).

Line Clipping

Line clipping against a rectangular window



Line Clipping

Inside – Outside test:

- **Completely Inside:** A line with both endpoints **inside** all clipping boundaries, such as the line from $\mathbf{p}_1$ to $\mathbf{p}_2$, is saved.
- **Completely Outside:** A line with both endpoints **outside** any one of the clip boundaries, such as the line from $\mathbf{p}_3$ to $\mathbf{p}_4$, is not saved.
- If the line is not completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries.

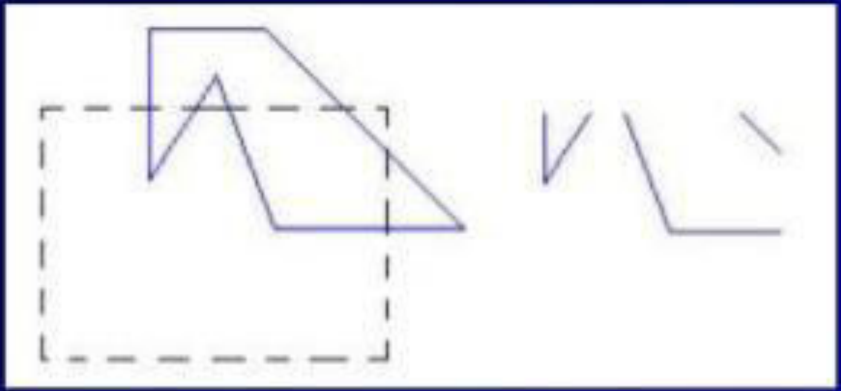
Cohen-Sutherland Line Clipping

- This method speeds up the processing of the line segment by performing initial tests that reduce the number of intersections that must be calculated.
- Every line endpoint in a picture is assigned a four-digit binary code, called **region code** that identifies the location of the point relative to the boundaries of the clipping rectangle.

Bit 4	Bit 3	Bit 2	Bit 1
Above	Below	Right	Left

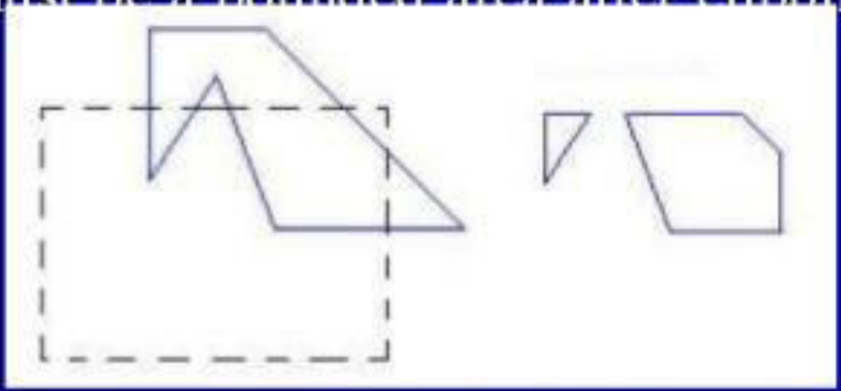
Area Clipping (polygon clipping)

To clip a **polygon**, we cannot directly apply a line-clipping method to the individual polygon edges because this approach would produce a series of **unconnected line segments** as shown in figure .



Area Clipping (polygon clipping)

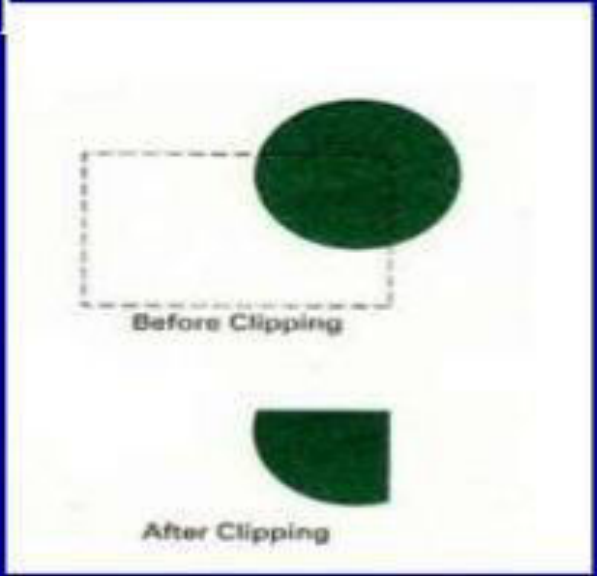
- The clipped polygons must be a bounded area after clipping as shown in figure.



- For polygon clipping, we require an algorithm that will generate one or more **closed areas** that are then scan converted for the appreciate area fill.
- The output of a polygon clipper should be a **sequence of vertices** that defines the clipped polygon boundaries.

Curve Clipping

- Curve clipping procedures will involve non-linear equations.
- So requires more processing than for objects with linear Boundaries.

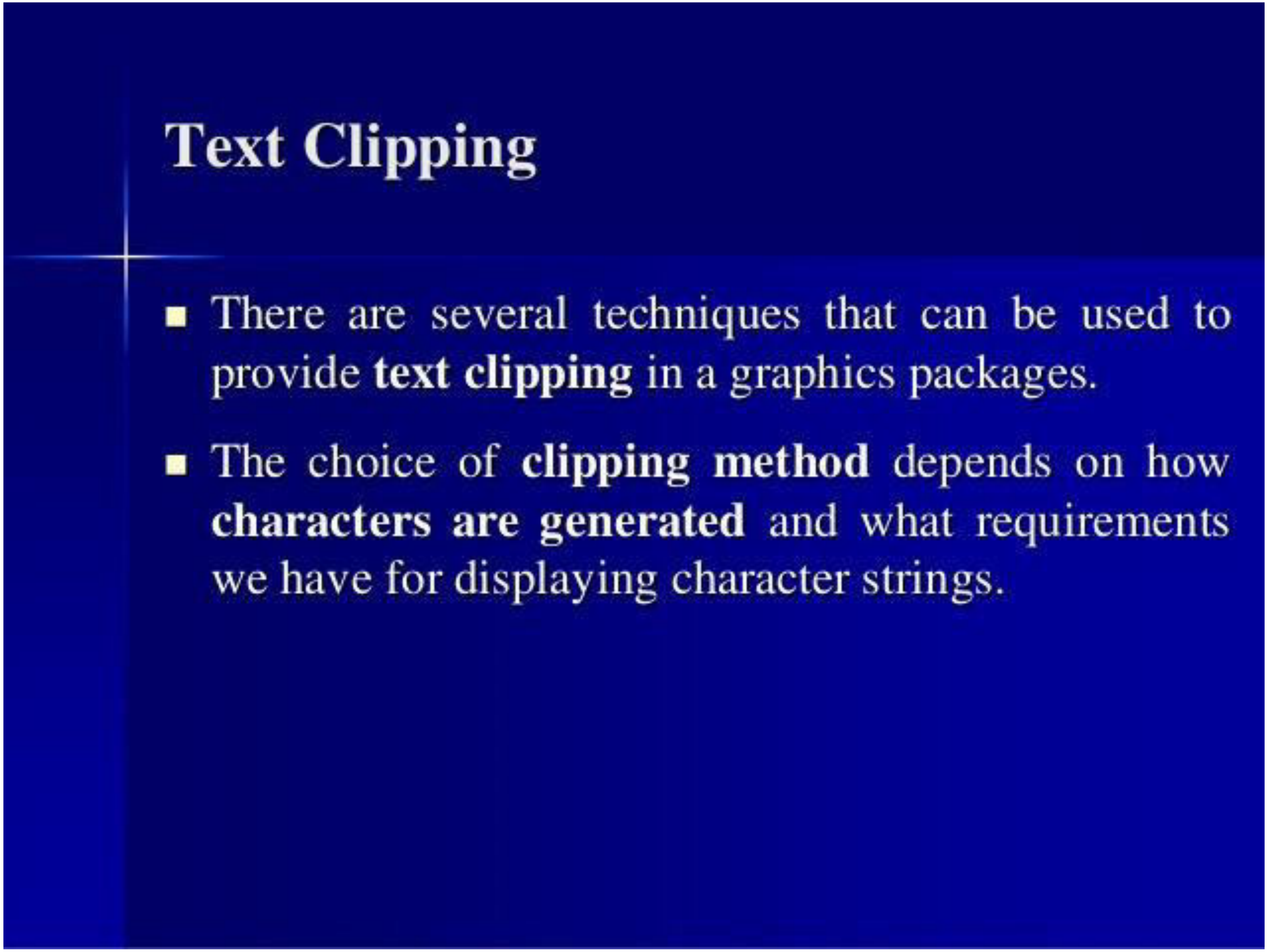


Curve Clipping

- Preliminary test (Test for overlapping)
- -The bounding rectangle for a circle or other curved object is used to test for overlap with a rectangular clip window.
- -If the bounding rectangle is completely inside (save object), completely outside (discard the object)
- -Both cases-no computation is necessary.
- -If bounding rectangle test fails, use computation-saving approaches.

Curve Clipping

- Circle-coordinate extents of individual quadrants & then octants are used for preliminary testing before calculating curve-window intersections
- Ellipse- coordinate extents of individual quadrants are used.
- If 2 regions overlap, solve the simultaneous line-curve equations to obtain the clipping intersection points.

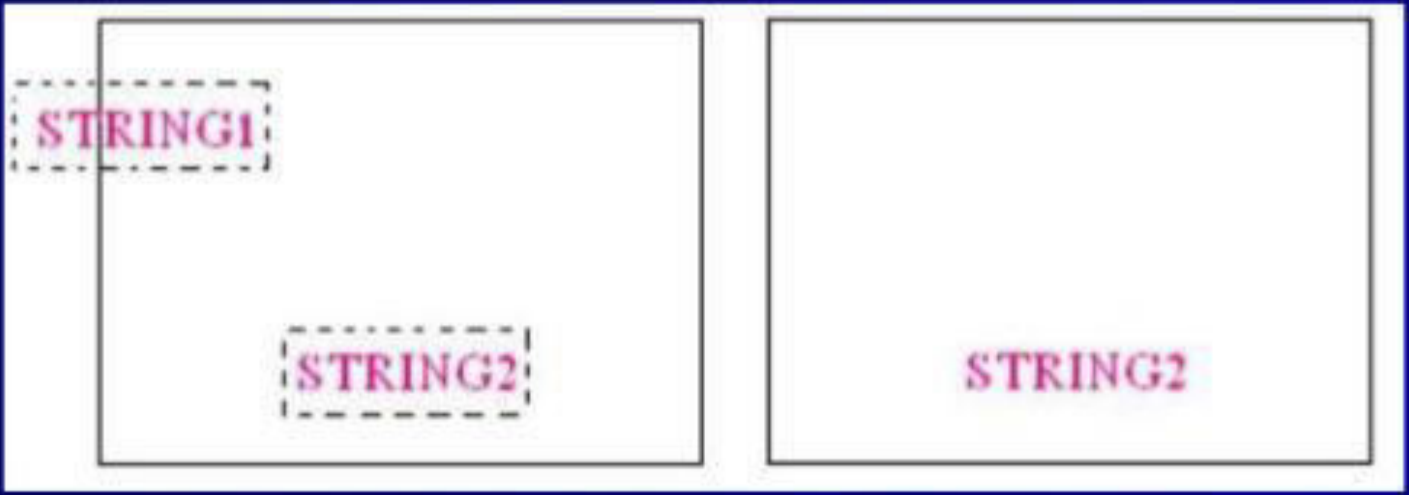


Text Clipping

- There are several techniques that can be used to provide **text clipping** in a graphics packages.
- The choice of **clipping method** depends on how **characters are generated** and what requirements we have for displaying character strings.

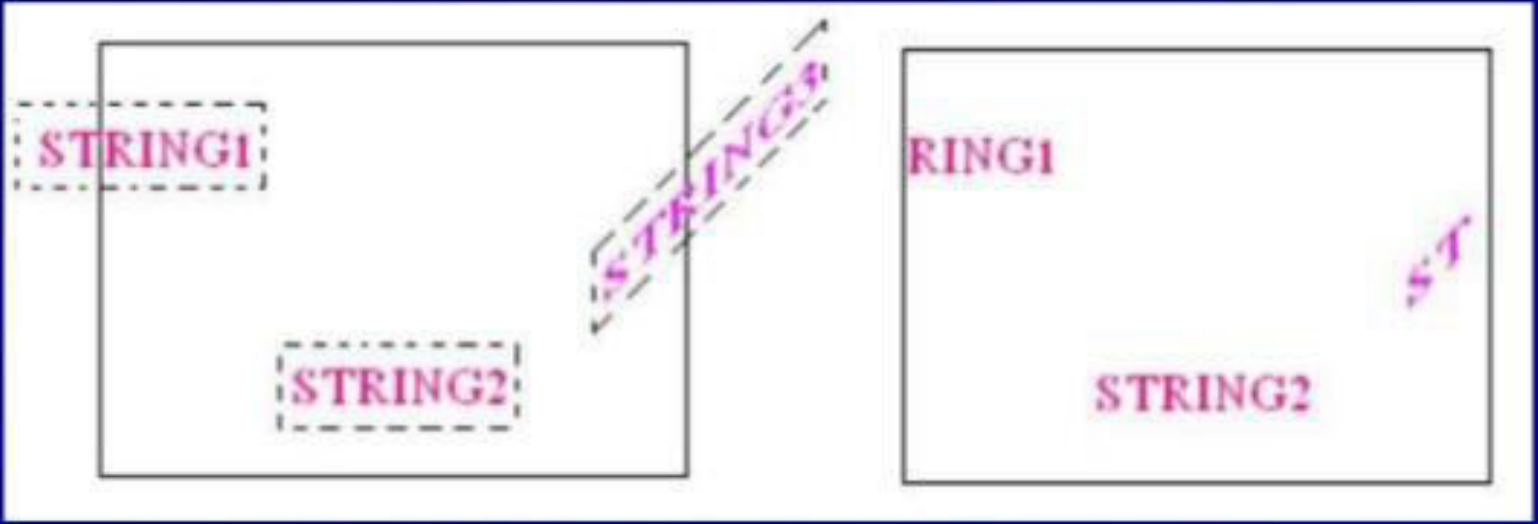
Text Clipping

- **All-or-none string-clipping**
- If all of the string is **inside** a clip window, we keep it.



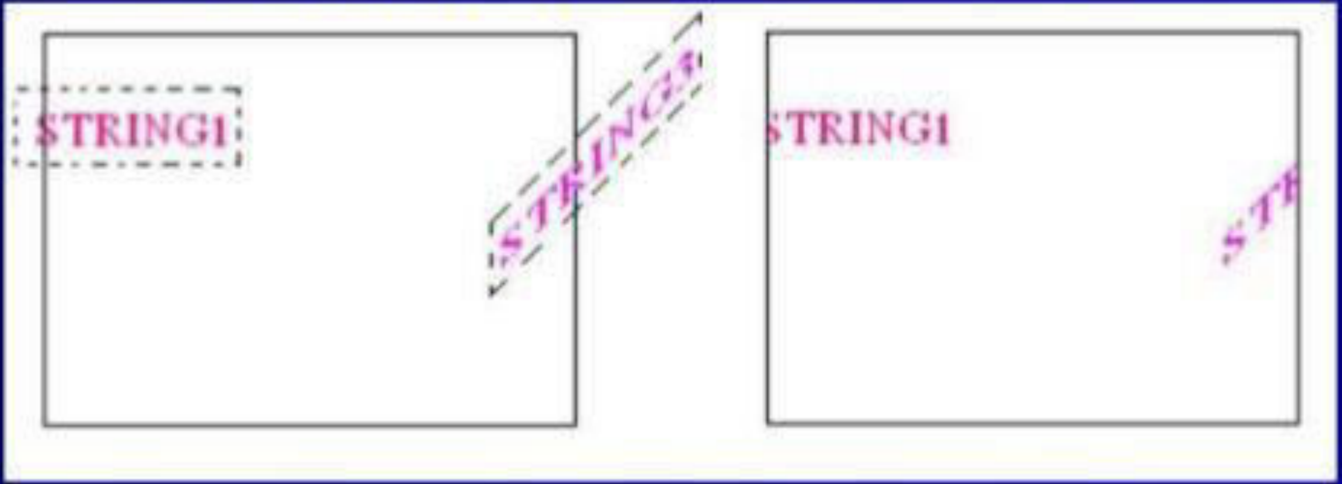
Text Clipping

- **All-or-none character-clipping**
- Here we discard only those characters that are not completely inside the window



Text Clipping

- **Clip the components of individual characters**
- We treat characters in much the same way that we treated lines.
- If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window


STRING1
STRING3
STRING1
STR