Parul® University

05201330 / 05202182 – Computer Graphics

Prof. Dampy Singh, Assistant Professor
Parul Institute of Engineering and Technology - MCA

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-4

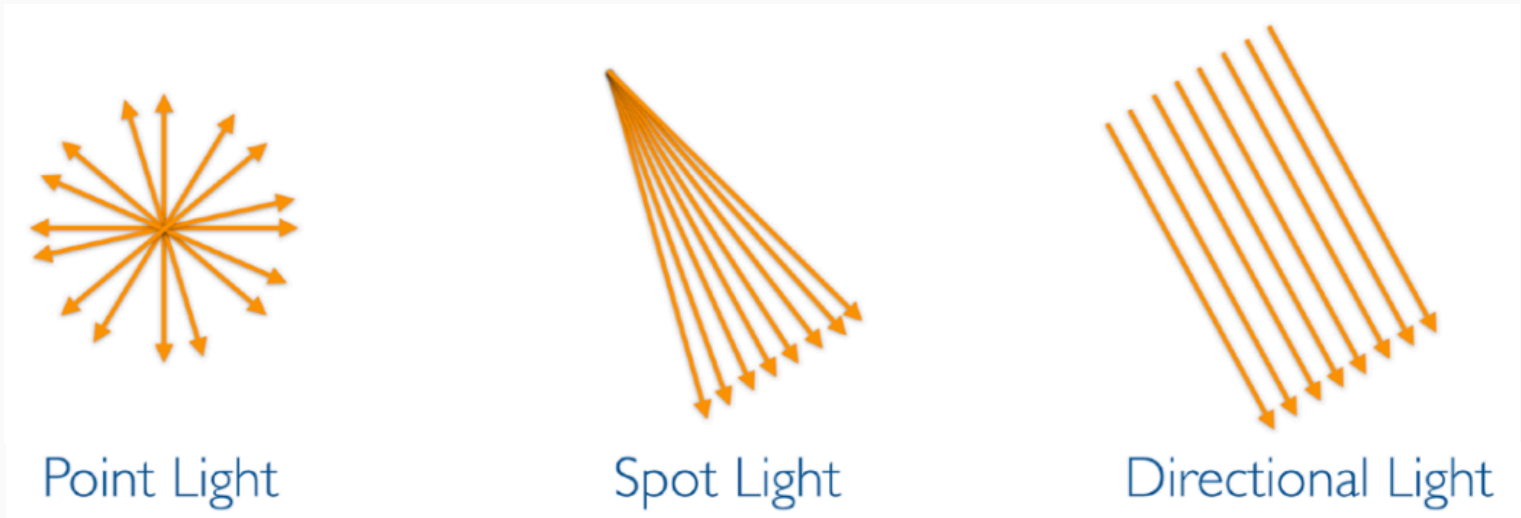
Illumination Model

Illumination Model

Illumination model, also known as Shading model or Lighting model, is used to calculate the intensity of light that is reflected at a given point on surface. There are three factors on which lighting effect depends on:

. **Light Source :** Light source is the light emitting source. There are three types of light sources: Their position, electromagnetic spectrum and shape determine the lighting effect.

1. **Point Sources –** The source that emit rays in all directions (A bulb in a room).
2. **Parallel Sources –** Can be considered as a point source which is far from the surface (The sun).
3. **Distributed Sources –** Rays originate from a finite area (A tube light).



. **Surface :** When light falls on a surface part of it is reflected and part of it is absorbed. Now the surface structure decides the amount of reflection and absorption of light. The position of the surface and positions of all the nearby surfaces also determine the lighting effect.

. **Observer :** The observer's position and sensor spectrum sensitivities also affect the lighting effect

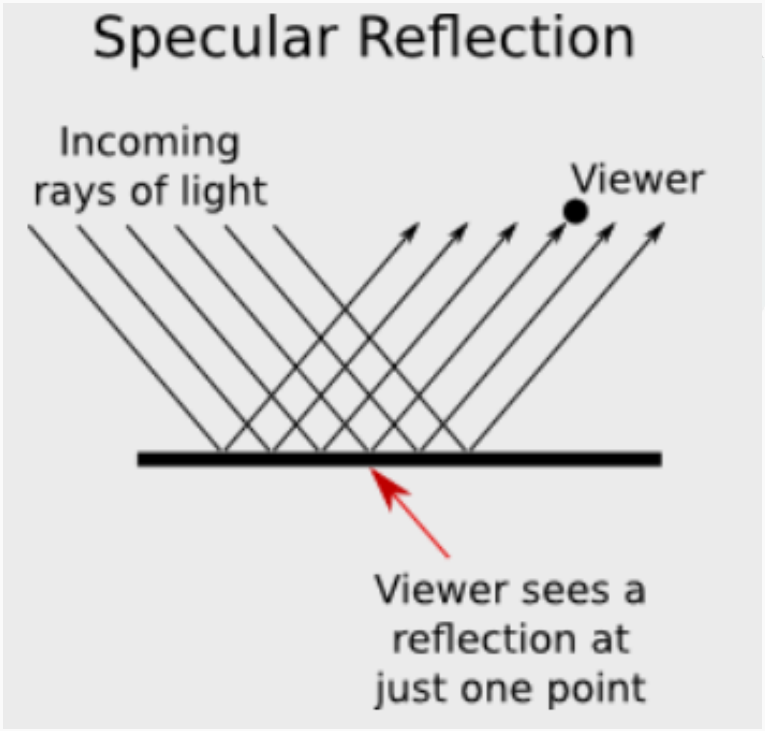
Diffuse Reflection:

- Surfaces that are rough, or grainy, tend to scatter the reflected light in all directions. ***This scattered light is called defuse reflection.***
- Diffuse reflection is the scattering of light in different directions when it encounters a rough or irregular surface.
- Unlike ***specular reflection***, which occurs on smooth surfaces and results in a clear reflection, diffuse reflection causes light to be scattered in various angles, resulting in a softer and less distinct reflection.
- Diffuse reflection occurs when light strikes a surface with microscopic irregularities or roughness.
- These imperfections cause the light to bounce off in different directions instead of reflecting at a single angle.
- As a result, the light is scattered and diffused, creating a more even distribution of reflected light.

Diffuse Reflection:

- Rough surfaces have many different angles that they can reflect light off of, whereas smooth surfaces have fewer angles.
- The color of a diffusely reflecting object depends on the wavelength of the incoming light and the wavelength-dependent absorption coefficient of the material.
- For example, a red object will appear red because it absorbs all wavelengths of light except for red (which it then reflects).
- Applications: In photography, diffusers are often used to soften and distribute light evenly, reducing harsh shadows. Additionally, display screens with anti-glare coatings rely on diffuse reflection to minimize reflections and enhance visibility. Optical sensors and laser scanners also utilize diffuse reflection to detect objects and measure distances accurately.

Diffuse Reflection:



Types of Diffuse Reflection

- **Lambertian reflection** is the simplest type of diffuse reflection. It assumes that the surface is perfectly diffuse, meaning that the angle at which light hits the surface has no effect on how much light is reflected. This results in a smooth, even reflection without any highlights or shadows.

- **Oren-Nayar reflection** is a more realistic type of diffuse reflection that takes into account the fact that light doesn't always hit a surface evenly. It accounts for both the angle at which light hits the surface and the roughness of the surface. This results in a more natural-looking reflection with highlights and shadows.

- **Phong reflection** is the most realistic type of diffuse reflection. It takes into account not only the angle at which light hits the surface and the roughness of the surface but also the shininess of the surface. This results in a very natural-looking reflection with highlights, shadows, and specular reflections (the bright highlights you see on polished surfaces).

Color Models

The color spaces in image processing aim to facilitate the specifications of colors in some standard way.

Different types of color models are used in multiple fields like in hardware, in multiple applications of creating animation, etc.

Let's see each color model and its application.

- RGB
- CMYK
- HSV
- YIQ

RGB Color Model

The RGB Color Model is the most common color model used in Digital Image Processing and OpenCV (Open-Source Computer Vision Library) is a free and open-source library primarily used for computer vision, machine learning, and image processing, offering a wide range of algorithms and functionalities for real-time image and video analysis).

The color image consists of 3 channels. One channel each for one color. Red, Green and Blue are the main color components of this model. All other colors are produced by the proportional ratio of these three colors only. 0 represents the black and as the value increases the color intensity increases.

Properties:

- This is an additive color model. The colors are added to the Red, Green and Blue.
- 3 main channels: Red, Green and Blue.
- Used in Digital Image Processing, OpenCV and online logos.

RGB Color Model

Colour combination:

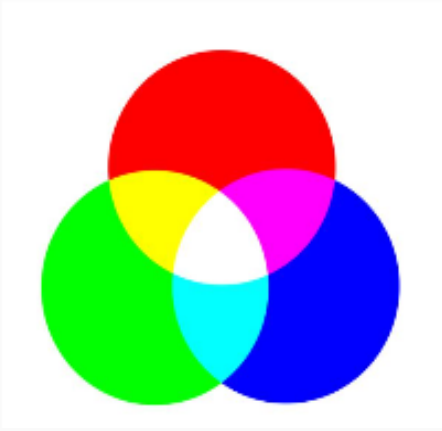
Green(255) + Red(255) = Yellow

Green(255) + Blue(255) = Cyan

Red(255) + Blue(255) = Magenta

Red(255) + Green(255) + Blue(255) = White

RGB - Red Green Blue

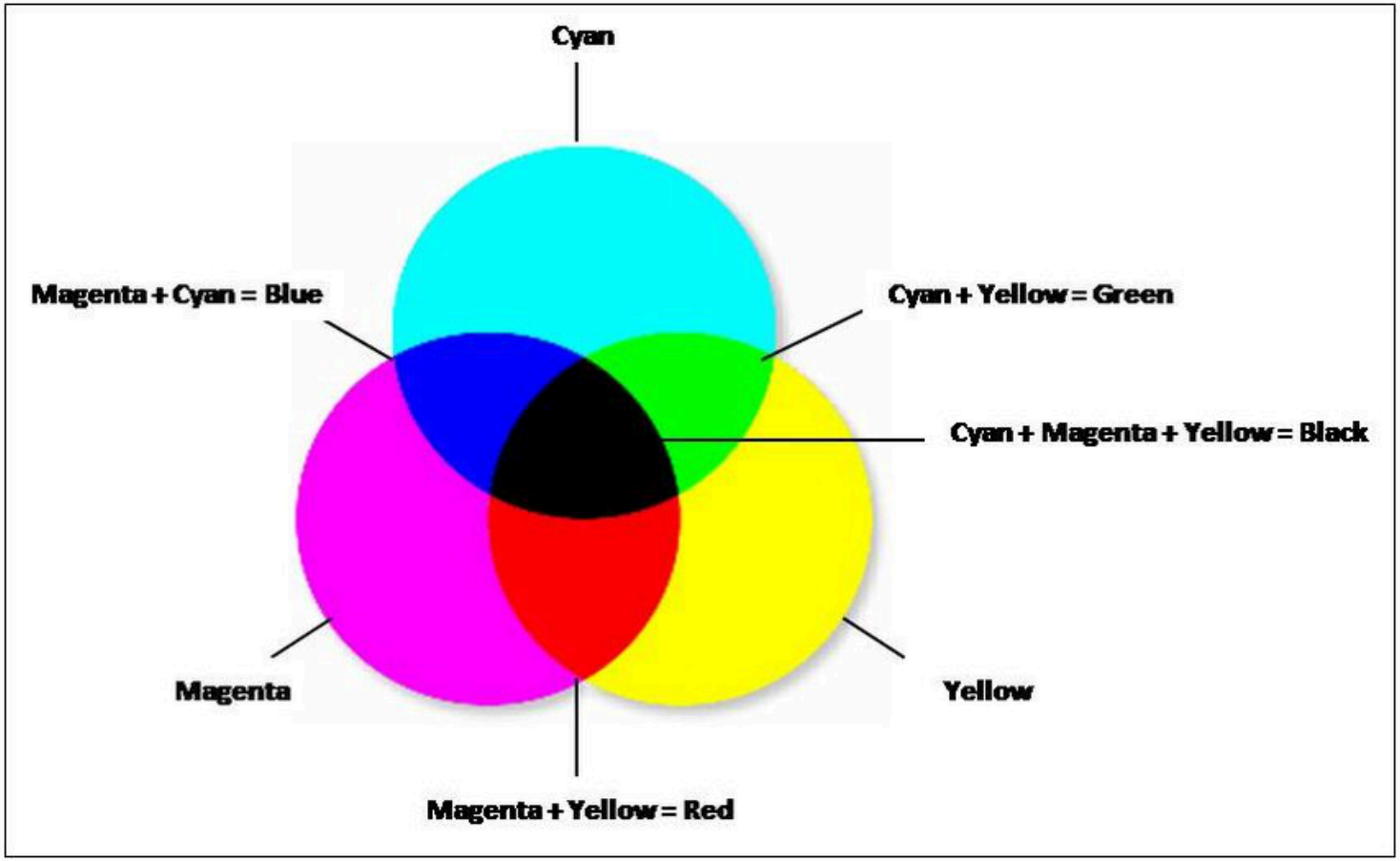


CMYK Color Model:

- CMYK color model is widely used in printers.
- It stands for Cyan, Magenta, Yellow and Black (key).
- It is a subtractive color model. 0 represents the primary color and 1 represents the lightest color.
- In this model, point (1, 1, 1) represents black, and (0,0,0) represents white.
- It is a ***subtractive model*** thus the value is subtracted from 1 to vary from least intense to a most intense color value.

```
1-RGB = CMY
Cyan is negative of Red.
Magenta is negative of Green.
Yellow is negative of Blue.
```

CMYK Color Model:



HSV Color Model:

HSV: The image consists of three channels. Hue, Saturation and Value are three channels. This color model does not use primary colors directly. It uses color in the way humans perceive them. HSV color when is represented by a cone.

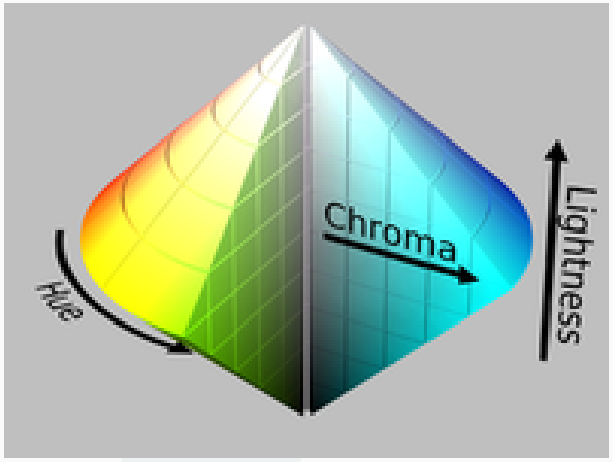
Hue is a color component. Since the cone represents the HSV model, the hue represents different colors in different angle ranges.

```
Red colour falls between 0 and 60 degrees in the HSV cone.
Yellow colour falls between 61 and 120 degrees in the HSV cone.
Green colour falls between 121 and 180 degrees in the HSV cone.
Cyan colour falls between 181 and 240 degrees in the HSV cone.
Blue colour falls between 241 and 300 degrees in the HSV cone.
Magenta colour falls between 301 and 360 degrees in the HSV cone.
```

HSV Color Model:

Saturation as the name suggest describes the percentage of the color. Sometimes this value lies in the 0 to 1 range. 0 being the grey and 1 being the primary color. Saturation describes the grey color.

The value represents the intensity of the color chosen. Its value lies in percentage from 0 to 100. 0 is black and 100 is the brightest and reveals the color.



HSV model is used in histogram equalization and converting grayscale images to RGB Color Images.

YIQ (Luminance, In-phase, Quadrature) Color Model:

- YIQ is the most widely used color model in ***Television broadcasting***.
- Y stands for luminance part and IQ stands for chrominance part.
- luminance refers to the brightness or intensity of an image, while chrominance describes the color information, often represented as the difference between the color and a grayscale version of the image.
- In the black and white television, only the luminance part (Y) was broadcast.
- The Y value is similar to the grayscale part. The color information is represented by the IQ part.
- There exist a formula to convert RGB into YIQ and vice-versa.

YIQ Color Model:

- In-Phase = Red – Yellow
- Quadrature = Blue – Yellow
- It is not possible to directly display a YIQ image while developing.
- The show function only recognizes RGB colors.
- If you try to display an image in another color space, the show function will display the wrong color.
- To use show, we have to use conversions.

- **From YIQ to RGB conversion:**

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.619 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

- **From RGB to YIQ conversion:**

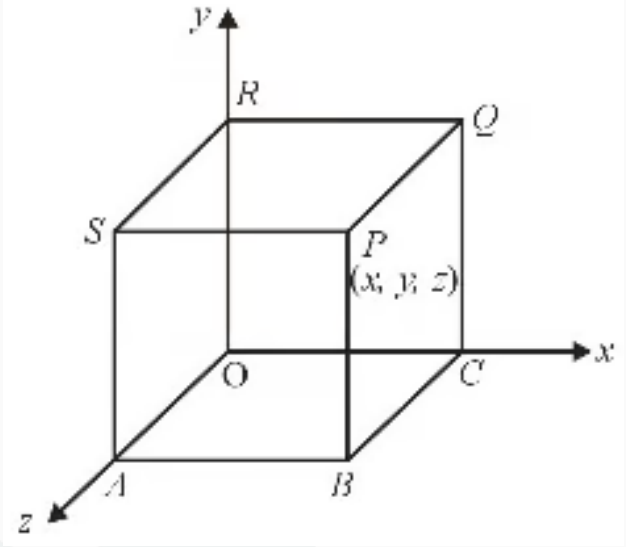
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3D Viewing and Projections

- **Three-dimensional (3D) viewing** in computer graphics involves the representation and visualization of three-dimensional objects on a two-dimensional display.
- The process of 3D viewing encompasses various techniques such as projections to accurately portray the spatial relationships of objects in a virtual environment.
- **Projections** are a crucial aspect of 3D viewing as they transform three-dimensional points onto a two-dimensional plane.
- Different projection methods are used to simulate how objects appear when viewed from a particular perspective or angle.
- The two main types of projections are parallel projections and perspective projections.

3D Geometry

- Three-dimension system has three axis x, y, z. The orientation of a 3D coordinate system is of two types. Right-handed system and left-handed system.
- In the right -handed system thumb of right-hand points to positive z-direction and left-hand system thumb point to negative two directions. Following figure show right-hand orientation of the cube.
- Using right-handed system co-ordinates of corners: A, B, C, O and P, Q, R, S of the cube



A: (0, 0, z)

B: (x, 0, z)

C: (x, 0, 0)

O: (0, 0, 0)

P: (x, y, z)

Q: (x, y, 0)

R: (0, y, 0)

S: (0, y, z)

3D Transformation

- In Computer graphics, Transformation is a process of modifying and re-positioning the existing graphics.
- 3D Transformations take place in a three-dimensional plane.
- 3D Transformations are bit more complex than 2D Transformations.
- Transformations are helpful in changing the position, size, orientation, shape etc., of the object.
    1. Translation
    2. Scaling
    3. Rotation
    4. Reflection
    5. Shear

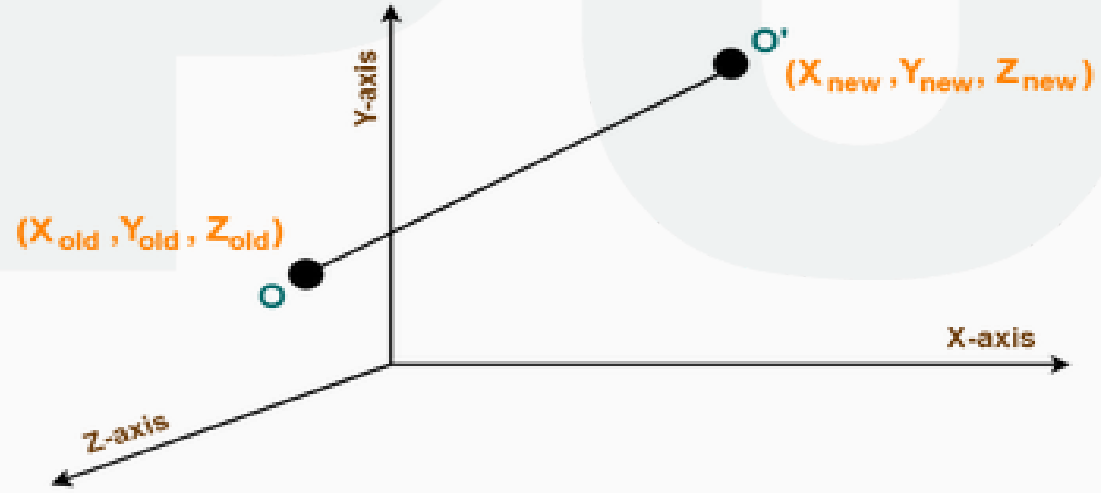
3D Translation

- In Computer graphics, 3D Translation is a process of moving an object from one position to another in a three-dimensional plane.
- Consider a point object O has to be moved from one position to another in a 3D plane.
- Let-
- Initial coordinates of the object O = $(X_{old}, Y_{old}, Z_{old})$
- New coordinates of the object O after translation = $(X_{new}, Y_{new}, Z_{new})$
- Translation vector or Shift vector = (T_x, T_y, T_z)

3D Translation

Given a Translation vector (T_x, T_y, T_z)-

- T_x defines the distance the X_{old} coordinate has to be moved.
- T_y defines the distance the Y_{old} coordinate has to be moved.
- T_z defines the distance the Z_{old} coordinate has to be moved.



3D Translation

This translation is achieved by adding the translation coordinates to the old coordinates of the object as-

- $X_{new} = X_{old} + T_x$ (This denotes translation towards X axis)
- $Y_{new} = Y_{old} + T_y$ (This denotes translation towards Y axis)
- $Z_{new} = Z_{old} + T_z$ (This denotes translation towards Z axis)

In Matrix form, the above translation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Translation

Problem-1:

Given a 3D object with coordinate points A(0, 3, 1), B(3, 3, 2), C(3, 0, 0), D(0, 0, 0). Apply the translation with the distance 1 towards X axis, 1 towards Y axis and 2 towards Z axis and obtain the new coordinates of the object.

Problem-2:

A point A(2,3,5), B(2, 3, 5), C(2,3,5) is translated by $T_A(4,-2,3)$, $T_B(4, -2, 3)$, $T_C(4,-2,3)$. Find the new coordinates.

Problem-3:

A 3D object is positioned at (2, 5, 7). It needs to be shifted by 10 units in the X-direction, -4 in the Y-direction, and 3 in the Z-direction. What will be the new position?

Problem-4:

A triangle has vertices at A(1,2,3), B(4,5,6), C(7,8,9). Translate it by $T_A(3,-3,2)$ $T_B(3, -3, 2)$ $T_C(3,-3,2)$.

3D Rotation

In Computer graphics,3D Rotation is a process of rotating an object with respect to an angle in a three-dimensional plane.

Consider a point object O has to be rotated from one angle to another in a 3D plane.

Let-

- Initial coordinates of the object O = $(X_{old}, Y_{old}, Z_{old})$
- Initial angle of the object O with respect to origin = Φ
- Rotation angle = θ

New coordinates of the object O after rotation = $(X_{new}, Y_{new}, Z_{new})$

In 3 dimensions, there are 3 possible types of rotation-

- X-axis Rotation
- Y-axis Rotation
- Z-axis Rotation

Rotation respect to X-Axis

This rotation is achieved by using the following rotation equations-

- $X_{new} = X_{old}$
- $Y_{new} = Y_{old} \times \cos\theta - Z_{old} \times \sin\theta$
- $Z_{new} = Y_{old} \times \sin\theta + Z_{old} \times \cos\theta$
- In Matrix form, the above rotation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Rotation respect to Y-Axis

This rotation is achieved by using the following rotation equations-

- $X_{new} = Z_{old} \times \sin\theta + X_{old} \times \cos\theta$
- $Y_{new} = Y_{old}$
- $Z_{new} = Y_{old} \times \cos\theta - X_{old} \times \sin\theta$

In Matrix form, the above rotation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Rotation respect to Z-Axis

This rotation is achieved by using the following rotation equations-

- $X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta$
- $Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta$
- $Z_{new} = Z_{old}$

In Matrix form, the above rotation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Rotation Example

Problem-01: Given a homogeneous point (1, 2, 3). Apply rotation 90 degree towards X, Y and Z axis and find out the new coordinate points.

- $X_{new} = X_{old} = 1$
- $Y_{new} = Y_{old} \times \cos\theta - Z_{old} \times \sin\theta = 2 \times \cos90° - 3 \times \sin90° = 2 \times 0 - 3 \times 1 = -3$
- $Z_{new} = Y_{old} \times \sin\theta + Z_{old} \times \cos\theta = 2 \times \sin90° + 3 \times \cos90° = 2 \times 1 + 3 \times 0 = 2$

Solution-

- Old coordinates = $(X_{old}, Y_{old}, Z_{old}) = (1, 2, 3)$
- Rotation angle = $\theta = 90º$

For X-Axis Rotation-

Let the new coordinates after rotation = $(X_{new}, Y_{new}, Z_{new})$.

Applying the rotation equations, we have-

Thus, New coordinates after rotation = (1, -3, 2).

Rotation Example

For Y-Axis Rotation-

Let the new coordinates after rotation = $(X_{new}, Y_{new}, Z_{new})$.

Applying the rotation equations, we have-

- $X_{new} = Z_{old} \times \sin\theta + X_{old} \times \cos\theta = 3 \times \sin90° + 1 \times \cos90° = 3 \times 1 + 1 \times 0 = 3$
- $Y_{new} = Y_{old} = 2$
- $Z_{new} = Y_{old} \times \cos\theta - X_{old} \times \sin\theta = 2 \times \cos90° - 1 \times \sin90° = 2 \times 0 - 1 \times 1 = -1$

Thus, New coordinates after rotation = (3, 2, -1).

Rotation Example

For Z-Axis Rotation-

Let the new coordinates after rotation = $(X_{new}, Y_{new}, Z_{new})$.

Applying the rotation equations, we have-

- $X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta = 1 \times \cos90° - 2 \times \sin90° = 1 \times 0 - 2 \times 1 = -2$
- $Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta = 1 \times \sin90° + 2 \times \cos90° = 1 \times 1 + 2 \times 0 = 1$
- $Z_{new} = Z_{old} = 3$

Thus, New coordinates after rotation = (-2, 1, 3).

3D Scaling

In computer graphics, scaling is a process of modifying or altering the size of objects.

- Scaling may be used to increase or reduce the size of object.
- Scaling subjects the coordinate points of the original object to change.
- Scaling factor determines whether the object size is to be increased or reduced.
- If scaling factor > 1, then the object size is increased.
- If scaling factor < 1, then the object size is reduced.

Consider a point object O has to be scaled in a 3D plane.

Let-

- Initial coordinates of the object O = $(X_{old}, Y_{old}, Z_{old})$
- Scaling factor for X-axis = S_x
- Scaling factor for Y-axis = S_y
- Scaling factor for Z-axis = S_z

3D Scaling

New coordinates of the object O after scaling = $(X_{new}, Y_{new}, Z_{new})$

This scaling is achieved by using the following scaling equations-

- $X_{new} = X_{old} \times S_x$
- $Y_{new} = Y_{old} \times S_y$
- $Z_{new} = Z_{old} \times S_z$

In Matrix form, the above scaling equations may be represented as-

3D Scaling Example

Problem-01: Given a 3D object with coordinate points A(0, 3, 3), B(3, 3, 6), C(3, 0, 1), D(0, 0, 0). Apply the scaling parameter 2 towards X axis, 3 towards Y axis and 3 towards Z axis and obtain the new coordinates of the object.

Solution-

Given-

- Old coordinates of the object =

  A (0, 3, 3), B(3, 3, 6), C(3, 0, 1), D(0, 0, 0)
  - Scaling factor along X axis = 2
  - Scaling factor along Y axis = 3
  - Scaling factor along Z axis = 3

For Coordinates A(0, 3, 3)

Let the new coordinates of A after scaling = $(X_{new}, Y_{new}, Z_{new})$.

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 0 \times 2 = 0$
- $Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$
- $Z_{new} = Z_{old} \times S_z = 3 \times 3 = 9$

Thus, New coordinates of corner A after scaling = (0, 9, 9).

3D Scaling Example

For Coordinates B(3, 3, 6)

Let the new coordinates of B after scaling = $(X_{new}, Y_{new}, Z_{new})$.

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 3 \times 2 = 6$
- $Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$
- $Z_{new} = Z_{old} \times S_z = 6 \times 3 = 18$

Thus, New coordinates of corner B after scaling = (6, 9, 18).

For Coordinates C(3, 0, 1)

Let the new coordinates of C after scaling = $(X_{new}, Y_{new}, Z_{new})$.

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 3 \times 2 = 6$
- $Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$
- $Z_{new} = Z_{old} \times S_z = 1 \times 3 = 3$

Thus, New coordinates of corner C after scaling = (6, 0, 3).

3D Scaling Example

For Coordinates D(0, 0, 0)

Let the new coordinates of D after scaling = (X_{new}, Y_{new}, Z_{new}).

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 0 \times 2 = 0$
- $Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$
- $Z_{new} = Z_{old} \times S_z = 0 \times 3 = 0$

Thus, New coordinates of corner D after scaling = (0, 0, 0).

3D Reflection

- Reflection is a kind of rotation where the angle of rotation is 180 degree.
- The reflected object is always formed on the other side of mirror.
- The size of reflected object is same as the size of original object.

Consider a point object O has to be reflected in a 3D plane.

Let-

- Initial coordinates of the object O = $(X_{old}, Y_{old}, Z_{old})$
- New coordinates of the reflected object O after reflection = $(X_{new}, Y_{new}, Z_{new})$

3D Reflection Types

In 3 dimensions, there are 3 possible types of reflection-

Reflection Relative to XY Plane:

This reflection is achieved by using the following reflection equations-

- $X_{new} = X_{old}$
- $Y_{new} = Y_{old}$
- $Z_{new} = -Z_{old}$

In Matrix form, the above reflection equations may be represented as-

Reflection Relative to YZ Plane

This reflection is achieved by using the following reflection equations-

- $X_{new} = -X_{old}$
- $Y_{new} = Y_{old}$
- $Z_{new} = Z_{old}$

In Matrix form, the above reflection equations may be represented as-

Reflection Relative to XZ Plane

This reflection is achieved by using the following reflection equations-

- $X_{new} = X_{old}$
- $Y_{new} = -Y_{old}$
- $Z_{new} = Z_{old}$

In Matrix form, the above reflection equations may be represented as-

Reflection Example

Given a 3D triangle with coordinate points A(3, 4, 1), B(6, 4, 2), C(5, 6, 3). Apply the reflection on the XY plane and find out the new coordinates of the object.

Solution-

- Old corner coordinates of the triangle = A (3, 4, 1), B(6, 4, 2), C(5, 6, 3)
- Reflection has to be taken on the XY plane

For Coordinates A(3, 4, 1)

Let the new coordinates of corner A after reflection = $(X_{new}, Y_{new}, Z_{new})$.

- $X_{new} = X_{old} = 3$
- $Y_{new} = Y_{old} = 4$
- $Z_{new} = -Z_{old} = -1$

Thus, New coordinates of corner A after reflection = (3, 4, -1)

Reflection Example

For Coordinates B(6, 4, 2)

Let the new coordinates of corner B after reflection = (X_{new}, Y_{new}, Z_{new}).

Applying the reflection equations, we have-

- $X_{new} = X_{old} = 6$
- $Y_{new} = Y_{old} = 4$
- $Z_{new} = -Z_{old} = -2$

Thus, New coordinates of corner B after reflection = (6, 4, -2).

For Coordinates C(5, 6, 3)

Applying the reflection equations, we have-

- $X_{new} = X_{old} = 5$
- $Y_{new} = Y_{old} = 6$
- $Z_{new} = -Z_{old} = -3$

Thus, New coordinates of corner C after reflection = (5, 6, -3).

Thus, New coordinates of the triangle after reflection = A (3, 4, -1), B(6, 4, -2), C(5, 6, -3).

Problem-02:

Given a 3D triangle with coordinate points A(3, 4, 1), B(6, 4, 2), C(5, 6, 3). Apply the reflection on the XZ plane and find out the new coordinates of the object.

3D Shearing

In Computer graphics,3D Shearing is an ideal technique to change the shape of an existing object in a three-dimensional plane.

In a three-dimensional plane, the object size can be changed along X direction, Y direction as well as Z direction.

So, there are three versions of shearing-

3D Shearing

Consider a point object O has to be sheared in a 3D plane.

Let-

- Initial coordinates of the object O = $(X_{old}, Y_{old}, Z_{old})$
- Shearing parameter towards X direction = Sh_x
- Shearing parameter towards Y direction = Sh_y
- Shearing parameter towards Z direction = Sh_z
- New coordinates of the object O after shearing = $(X_{new}, Y_{new}, Z_{new})$

3D Shearing in X Axis

Shearing in X axis is achieved by using the following shearing equations-

- $X_{new} = X_{old}$
- $Y_{new} = Y_{old} + Sh_y \times X_{old}$
- $Z_{new} = Z_{old} + Sh_z \times X_{old}$

In Matrix form, the above shearing equations may be represented as-

3D Shearing in Y Axis

Shearing in Y axis is achieved by using the following shearing equations-

- $X_{new} = X_{old} + Sh_x \times Y_{old}$
- $Y_{new} = Y_{old}$
- $Z_{new} = Z_{old} + Sh_z \times Y_{old}$

In Matrix form, the above shearing equations may be represented as-

3D Shearing in Z Axis

Shearing in Z axis is achieved by using the following shearing equations-

- $X_{new} = X_{old} + Sh_x \times Z_{old}$
- $Y_{new} = Y_{old} + Sh_y \times Z_{old}$
- $Z_{new} = Z_{old}$

In Matrix form, the above shearing equations may be represented as-

3D Shearing Example

Given a 3D triangle with points (0, 0, 0), (1, 1, 2) and (1, 1, 3). Apply shear parameter 2 on X axis, 2 on Y axis and 3 on Z axis and find out the new coordinates of the object.

Solution-

- Old corner coordinates of the triangle = A (0, 0, 0), B(1, 1, 2), C(1, 1, 3)
- Shearing parameter towards X direction (Sh_x) = 2
- Shearing parameter towards Y direction (Sh_y) = 2
- Shearing parameter towards Y direction (Sh_z) = 3

3D Shearing Example (Respect to X Axis)

For Coordinates A(0, 0, 0)

Let the new coordinates of corner A after shearing = $(X_{new}, Y_{new}, Z_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} = 0$
- $Y_{new} = Y_{old} + Sh_y \times X_{old} = 0 + 2 \times 0 = 0$
- $Z_{new} = Z_{old} + Sh_z \times X_{old} = 0 + 3 \times 0 = 0$

Thus, New coordinates of corner A after shearing = (0, 0, 0).

For Coordinates B(1, 1, 2)

Let the new coordinates of corner B after shearing = $(X_{new}, Y_{new}, Z_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} = 1$
- $Y_{new} = Y_{old} + Sh_y \times X_{old} = 1 + 2 \times 1 = 3$
- $Z_{new} = Z_{old} + Sh_z \times X_{old} = 2 + 3 \times 1 = 5$

Thus, New coordinates of corner B after shearing = (1, 3, 5).

3D Shearing Example (Respect to X Axis)

For Coordinates C(1, 1, 3)

Let the new coordinates of corner C after shearing = (X_{new}, Y_{new}, Z_{new}).

Applying the shearing equations, we have-

- $X_{new} = X_{old} = 1$
- $Y_{new} = Y_{old} + Sh_y \times X_{old} = 1 + 2 \times 1 = 3$
- $Z_{new} = Z_{old} + Sh_z \times X_{old} = 3 + 3 \times 1 = 6$

Thus, New coordinates of corner C after shearing = (1, 3, 6).

Thus, New coordinates of the triangle after shearing in X axis = A (0, 0, 0), B(1, 3, 5), C(1, 3, 6).

3D Shearing Example (Respect to Y Axis)

For Coordinates A(0, 0, 0)

Let the new coordinates of corner A after shearing = $(X_{new}, Y_{new}, Z_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Y_{old} = 0 + 2 \times 0 = 0$
- $Y_{new} = Y_{old} = 0$
- $Z_{new} = Z_{old} + Sh_z \times Y_{old} = 0 + 3 \times 0 = 0$

Thus, New coordinates of corner A after shearing = (0, 0, 0).

For Coordinates B(1, 1, 2)

Let the new coordinates of corner B after shearing = $(X_{new}, Y_{new}, Z_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Y_{old} = 1 + 2 \times 1 = 3$
- $Y_{new} = Y_{old} = 1$
- $Z_{new} = Z_{old} + Sh_z \times Y_{old} = 2 + 3 \times 1 = 5$

Thus, New coordinates of corner B after shearing = (3, 1, 5).

3D Shearing Example (Respect to Y Axis)

For Coordinates C(1, 1, 3)

Let the new coordinates of corner C after shearing = (X_{new}, Y_{new}, Z_{new}).

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Y_{old} = 1 + 2 \times 1 = 3$
- $Y_{new} = Y_{old} = 1$
- $Z_{new} = Z_{old} + Sh_z \times Y_{old} = 3 + 3 \times 1 = 6$

Thus, New coordinates of corner C after shearing = (3, 1, 6).

Thus, New coordinates of the triangle after shearing in Y axis = A (0, 0, 0), B(3, 1, 5), C(3, 1, 6).

3D Shearing Example (Respect to Z Axis)

For Coordinates A(0, 0, 0)

Let the new coordinates of corner A after shearing = $(X_{new}, Y_{new}, Z_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Z_{old} = 0 + 2 \times 0 = 0$
- $Y_{new} = Y_{old} + Sh_y \times Z_{old} = 0 + 2 \times 0 = 0$
- $Z_{new} = Z_{old} = 0$

Thus, New coordinates of corner A after shearing = (0, 0, 0).

For Coordinates B(1, 1, 2)

Let the new coordinates of corner B after shearing = $(X_{new}, Y_{new}, Z_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Z_{old} = 1 + 2 \times 2 = 5$
- $Y_{new} = Y_{old} + Sh_y \times Z_{old} = 1 + 2 \times 2 = 5$
- $Z_{new} = Z_{old} = 2$

Thus, New coordinates of corner B after shearing = (5, 5, 2).

3D Shearing Example (Respect to Z Axis)

For Coordinates C(1, 1, 3)

Let the new coordinates of corner C after shearing = (X_{new}, Y_{new}, Z_{new}).

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Z_{old} = 1 + 2 \times 3 = 7$
- $Y_{new} = Y_{old} + Sh_y \times Z_{old} = 1 + 2 \times 3 = 7$
- $Z_{new} = Z_{old} = 3$

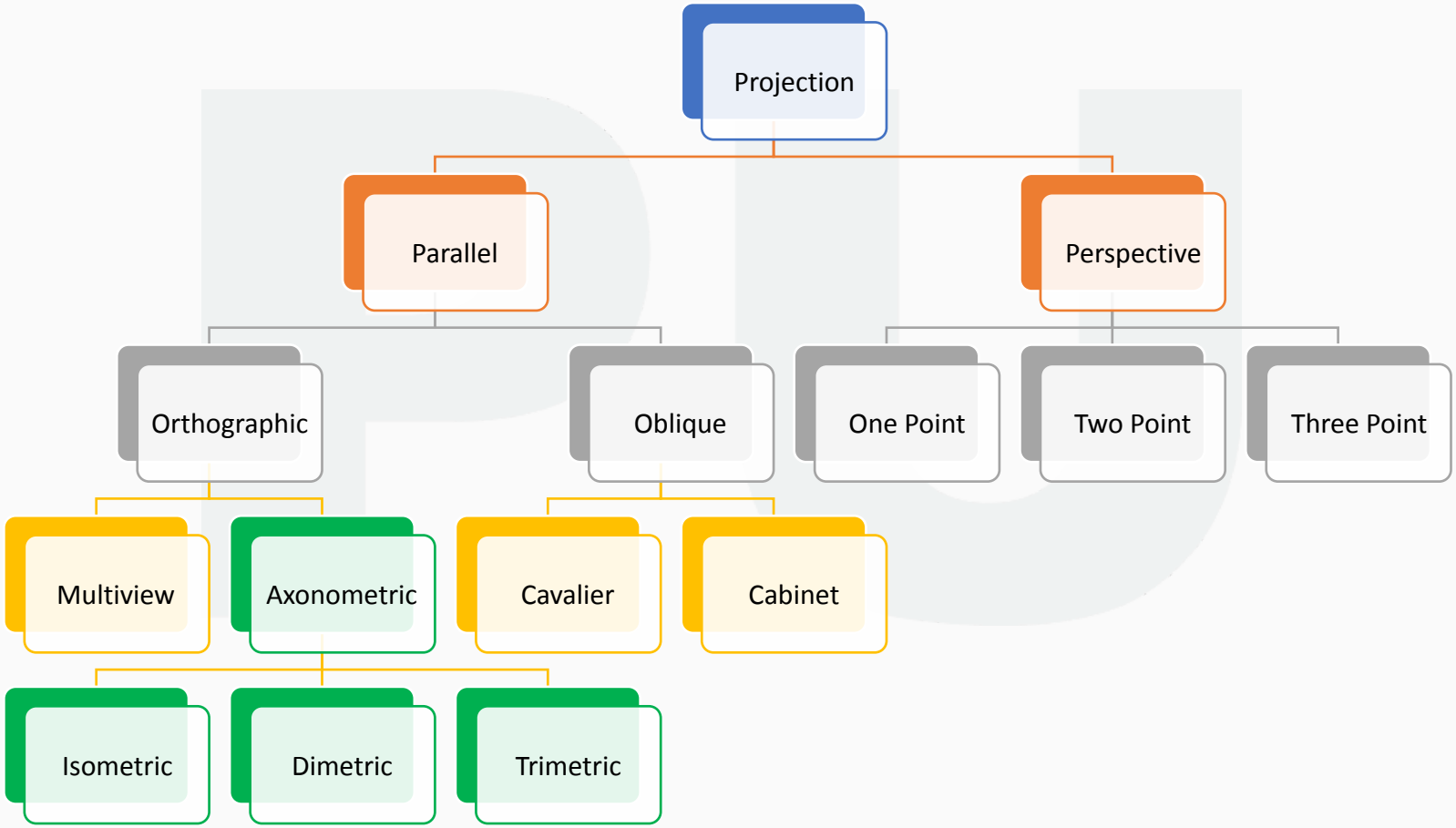
Thus, New coordinates of corner C after shearing = (7, 7, 3).

Thus, New coordinates of the triangle after shearing in Z axis = A (0, 0, 0), B(5, 5, 2), C(7, 7, 3).

Projection

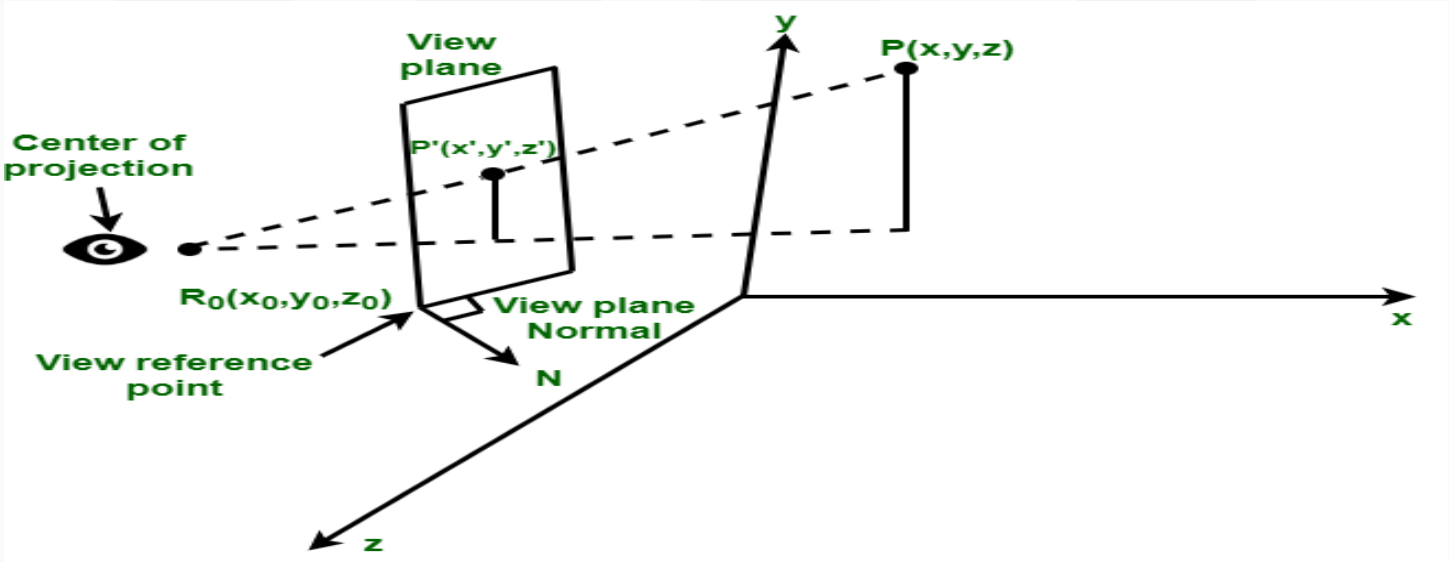
- Projection is a kind of phenomena that are used in computer graphics to map the view of a 3D object onto the projecting display panel where the viewing volume is specified by the world coordinate and then map these world coordinate over the view port.
- It is the process of converting a 3D object into a 2D object.
- It is also defined as mapping or transformation of the object in projection plane or view plane.
- The view plane is displayed surface.

Projection


Projection
Parallel
Perspective
Orthographic
Oblique
One Point
Two Point
Three Point
Multiview
Axonometric
Cavalier
Cabinet
Isometric
Dimetric
Trimetric

Perspective Projection

In Perspective Projection the center of projection is at finite distance from projection plane. This projection produces realistic views but does not preserve relative proportions of an object dimensions. Projections of distant object are smaller than projections of objects of same size that are closer to projection plane. The perspective projection can be easily described by the following figure:



Important terms related to Perspective

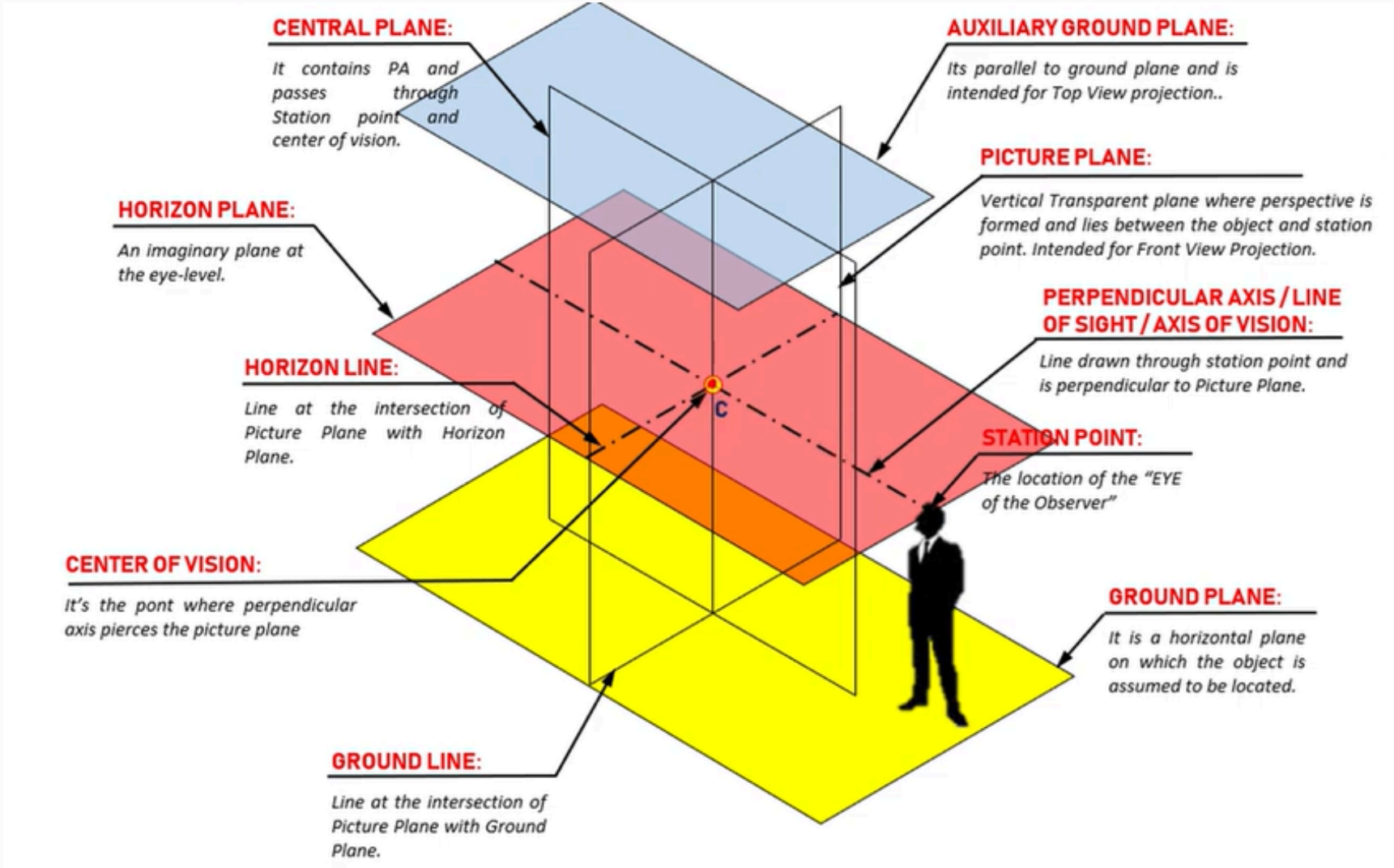
1. **Center of Projection –** It is a point where lines or projection that are not parallel to projection plane appear to meet.

2. **View Plane or Projection Plane –** The view plane is determined by :

    1. View reference point $R_0(x_0, y_0, z_0)$

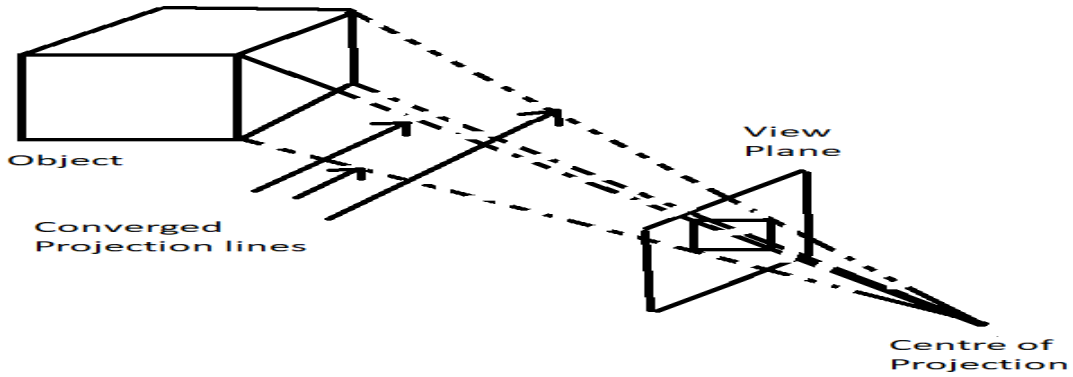
    2. View plane normal.

3. **Location of an Object –** It is specified by a point P that is located in world coordinates at (x, y, z) location. The objective of perspective projection is to determine the image point P' whose coordinates are (x', y', z')


CENTRAL PLANE:
It contains PA and passes through Station point and center of vision.
AUXILIARY GROUND PLANE:
Its parallel to ground plane and is intended for Top View projection..
PICTURE PLANE:
Vertical Transparent plane where perspective is formed and lies between the object and station point. Intended for Front View Projection.
HORIZON PLANE:
An imaginary plane at the eye-level.
PERPENDICULAR AXIS / LINE OF SIGHT / AXIS OF VISION:
Line drawn through station point and is perpendicular to Picture Plane.
HORIZON LINE:
Line at the intersection of Picture Plane with Horizon Plane.
C
STATION POINT:
The location of the "EYE of the Observer"
CENTER OF VISION:
It's the pont where perpendicular axis pierces the picture plane
GROUND PLANE:
It is a horizontal plane on which the object is assumed to be located.
GROUND LINE:
Line at the intersection of Picture Plane with Ground Plane.

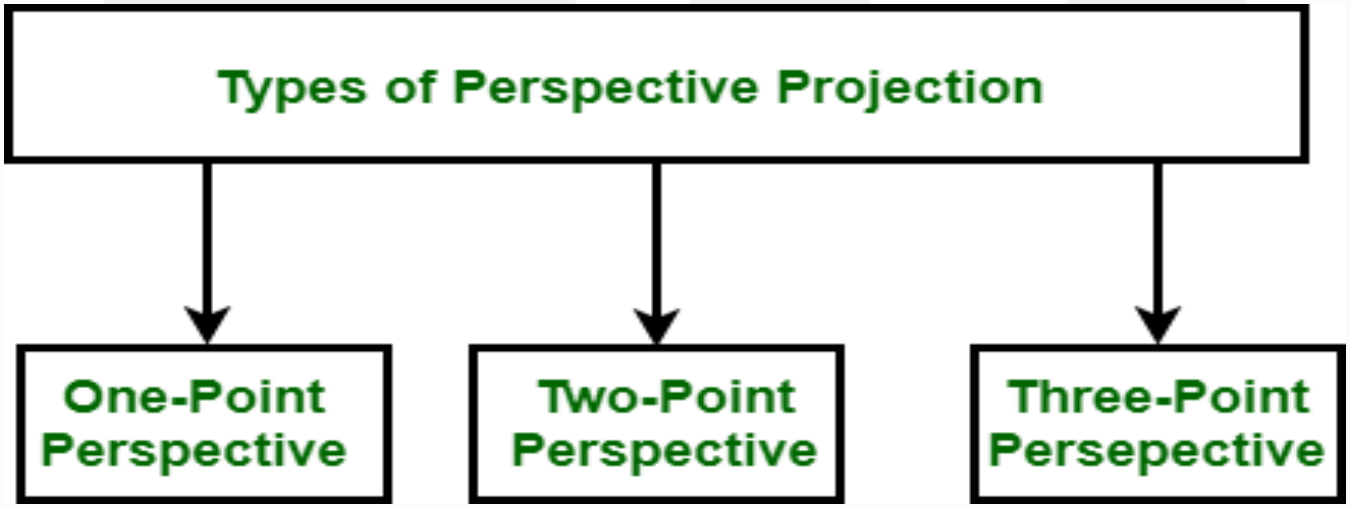
Perspective Projection

- In perspective projection, the lines of projection are not parallel. Instead, they all converge at a single point called the center of projection or projection reference point.
- The object positions are transformed to the view plane along these converged projection lines and the projected view of an object is determined by calculating the intersection of the converged projection lines with the view plane, as shown below figure:



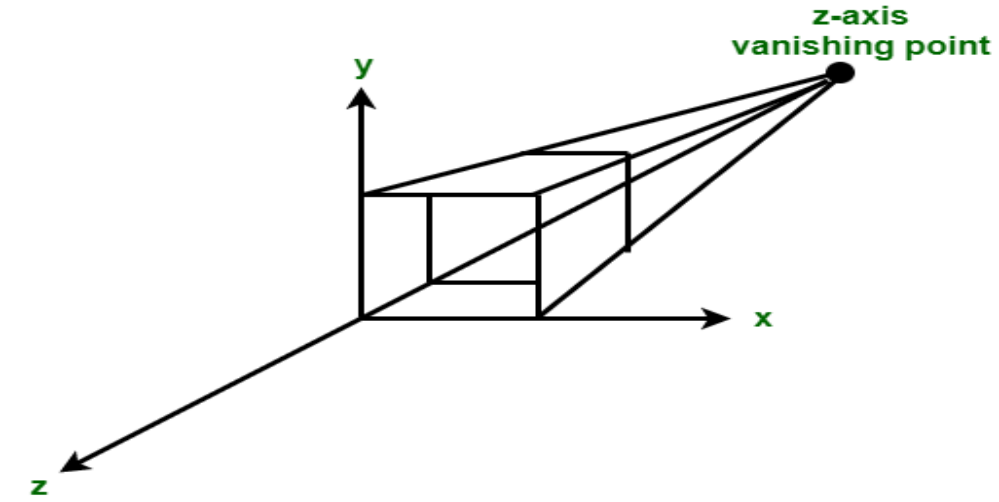
Types of Perspective Projection

- Classification of perspective projection is on basis of **vanishing points** (It is a point in image where a parallel line through center of projection intersects view plane).
- We can say that a vanishing point is a point where projection line intersects view plane. The classification is as follows :



One-Point Perspective Projection

1. **One Point Perspective Projection:** One-point perspective projection occurs when any of principal axes intersects with projection plane or we can say when projection plane is perpendicular to principal axis.

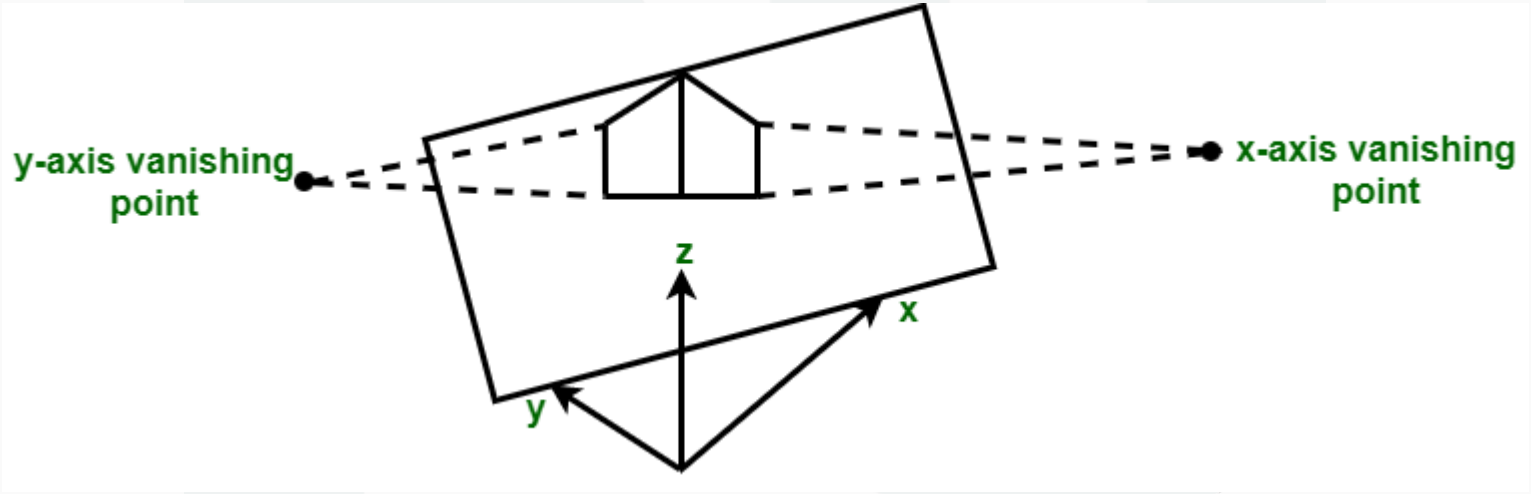


- In the above figure, z axis intersects projection plane whereas x and y axis remain parallel to projection plane.

One-Point Perspective Projection

Two-Point Perspective Projection

2. **Two Point Perspective Projection:** Two-point perspective projection occurs when projection plane intersects two of principal axis.

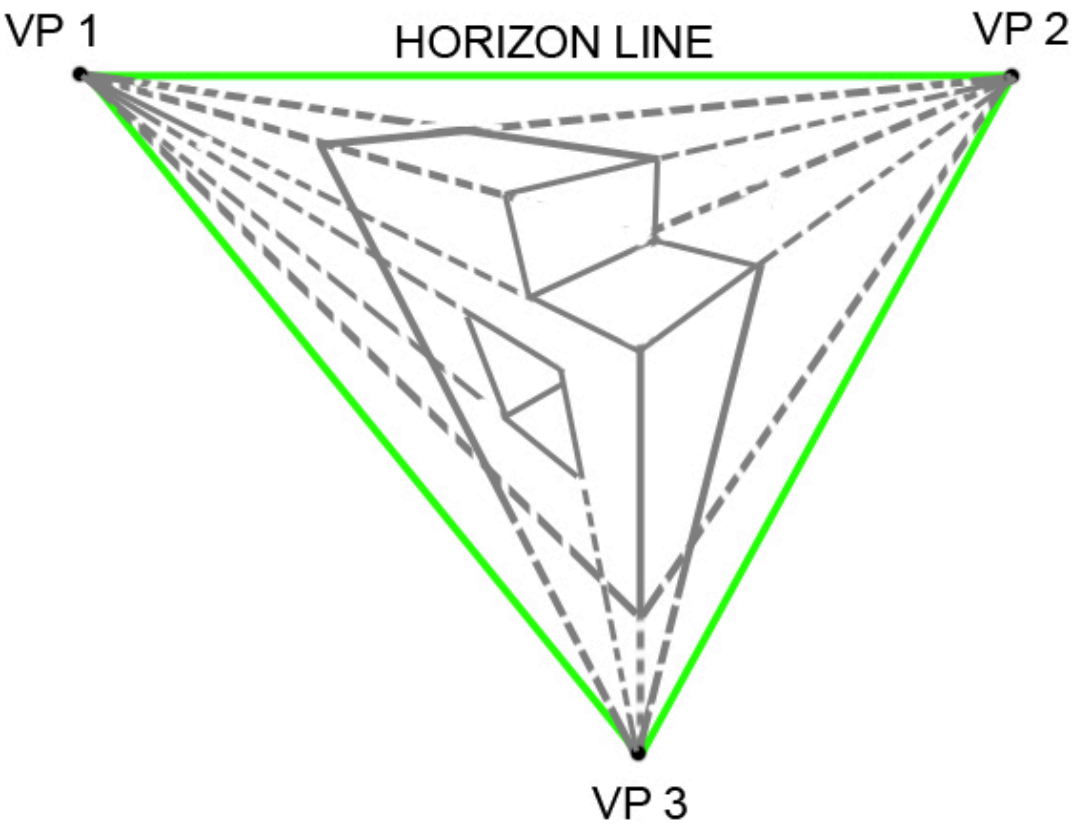


- In the above figure, projection plane intersects x and y axis whereas z axis remains parallel to projection plane.

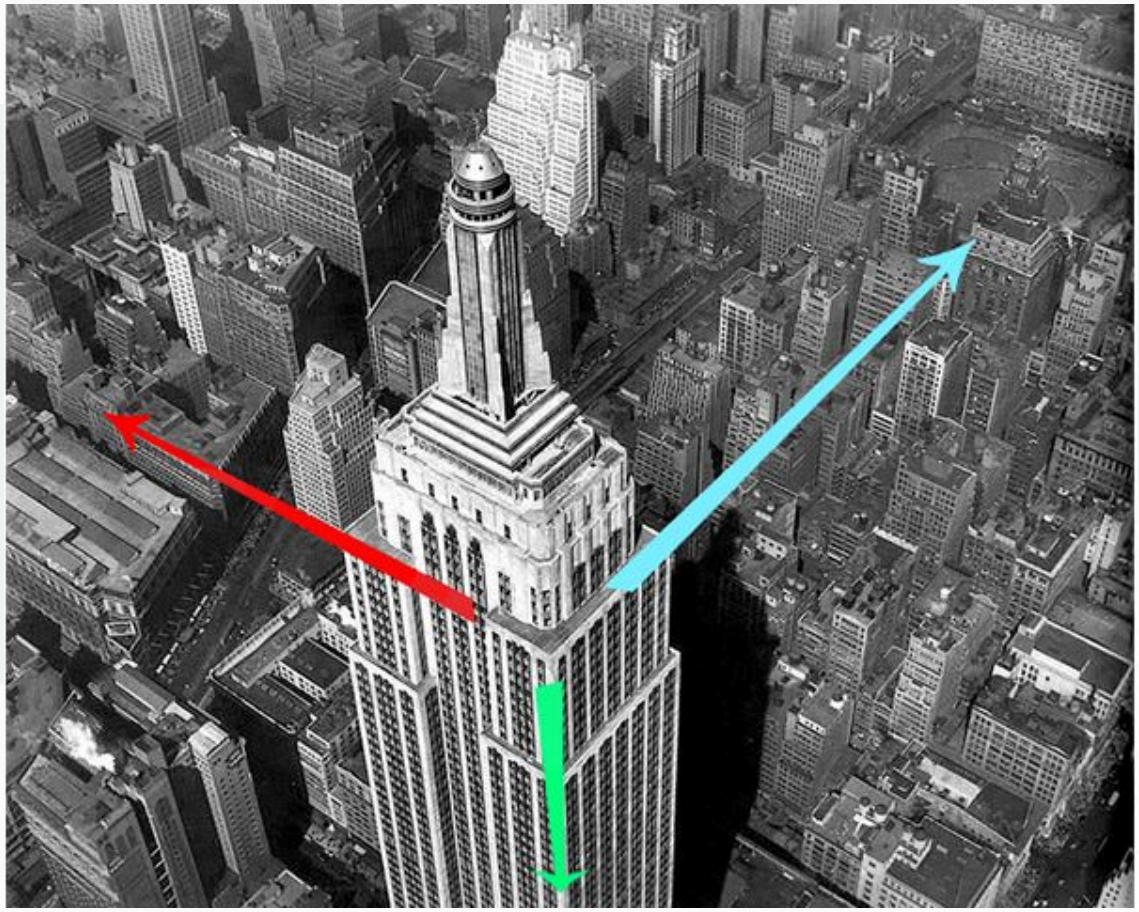
Two-Point Perspective Projection

Three-Point Perspective Projection

3. Three-Point perspective projection occurs when all three axis intersects with projection plane. There is no any principal axis which is parallel to projection plane.



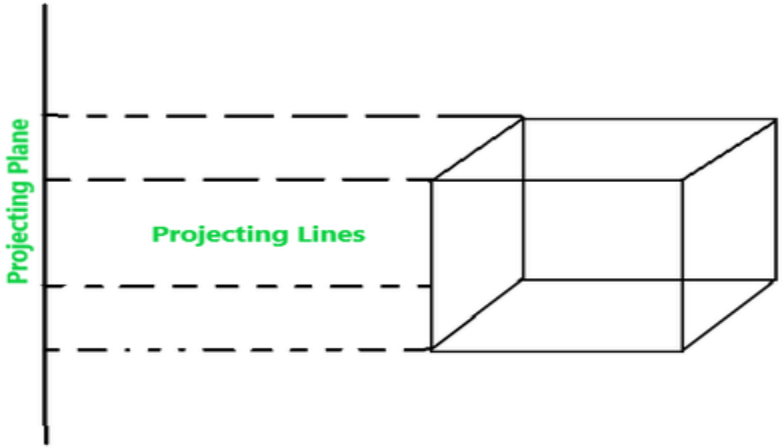
Three-Point Perspective Projection



Parallel Projection

- Parallel projection is a kind of projection where the projecting lines emerge parallelly from the polygon surface and then incident parallelly on the plane.

- In parallel projection, the center of the projection lies at infinity.
- In parallel projection, the view of the object obtained at the plane is less-realistic as there is no for-shortcoming and the relative dimension of the object remains preserves.



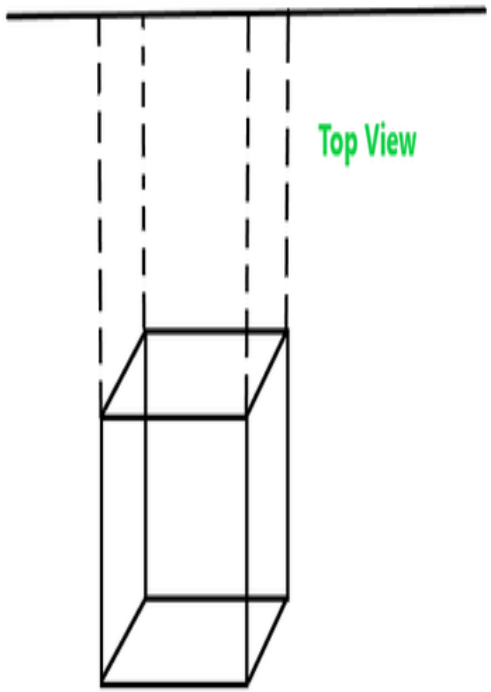
Two Categories of Parallel Projection (1. Orthographic)

1. **Orthographic Projection :** It is a kind of parallel projection where the projecting lines emerge parallelly from the object surface and incident perpendicularly at the projecting plane.

   Orthographic Projection is of two categories :

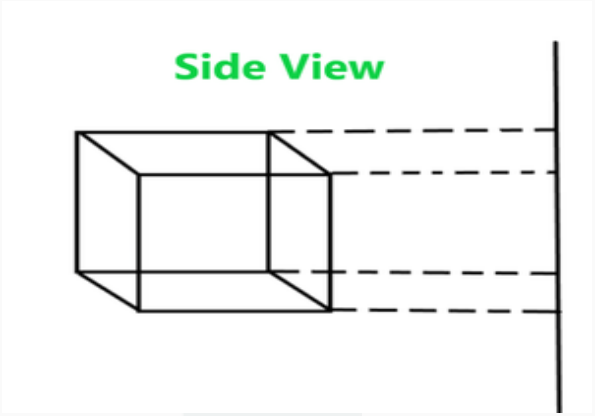
   A. **Multiview Projection:** It is further divided into three categories –

      1) **Top-View :** In this projection, the rays that emerge from the top of the polygon surface are observed.

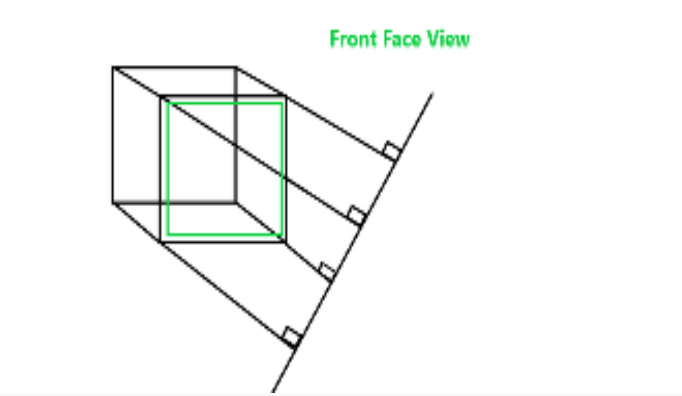


1. Orthographic Projection (1. Multiview)

2) **<u>Side-View</u>:** It is another type of projection orthographic projection where the side view of the polygon surface is observed.



3) **<u>Front-view</u>:** In this orthographic projection front face view of the object is observed.



1. Orthographic Projection (2. Axonometric)

B. **<u>Axonometric</u>:** Axonometric projection is an orthographic projection, where the projection lines are perpendicular to the plane of projection, and the object is rotated around one or more of its axes to show multiple sides.

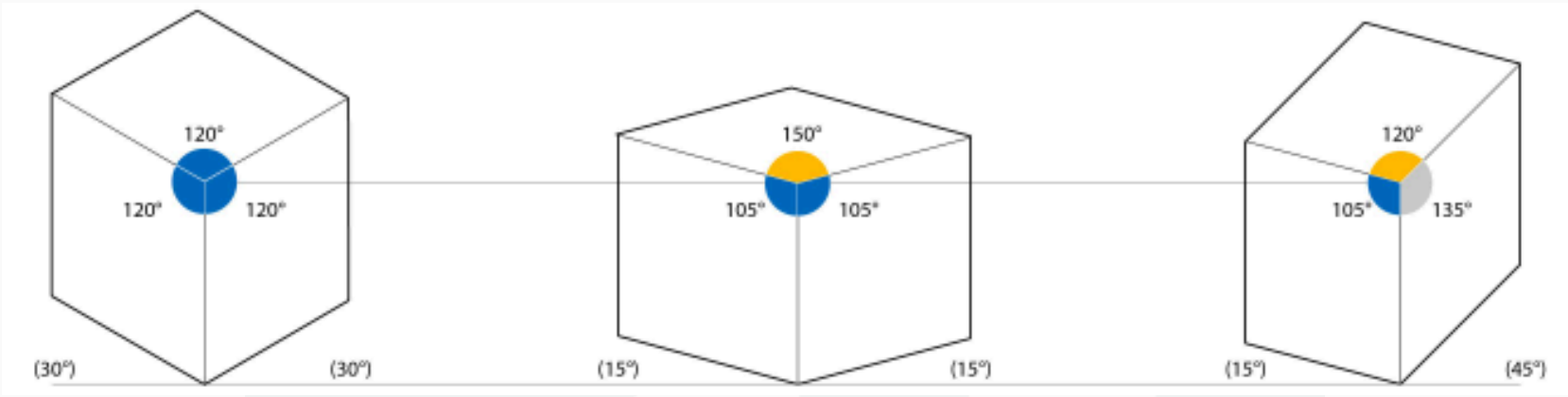
It is further divided into three categories :

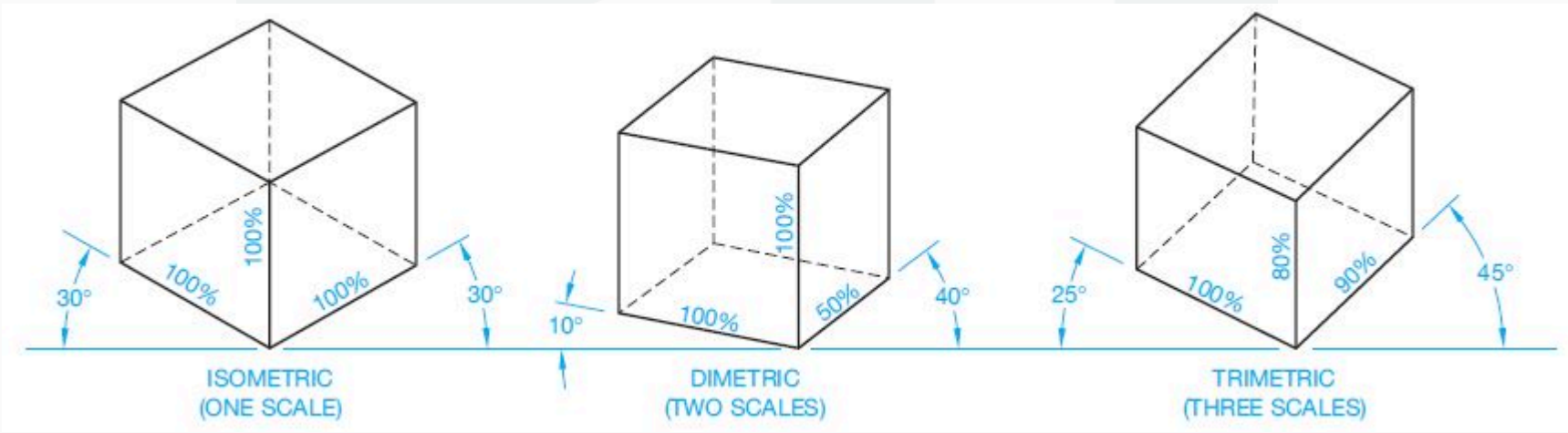
1) **Isometric Projection:** It is a method for visually representing three-dimensional objects in two-dimensional display in technical and engineering drawings. Here, in this projection, the object appears have three adjacent sides and angles are equal.

2) **Dimetric Projection:** It is a kind of orthographic projection where the visualized object appears to have only two adjacent sides and angles are equal.

3) **Trimetric Projection:** It is a kind of orthographic projection where the visualized object appears to have all the adjacent sides and angles unequal.

1. Orthographic Projection (2. Axonometric)





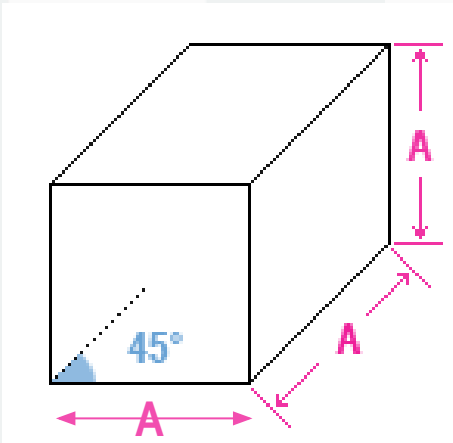
Two Categories of Parallel Projection (2. Oblique)

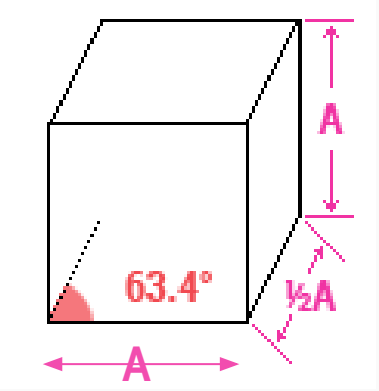
2. **Oblique Projection :** It is a kind of parallel projection where projecting rays emerges parallelly from the surface of the polygon and incident at an angle other than 90 degrees on the plane.

It is of two kinds:

A. **Cavalier Projection:** It is a kind of oblique projection where the projecting lines emerge parallelly from the object surface and incident at 45° rather than 90° at the projecting plane. In this projection, the length of the reading axis is larger than the cabinet projection.

B. **Cabinet Projection:** It is similar to that cavalier projection but here the length of reading axes just half than the cavalier projection and the incident angle at the projecting plane is 63.4° rather 45°.





Hidden Surface Removal and Line Removal

1. **Hidden Surface Removal (HSR):**
   - In computer graphics, **hidden surface determination** (also known as **hidden surface removal** or **hidden surface elimination**) is the process of determining which surfaces and parts of objects should be visible to the viewer and which should be hidden behind other objects. This is a crucial step in rendering realistic 3D scenes.
   - HSR ensures that only the front-most visible surfaces are displayed, while surfaces behind them are not rendered.
   - **Purpose:**
     - Prevents rendering of surfaces are not visible to viewer.
     - Enhances realism by correctly displaying depth and perspective.
     - Optimizes rendering by reducing computational load.
     - Provides accurate scene representation for simulations and visualizations.

Hidden Surface Removal and Line Removal

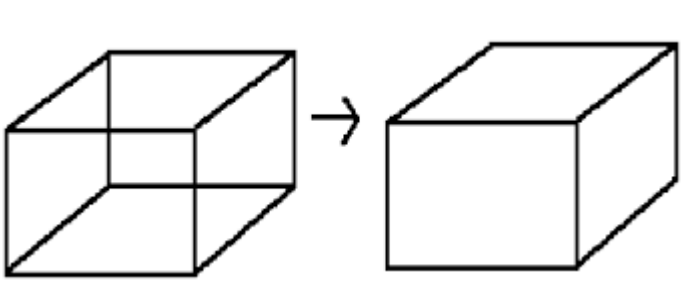
2. Hidden Line Removal (HLR):

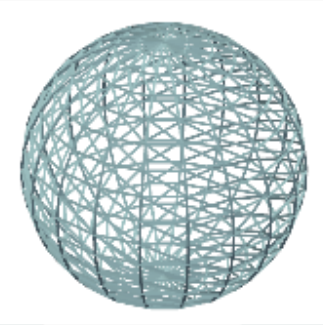
- HSR is a technique used to remove the lines of a 3D object that are not visible from a particular viewpoint.
- It is mainly used to generate clean wireframe models where only the visible edges of objects are displayed from front side, while the lines that would be obscured by other surfaces are hidden.
- **Purpose:**
  - To display only the visible edges of a 3D object.
  - To make wireframe representations more understandable by eliminating unnecessary visual clutter.
  - Commonly used in engineering drawings and CAD (Computer-Aided Design) applications.

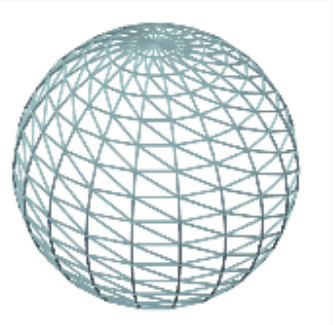
Hidden Surface Removal vs Hidden Line Removal

	HSR	HLR
Goal	Remove non-visible surfaces	Remove non-visible lines
Output	Realistic, shaded 3D image	Wireframe model with visible lines only
Rendering Type	Solid, coloured object	Edge-based representation
Use	Gaming, Simulations, VR	Engineering drawings, CAD designs
Computational Complexity	Typically, higher due to shading and lighting	Lower as it only focuses on lines

Hidden Surface Removal vs Hidden Line Removal



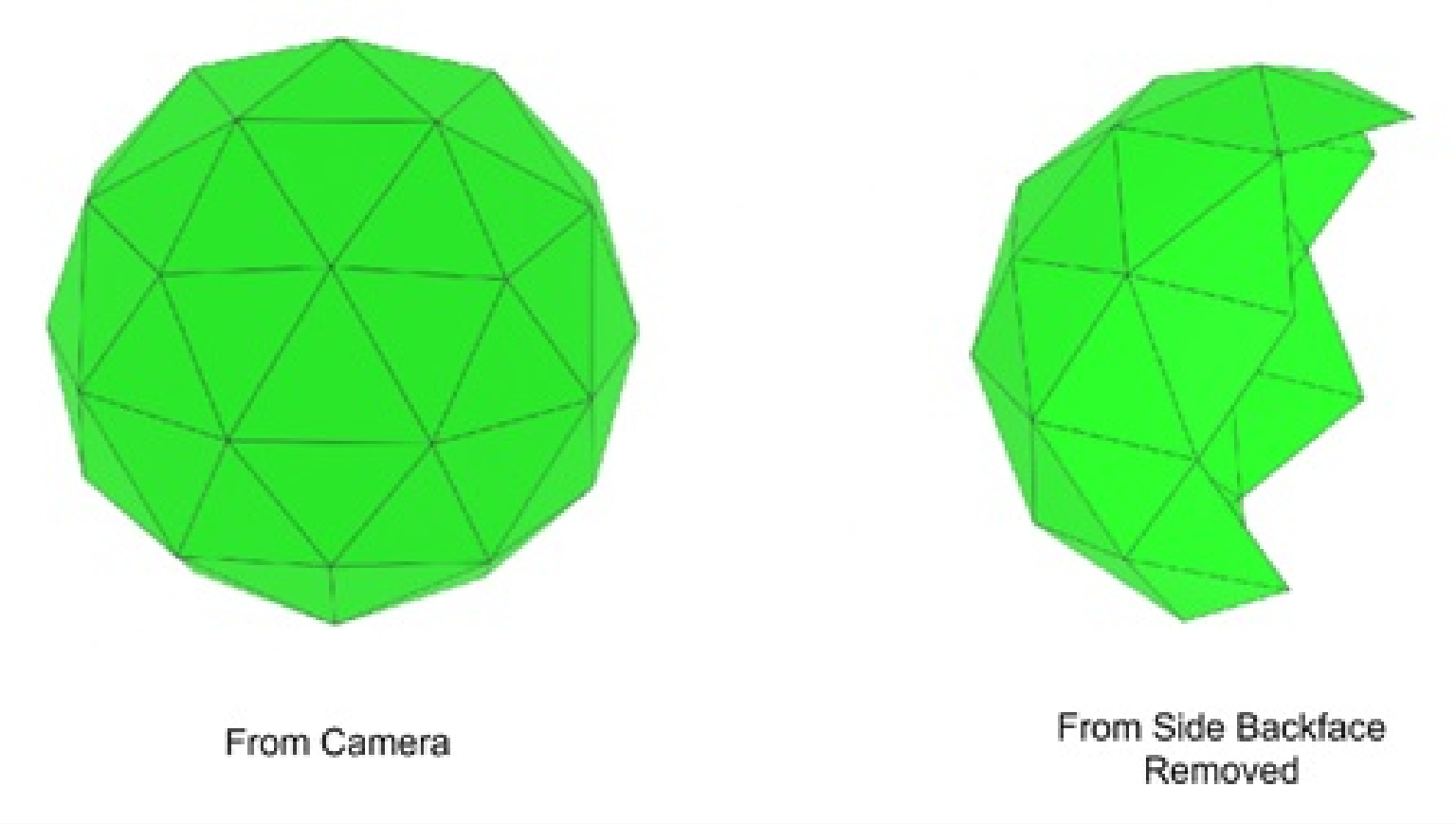




Back Face Removal (Culling) Algorithm

- This is used to improve rendering efficiency by eliminating surfaces of objects that are not visible to the viewer. It is similar to clipping but not exactly the same.
- When working on 3D models, some surfaces face away from the camera and should not be displayed. Removing these surfaces ensures that the computer does not waste time rendering parts of objects that cannot be seen.
- This method will remove 50% of polygons from the scene if the parallel projection is used. If the perspective projection is used then more than 50% of the invisible area will be removed. The object is nearer to the center of projection, number of polygons from the back will be removed.
- It applies to individual objects. It does not consider the interaction between various objects. Many polygons are obscured by front faces, although they are closer to the viewer, so for removing such faces back face removal algorithm is used.

Back Face Removal (Culling) Algorithm



Back Face Removal (Culling) Algorithm

- A vector is a mathematical quantity with both magnitude (length) and direction. In 3D graphics, vectors are represented as: $V=(x,y,z)$
- Vectors are used to describe points, directions, or surface orientations in 3D space.
- Types of Vectors:

  In the context of the back-face removal algorithm, two primary vectors are used:

  1. **Surface Normal Vector (N)**
     - The normal vector is a vector perpendicular to the surface of a polygon.
     - It is calculated using the cross product of two edge vectors of the polygon.
     - Formula to calculate the normal vector: $N=A\times B$
     - Where: A and B are two vectors along the edges of the polygon.

Back Face Removal (Culling) Algorithm

2. View Vector (V)

- The view vector points from the camera (eye) position to any point on the polygon's surface.
- If the camera is at the origin (0,0,0), the view vector is simply the position vector of the polygon:

$$V = (\text{Polygon Point} - \text{Camera Position})$$

- In most cases, for simplified calculations, if the camera is looking along the z-axis, the view vector can be represented as:

$$V = (0,0,-1)$$

Back Face Removal (Culling) Algorithm (Steps)

1. **Define Surface Norma's:**
   - Each polygon (typically triangles or quadrilaterals) has a normal vector, which is a perpendicular vector to the surface.
   - The normal vector is often calculated using the cross product of two edge vectors of the polygon.
2. **Calculate the View Vector:**
   - The view vector points from the camera (eye) position to a point on the polygon.
   - For simplicity, if the camera is at the origin, the view vector is often just the vector from the origin to the polygon.

Back Face Removal (Culling) Algorithm (Steps)

3. **Determine Visibility Using the Dot Product:**
   - Compute the dot product between the normal vector (N) and the view vector (V).
   - If the dot product is greater than or equal to zero:
     - $N \cdot V \geq 0 \rightarrow$ The face is not visible (Back-face).
   - If the dot product is less than zero:
     - $N \cdot V < 0 \rightarrow$ The face is visible (Front-face).
4. **Cull the Back-Faces:**
   - Faces with non-negative dot products are removed from the rendering pipeline.

Back Face Removal Algorithm

1. Define the Polygon: Represent the 3D object using polygons (typically triangles or quadrilaterals).
2. Calculate Surface Normal: Compute the normal vector of each polygon using the cross-product of its edges.
3. Determine the Viewing Vector: The viewing vector is a vector from the camera (eye) to any point on the polygon.
4. Compute the Dot Product: Calculate the dot product between the surface normal and the viewing vector.
   - If the dot product is positive, the polygon is facing away from the camera and can be removed.
   - If the dot product is negative or zero, the polygon is facing the camera and should be rendered.
5. Cull the Back-Face: Remove or ignore polygons with a positive dot product.

Back Face Removal Algorithm

1. **Consider a cube with a face at coordinates:**

   $A(1,1,1)$ $B(1,-1,1)$ $C(-1,-1,1)$ $D(-1,1,1)$

2. **Calculate the Normal Vector**:
   - Using vectors **AB** and **AC**:

     $N=(B-A)\times(C-A)$
   - After cross-product calculation:

     $N=(0,0,1)$
3. **Determine the Viewing Vector**:
   - Assume the camera is at (0,0,5) and looking towards the origin:

     $V=(0,0,-1)$
4. **Calculate Dot Product**:

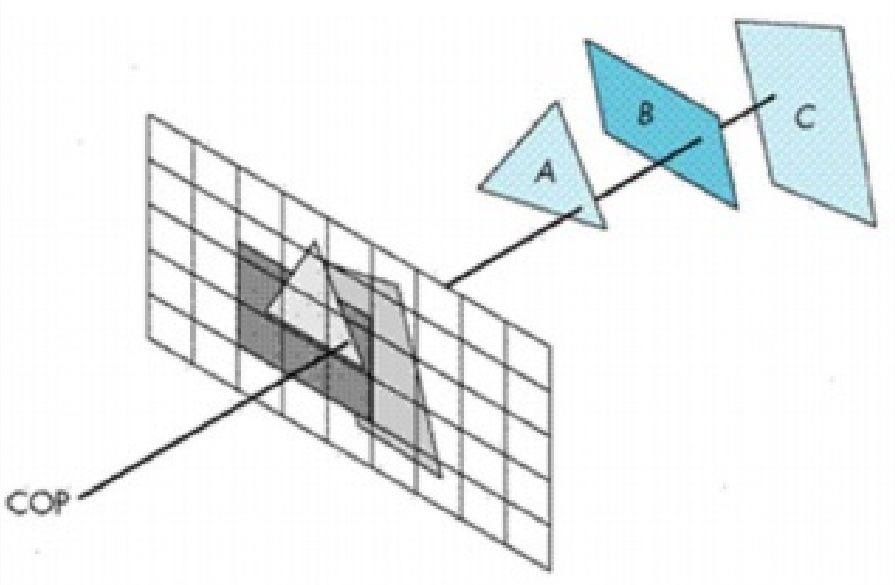
   $N\cdot V=(0,0,1)\cdot(0,0,-1)=-1$

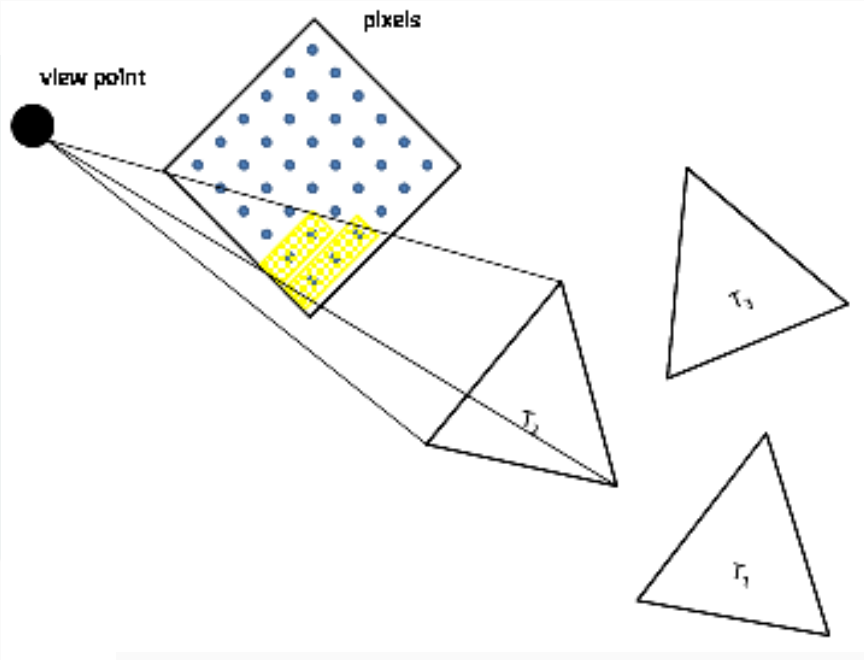
Since the dot product is negative, this face is a front-face and will be rendered.

Z-Buffer Algorithm

- The Z-buffer algorithm (also known as the depth-buffer algorithm) is a widely used technique in computer graphics for hidden surface removal.
- It helps determine which objects or surfaces should be visible in a 3D scene when projected onto a 2D screen.
- The algorithm ensures that objects closer to the camera are drawn while objects behind them are hidden.
- The algorithm uses a Z-buffer (also called a depth buffer) to store depth information for each pixel on the screen.
- Along with the Z-buffer, a Frame Buffer is used to store the color information (intensity) of visible pixels.
- For each pixel, the algorithm keeps track of the smallest (closest) z-coordinate — meaning the nearest object to the camera.

Z-Buffer Algorithm





Z-Buffer Algorithm (Steps)

1. **Initialization:**
   - Create two buffers of the same resolution as the screen:
     - Z-buffer (Depth Buffer): Stores the depth (z-value) for each pixel.
     - Frame Buffer (Color Buffer): Stores the pixel color values.
   - Initialize the Z-buffer with a very large value (e.g., infinity or the farthest plane).
   - Initialize the Frame Buffer with the background color.
2. **Processing Each Polygon:**
   - For every polygon (or surface) in the 3D scene:
     - Calculate the depth (z-coordinate) for each pixel covered by the polygon using interpolation or depth calculation methods.

Z-Buffer Algorithm (Steps)

3. **Compare Depths:**
   - For each pixel, compare the polygon's depth (z-value) with the current value in the Z-buffer.
   - If the polygon's z-value is smaller (closer to the camera):
     - Update the Z-buffer with the new depth value.
     - Update the Frame Buffer with the polygon's color.
   - Otherwise, discard the pixel since it is hidden.
4. **Final Image Rendering:**
   - After processing all polygons, the Frame Buffer contains the final image, displaying only the visible parts of objects.

Scan-Line algorithm

- The core idea of the scan-line algorithm is to process the scene line by line (in horizontal or vertical lines, depending on the implementation) and determine which surfaces are visible at each line.
- This is done by projecting the 3D scene onto a 2D plane (the screen or the view plane) and sorting the surfaces in each scan-line based on their depth.
- Instead of processing all pixels individually like the Z-buffer algorithm, the scan-line algorithm processes the scene line by line.
- It compares the depth values (Z-values) of overlapping polygons along each scan line to identify the visible surface.
- The algorithm maintains data about active polygons and edges as it progresses from one scan line to the next.

Scan-Line Algorithm (Steps)

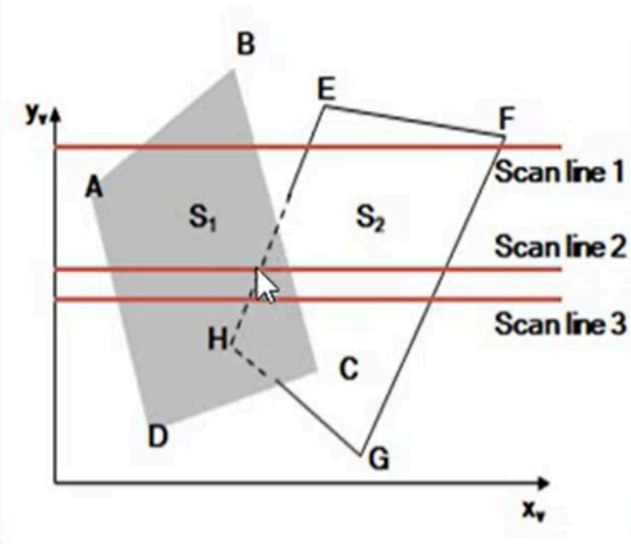
1. **Initialization:**
   - Initialize a buffer (called the Scan-line Buffer) to store pixel color values for the current scan line.
   - Initialize a depth buffer (Z-buffer) to store the depth (Z) value for each pixel.
2. **Determine Active Polygons and Edges:**
   - Identify which polygons intersect the current scan line.
   - Maintain an ***Active Edge List (AEL)*** that contains all the polygon edges that cross the scan line.

Scan-Line Algorithm (Steps)

3. **Compute Depth and Color:**
   - For each pixel along the scan line, calculate the Z-value using linear interpolation.
   - Compare the Z-values of overlapping polygons to determine the nearest polygon.
   - Update the Scan-line Buffer and Z-buffer with the color and depth of the visible polygon..
4. **Move to the Next Scan Line:**
   - Increment the scan line and repeat the process until all pixels are processed.

Scan-Line Algorithm

- We can deal with multiple surface as each scan line is processed.
- All polygon surface intersecting that line are examined to determine visible for all polygons intersecting each scan line.
  - Processed from left to right
  - Depth calculations for each overlapping surfaces.
  - The intensity of the nearest position is entered into the refresh buffer.



Differences: Back-Face, Z-Buffer, Scan-Line

	Back-Face	Z-Buffer	Scan-Line
Purpose	Removes polygons facing away from the camera.	Identifies the closest surface for each pixel using depth comparison.	Determines visible surfaces along horizontal scan lines.
Working Principle	Uses the surface normal to check if the polygon is a back-face.	Uses a depth buffer to store the z-values (depth) for each pixel.	Compares depth values of polygons along each scan line.
Memory Usage	Very low memory usage.	High memory usage due to the Z-buffer.	Moderate memory usage.
Computational Cost	Fast and efficient for large scenes with many polygons.	Computationally expensive for complex scenes.	Efficient for scenes with fewer polygons.

Differences: Back-Face, Z-Buffer, Scan-Line

	Back-Face	Z-Buffer	Scan-Line
Handling Overlapping Objects	Cannot handle overlapping objects.	Effectively handles overlapping objects.	Can handle overlapping objects with depth comparison.
Accuracy	Less accurate, especially for complex scenes.	Highly accurate.	Moderately accurate.
Suitability	Best for wireframe models and simple 3D objects.	Suitable for real-time rendering (e.g., games).	Useful for rendering in CAD and simulation applications.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in