Computer Graphics

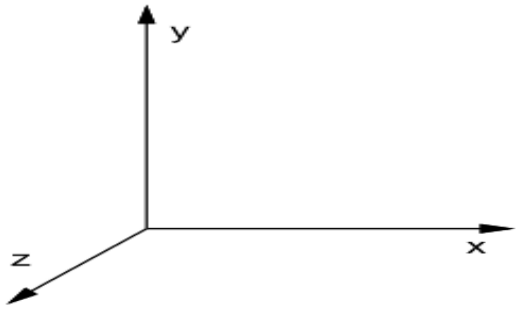
Ch5: 3D Graphics

Outline

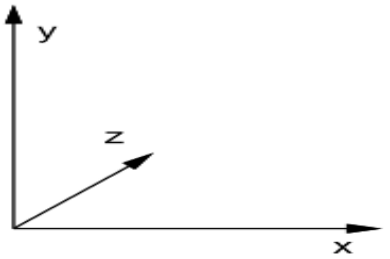
- 3D Geometry
- Representation of 3D Objects
- 3D Transformations
- 3D Display Methods
- Projections
  - Parallel Projection
  - Perspective Projection

3D Geometry - 3D co-ordinate systems

- A point in 3D space can be referenced relative to RHS or LHS Cartesian coordinate system using **three coordinate values.**
- **RHS -right-handedness** is defined as any positive axis (x, y, or z) pointing toward the viewer. or LHS Cartesian coordinate system- Left-handedness is defined as any positive axis (x, y, or z) pointing away from the viewer.
  - This can be represented as a 1×3 matrix, (or as a 1×4 matrix for homogenous transformations).
  - We will normally be working with an RHS system.
  - The transformation $S(1,1,-1)$ can be used to transform from a right-handed 3D Cartesian coordinate system to a left-handed one and vice-versa.



Right–handed system

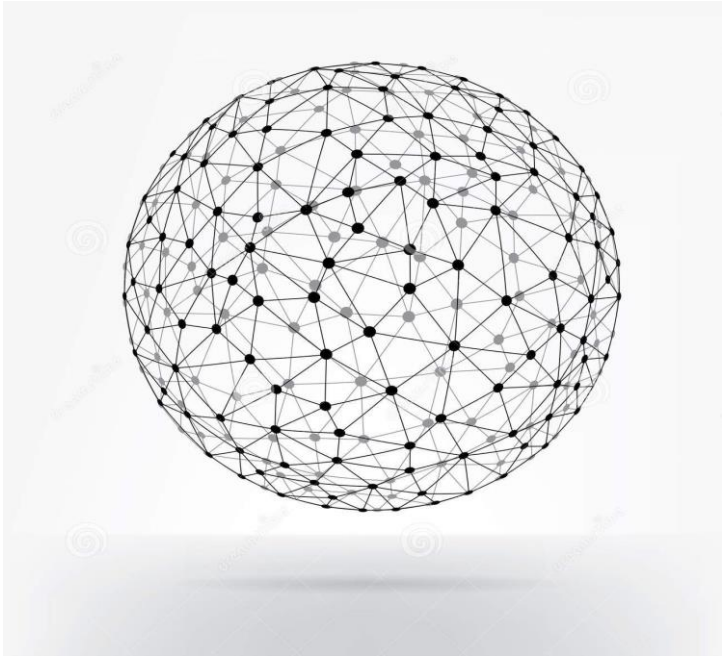


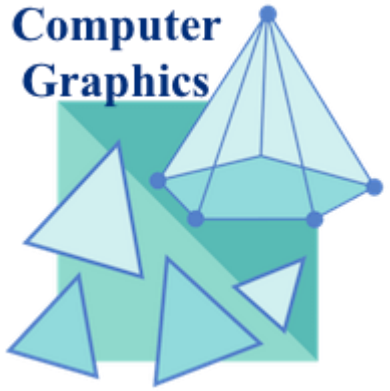
Left–handed system

Representation of 3D objects

- One way of representing a 3D object is as a list of 2D polygons in 3D space.
- For example, a pyramid consists of a square base and four triangles.
  - Curved surfaces may be approximated using polygons, although there are techniques which represent curved surfaces more accurately, including generalizations of Bezier and Spline curves.
  - If 3D objects are represented as lists of edges (as opposed to polygons), then this allows rendering of 'see-through' wireframe images.
  - However, **hidden-line removal** for more realistic images is not possible because this requires knowledge of the surfaces of the object.

wireframe images





Hidden line removal (HLR) is **the method of computing which edges are not hidden by the faces of parts for a specified view and the display of parts in the projection of a model into a 2D plane**. Hidden line removal **is utilized** by a CAD to display the visual lines

3D Transformations

- 3D transformations are just an extension of 2D ones.
  - Just like 2D transformations are represented by 3×3 matrices, 3D transformations are represented by $\mathbf{4 \times 4}$ **matrices.**

1. Translation:

- It is the movement of an object from ***one position to another position***.
- Translation is done using translation vectors.
- There are three vectors in 3D instead of two. These vectors are in x, y, and z directions.
- Translation in the x-direction is represented using Tx. The translation is y-direction is represented using Ty. The translation in the z- direction is represented using Tz.
- If P is a point having **co-ordinates** in three directions ***(x, y, z)*** is translated, then after *translation* its coordinates will be $(x' y' z')$ after translation. T_x T_y T_z are translation vectors in x, y, and z directions respectively.

$$x' = x + T_x$$
$$y' = y + T_y$$
$$z' = z + T_z$$

Cont....

- Three-dimensional transformations are performed by transforming each vertex of the object. If an object has five corners, then the translation will be accomplished by translating **all five points** to new locations. Following figure 1 shows the translation of point figure 2 shows the **translation** of the cube.

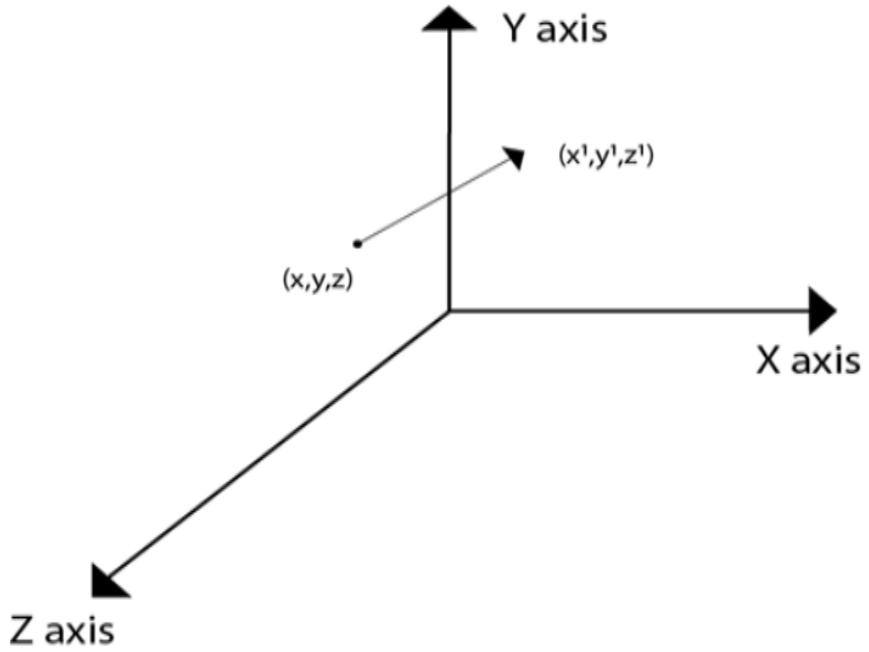


figure 1

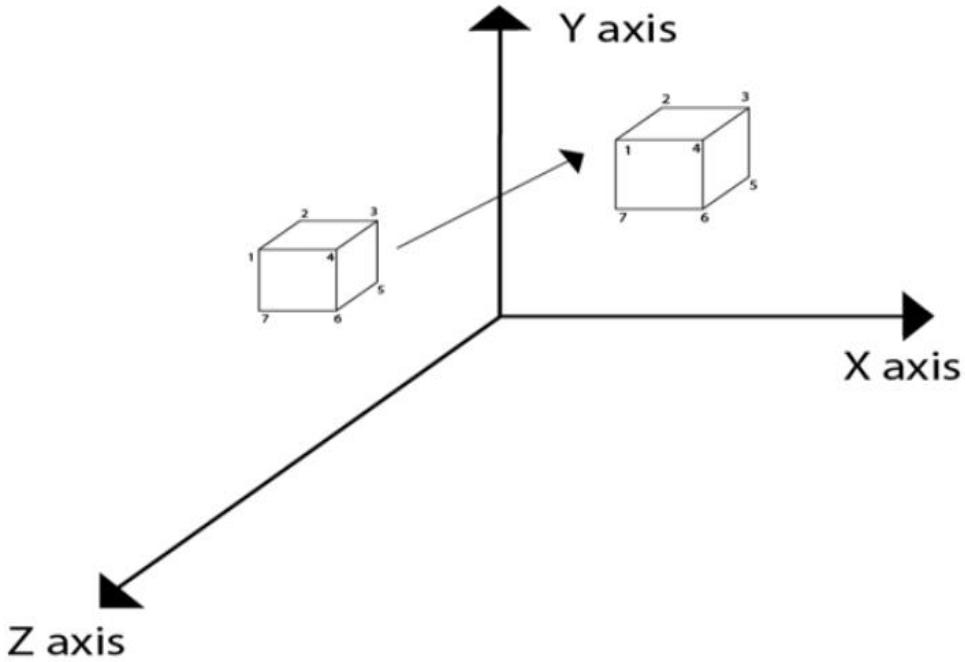


figure 2

Cont…

Translation by (t_x, t_y, t_z) is represented by the **matrix:**

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{pmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

or

$$\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{Bmatrix}$$

Example: A point has coordinates in the x, y, z direction i.e., **(5, 6, 7).** The translation is done in the x-direction by 3 coordinate and y direction. Three coordinates and in the z- direction by **two coordinates**. Shift the object. Find coordinates of the new position.

Solution: Co-ordinate of the point are (5, 6, 7)

Translation vector in x direction = 3

Translation vector in y direction = 3

Translation vector in z direction = 2

Translation matrix is

$$(x^1 y^1 z^1) = (5,6,7,1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 3 & 2 & 1 \end{pmatrix}$$

x becomes $x^1=8$

y becomes $y^1=9$

z becomes $z^1=9$

Example

- 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.

Solution

- Given-
- Old coordinates of the object = A (0, 3, 1), B(3, 3, 2), C(3, 0, 0), D(0, 0, 0)
- Translation vector = $(T_x, T_y, T_z) = (1, 1, 2)$
- Applying the translation equations, we have-

For Coordinates A(0, 3, 1)

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$
- $Z_{new} = Z_{old} + T_z = 1 + 2 = 3$

For Coordinates B(3, 3, 2)

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$
- $Z_{new} = Z_{old} + T_z = 2 + 2 = 4$

For Coordinates C(3, 0, 0)

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$
- $Z_{new} = Z_{old} + T_z = 0 + 2 = 2$

For Coordinates D(0, 0, 0)

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$
- $Z_{new} = Z_{old} + T_z = 0 + 2 = 2$

- Thus, New coordinates of the object = **A (1, 4, 3), B(4, 4, 4), C(4, 1, 2), D(1, 1, 2).**

2. Scaling

- Scaling is used to **change the size** of an object. The size can be **increased or decreased**. The scaling three factors are required S_x, S_y and S_z.
  - Scaling by the factors s_x, s_y and s_z, along the co-ordinate axes is given by the matrix:

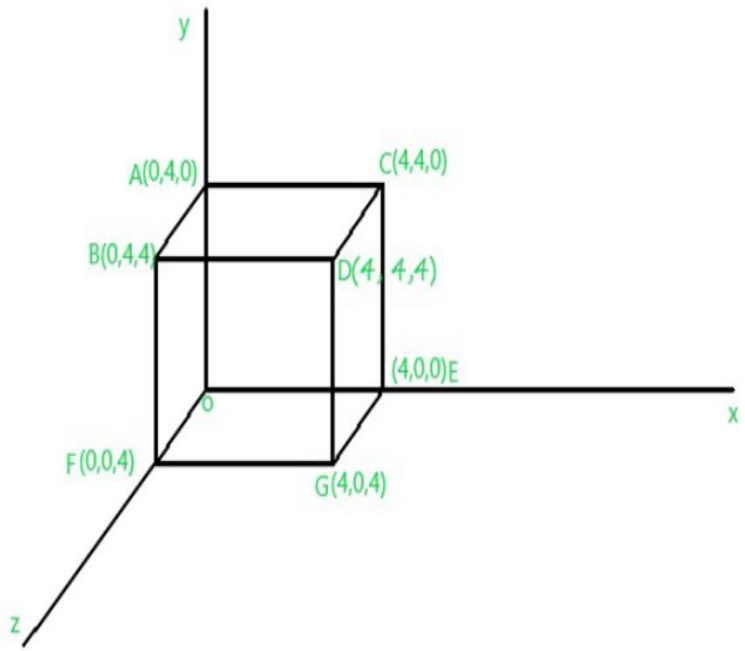
$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

  - The following kind of sequences occur while performing the *scaling transformations on a fixed point* –
    - The fixed point is translated to the origin.
    - The object is scaled.
    - The fixed point is translated to its original position.

Example

1. Consider the above problem where a cube" OABCDEFG" is given O(0, 0, 0,), A(0, 4, 0), B(0, 4, 4), C(4, 4, 0), D(4, 4, 4), E(4, 0, 0), F(0, 0, 4), G (4, 0, 4) and we are given with Scaling factor **Sx=2, Sy=3, Sz=2.** Perform Scaling operation over the cube.



$$O'[x,y,z,1] = [0\,0\,0\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [0\,0\,0\,1]$$

$$A'[x,y,z,1] = [0\,4\,0\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [0\,12\,0\,1]$$

$$B'[x,y,z,1] = [0\,4\,4\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [0\,12\,8\,1]$$

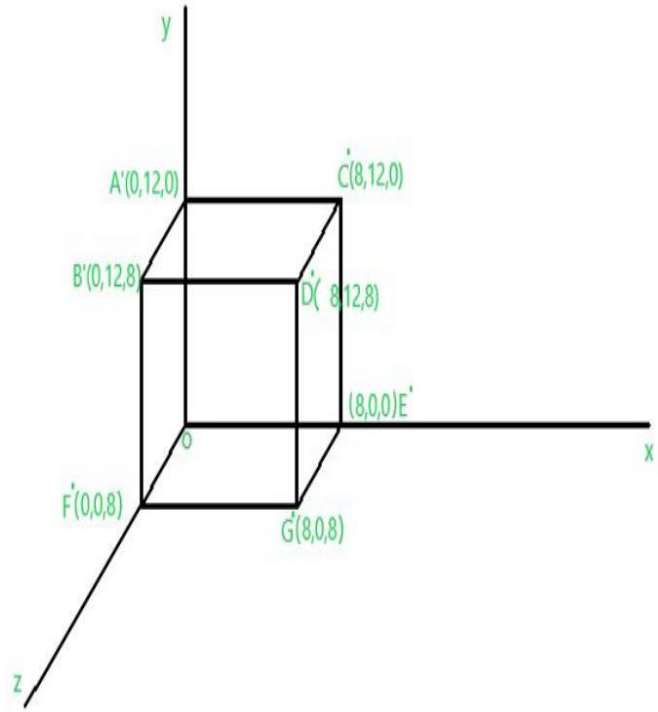
$$C'[x,y,z,1] = [4\,4\,0\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [8\,12\,0\,1]$$

$$D'[x,y,z,1] = [4\,4\,4\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [8\,12\,8\,1]$$

$$E'[x,y,z,1] = [4\,0\,0\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [8\,0\,0\,1]$$

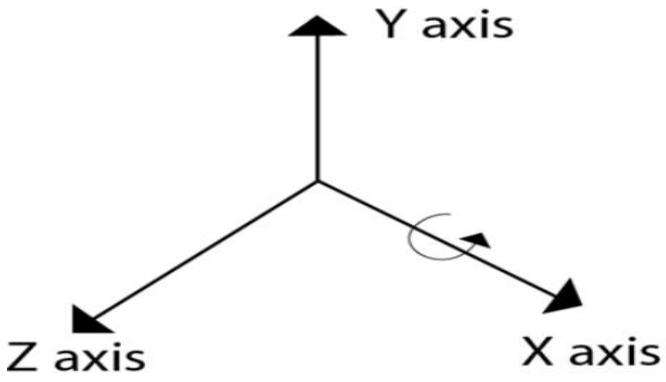
$$F'[x,y,z,1] = [0\,0\,4\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [0\,0\,8\,1]$$

$$G'[x,y,z,1] = [4\,0\,4\,1]\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&1\end{bmatrix} = [8\,0\,8\,1]$$

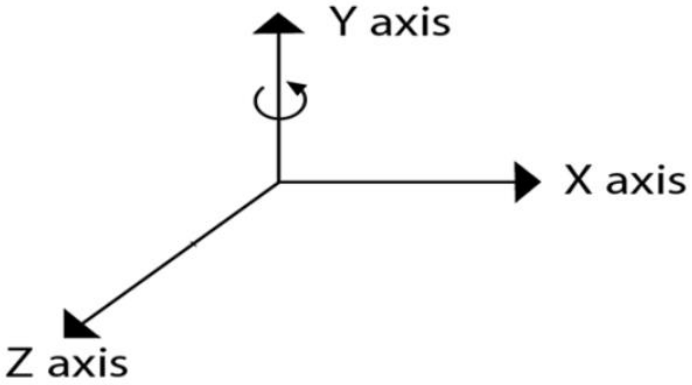


3. Rotation:

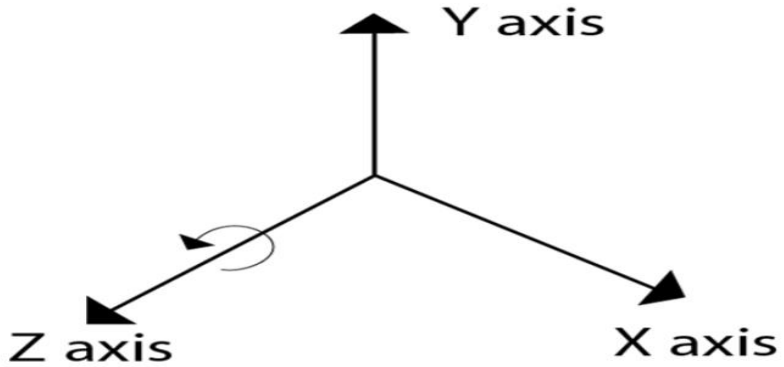
- It is moving of an object about an angle. Movement can be anticlockwise or clockwise. 3D rotation is complex as compared to the 2D rotation. For **2D we describe the angle of rotation**, but for a 3D angle of rotation and axis of rotation are required. The axis can be either x or y or z.



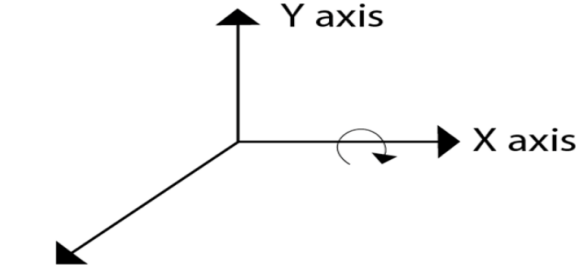
rotation object X axis anticlockwise



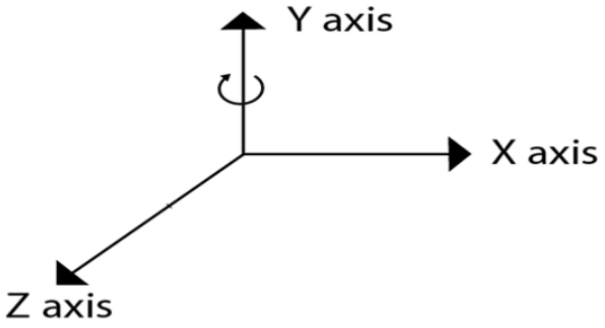
rotation object Y axis anticlockwise



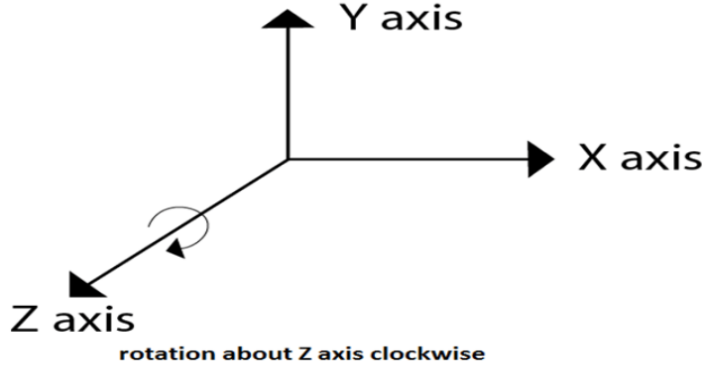
rotation object Z axis anticlockwise



rotation about X axis clockwise



rotation object Y axis clockwise



rotation about Z axis clockwise

Cont...

- Rotation about the $x-axis$:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Rotation about the $y-axis$:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Rotation about the $z-axis$:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Example

1. Given a triangle with corner coordinates (0, 0), (1, 0) and (1, 1). Rotate the triangle by 90 degree anticlockwise direction and find out the new coordinates.

Solution-

- Given- Old coordinates = $(X_{old}, Y_{old}, Z_{old}) = (1, 2, 3)$, Rotation angle = $\theta = 90^o$,$\cos 90^o=0$ $\sin 90^o=1$

For X-Axis Rotation-

Applying the rotation equations, we have-

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

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

For Y-Axis Rotation-

Applying the rotation equations, we have-

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

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

For Z-Axis Rotation-

Applying the rotation equations, we have-

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

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

Exercise

1. Consider a point p(5,4,9) and scaling factors are (2,3,5) using 3D scaling matrix find the new coordinates of P'.
2. Consider cube "OABCDEFG" is given

O(0,0,0), A(0,4,0),B(0,4,4),C(4,4,0),D(4,4,4),E(4,0,0),F(0,0,4),G(4,0,4) and scaling factors Sx=2,sy=3,sz=2.Perform scaling operation over the cube and find new coordinates.

3. Perform translation transformation on the following figure where the given translation distances are $D_x = 2$, $D_y = 4$, $D_z = 6$.

3.1.Rotate 90 and 180 degree

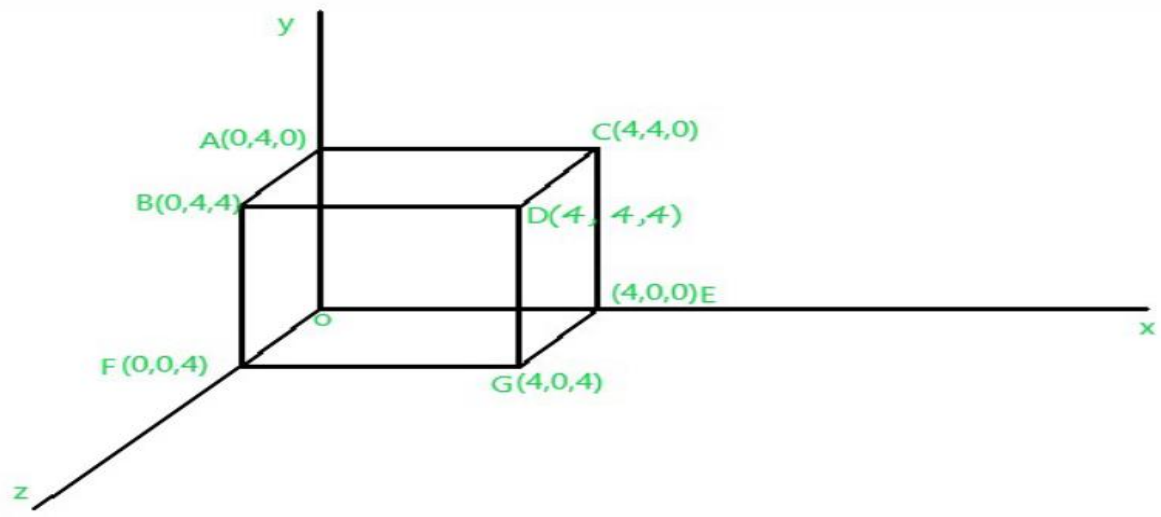


Fig.1

3D Display Methods – The 3D viewing pipeline

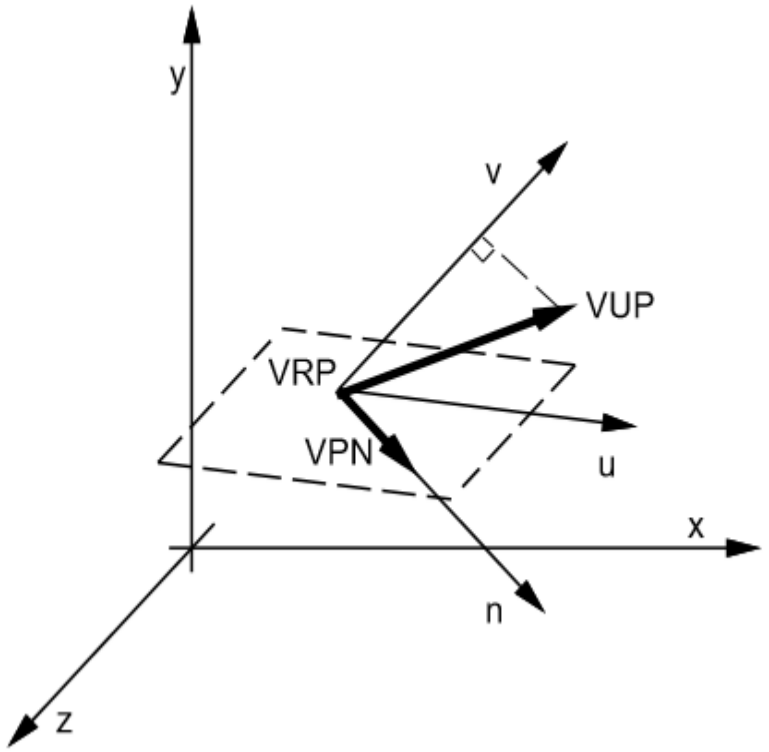
- 3D viewing is more complex than 2D viewing because, while the world coordinate systems is **three dimensional**, the device on which they are **displayed** is usually **two dimensional**.
  - This requires an **additional transformation**, namely the **projection transformation**, in the viewing pipeline, whereby **3D coordinates** are transformed into **2D**.
- The basic steps involved in ***3D viewing*** are:
  1. Transforming world coordinates into ***viewing coordinates***;
  2. Transforming viewing coordinates into **projection coordinates** (the projection transformation);
  3. Transforming projection coordinates into **device coordinates.**

3D Display Methods – The 3D viewing pipeline

- The first step effectively defines a **position and orientation** for a **virtual camera**, whereas the second step determines the kind of **picture to be taken**.
- Note that whereas clipping in 2D is carried out before applying the viewing pipeline, in 3D it is often delayed to just before the third step.
  - This is because in 3D, clipping has to be performed against a view volume which may be of an awkward shape e.g. a frustrum in the initial stages.
- After carrying out the projection transformation, the view volume is a regular parallelepiped which is much **easier to clip against**.

3D Display Methods–The 3D viewing pipeline The view reference coordinate system

- The **viewing-reference coordinate system** (VRC) is defined in terms of
  - a view **reference point** $(\boldsymbol{VRP})$ which represents the origin,
  - a **view-plane normal vector** (VPN) and
  - a **view-up vector** $(\boldsymbol{VUP})$.
- This is analogous to placing a camera (or an eye) at the VRP pointing in the direction given by the VPN.
- The camera is then **rotated** around the perpendicular to the plane that passes through it (the camera) so that the $\boldsymbol{VUP}$ is vertical.



3D Display Methods – The 3D viewing pipeline The view reference coordinate system

- The coordinate system $\boldsymbol{uvn}$ is set up so that n lies in the direction of the VPN, and $\boldsymbol{v}$ is in the direction of the vector which is in the same plane of the VPN and VUP, is perpendicular to the $\boldsymbol{VNP}$ and makes an acute angle with the $\boldsymbol{VUP}$.
- $\boldsymbol{u}$ is chosen so that $\boldsymbol{uvn}$ is **right-handed**.
- The orthogonal unit vectors u, v and n corresponding to the $\boldsymbol{VRC}$ can be constructed as follows:

$$
\begin{aligned}
n &= \frac{1}{|VPN|} VPN \\
u &= \frac{1}{|VUP \times VPN|} VUP \times VPN \\
v &= n \times u
\end{aligned}
$$

- Let the $\boldsymbol{VRP}$ have world coordinates $(\boldsymbol{x_0}, \boldsymbol{y_0}, \boldsymbol{z_0})$, and let

$$
\begin{aligned}
u &= (u_x, u_y, u_z) \\
v &= (v_x, v_y, v_z) \\
n &= (n_x, n_y, n_z)
\end{aligned}
$$

- Transforming from world coordinates to viewing coordinates can be achieved by first **translating** by $(-x_0, -y_0, -z_0)$ and then **aligning** the coordinate axes by **rotation.**
- Thus if $\boldsymbol{P}$ is a point with **world coordinates** $\boldsymbol{Pw}$, its viewing coordinates $\boldsymbol{Pv}$ can be **obtained** using:

where

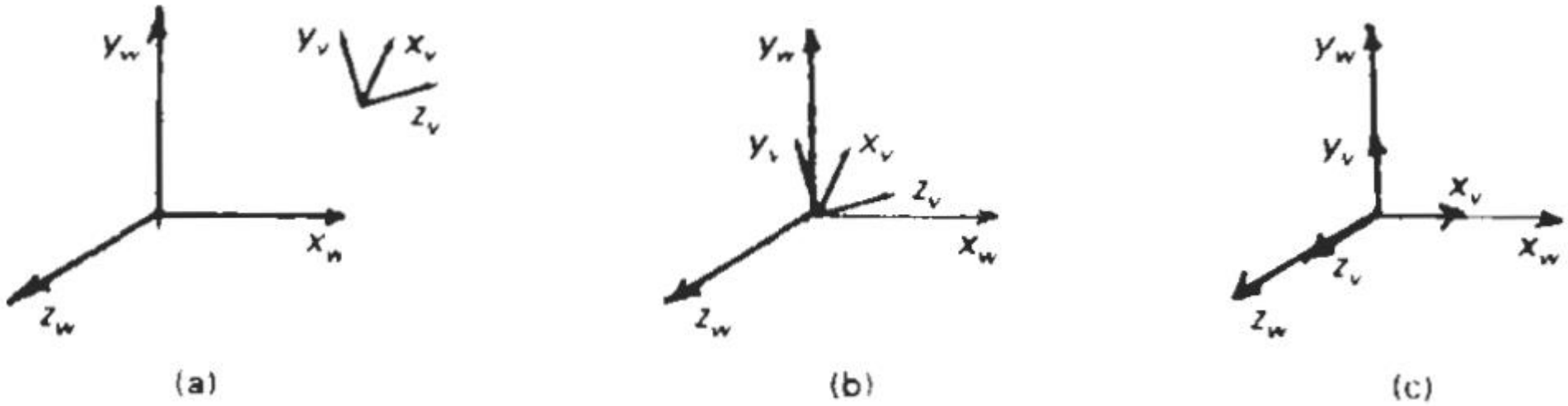
$$P_v = M_{wv} \cdot P_w$$

$$M_{wv} = R_{uvn} \cdot T(-x_0, -y_0, -z_0)$$

$$R_{uvn} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

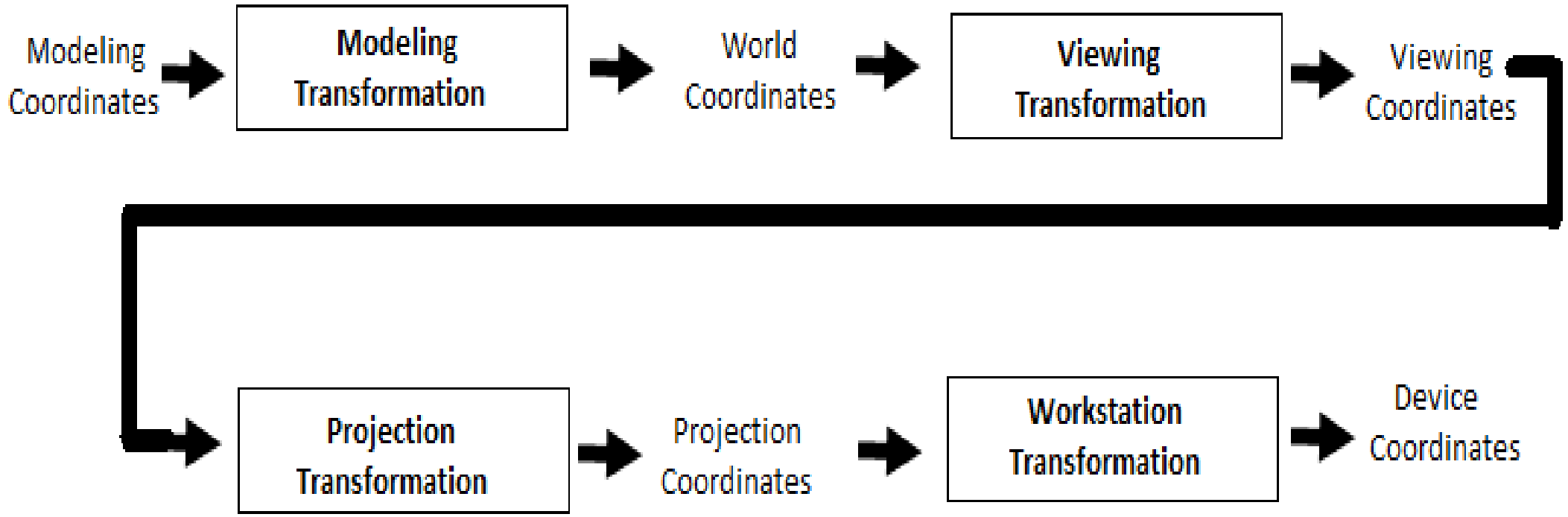
3D Display Methods – The 3D viewing pipeline The view reference coordinate system

- Aligning a viewing system with the world-coordinate axes using a sequence of translate-rotate transformations:



3D Display Methods – The 3D viewing pipeline The view reference coordinate system

- General *three-dimensional transformation* pipeline, from modeling coordinates to final device coordinate is:



Projections

- In general, a ***projection*** is a transformation from an *n* dimensional space to an *m* dimensional space, where $m < n$.
  - We **concentrate on projections** from a **3D Cartesian coordinate** system to a **2D** one.
- Consider a set of points (x_i, y_i, z_i) in **viewing coordinates.**
- To carry out a projection, a line (called a ***projector***) is drawn from each point to a fixed point called the ***center of projection*** (***COP***).
  - The projectors intersect the **view plane** (which is perpendicular to z_v) at the points $(x_i', y_i', 0)$.
  - Projections that can be represented as

$$J : (x_i, y_i, z_i) \mapsto (x_i', y_i')$$

are called planar ***geometric projections***.

Projections

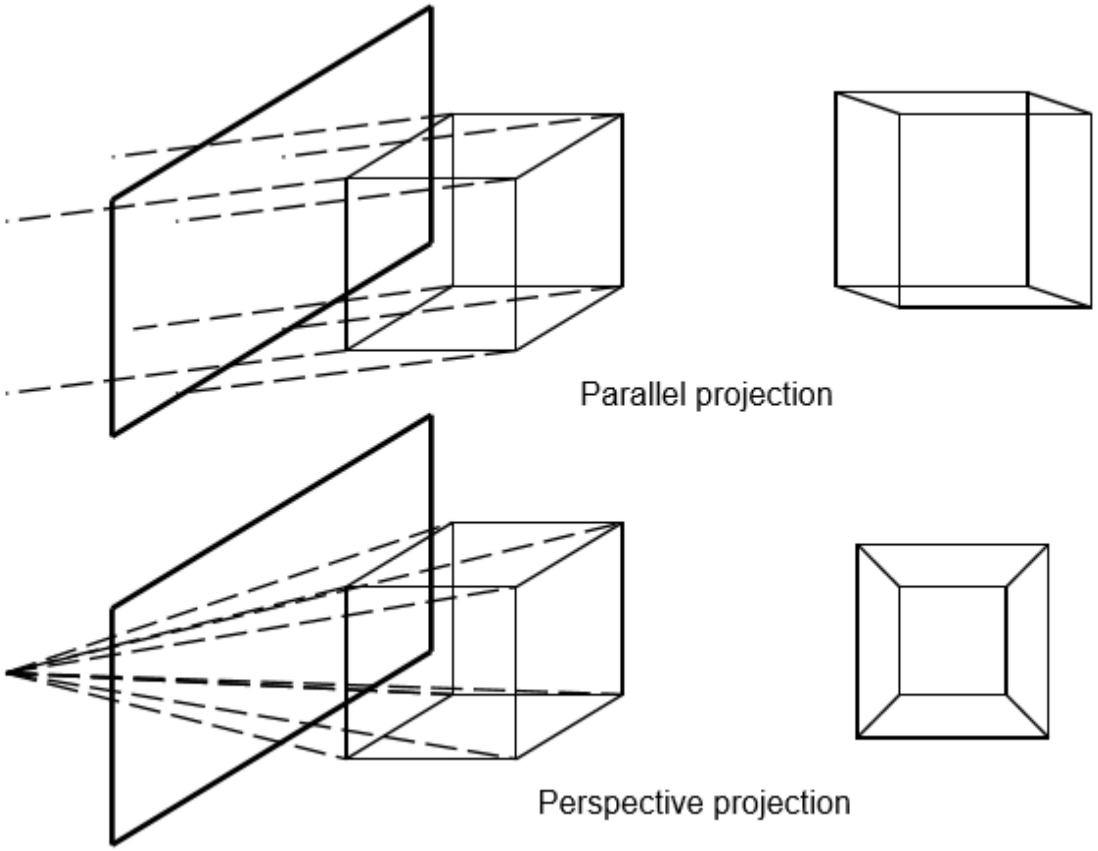
- The following parameters must be supplied to fully describe a projection:
  - a ***window*** in the view plane that defines the boundaries of the picture to be taken—this is specified in **VRC coordinates** and is usually, though not necessarily, taken to be a rectangle with the VRP at the center;
  - a ***COP*** towards which all projectors converge;
  - ***front*** and ***back*** **clipping *planes*:**
    - the ***front*** and ***back*** **clipping** planes define **cut-off points** beyond which graphic objects are not considered for projection, effectively eliminating objects which are too close to (or behind) the **camera**, or **too** far away from it.

Projections

- The position of the **COP** may tend to infinity; in such cases the projectors are parallel to each other and their direction is referred to as the ***direction of projection***.

  - A planar geometric projection constructed using parallel projectors is called a ***parallel projection***.

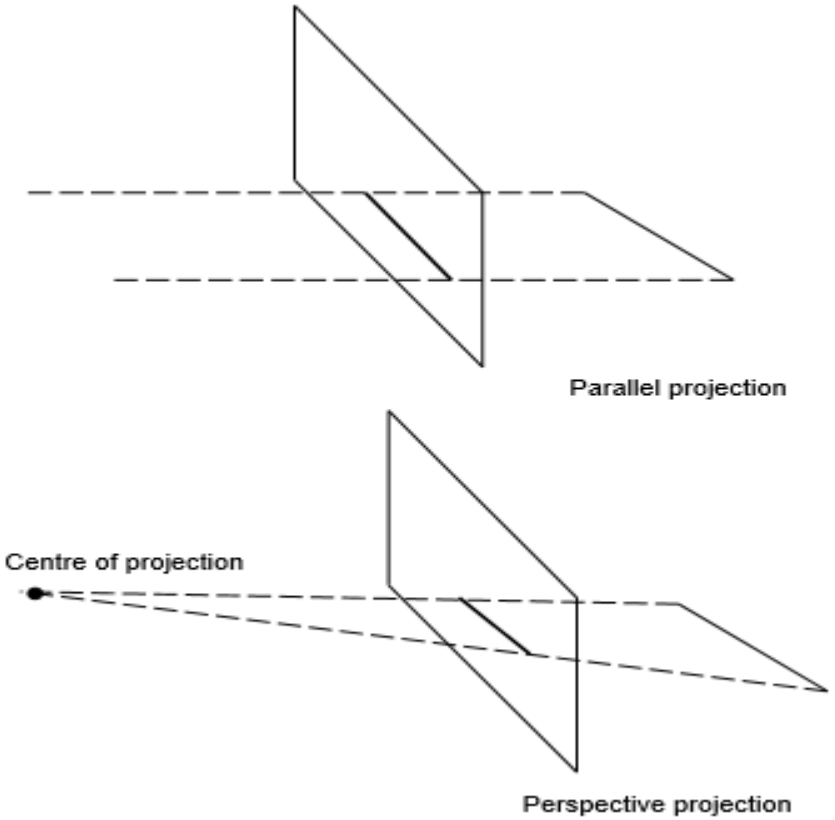
  - If the projectors intersect at the center of projection, the projection is called a ***perspective projection***.



Projections

- ***Parallel projections*** preserve all parallel lines as well as the angles in any plane parallel to the plane of projection.
- ***Perspective projections*** preserve parallel lines and angles only in the planes parallel to the plane of projection.

  - However, scenes produced using a perspective projection appear realistic because distant objects appear smaller than near ones.
  - In a scene produced by parallel projection, distant objects have the same size as near ones.
  - Perspective projections of parallel lines are either parallel lines themselves (if they are parallel to the view plane) or intersect at a ***vanishing point***.



Parallel Projections- Orthographic projections

- If the projectors are perpendicular to the plane of projection, the resulting parallel projection is called ***orthographic***, otherwise it is called ***oblique***.

Orthographic projections

- A useful set of views produced by orthographic projections are the ***front elevation***, ***side elevation*** and ***plan view*** of an object.
  - Such views are produced when the plane of projection is parallel to one of the planes defined by any two principal axes of the world coordinate system, or equivalently, the plane of projection intersects with only one world coordinate principal axis.

Parallel Projections- Orthographic projections

- The front elevation of orthographic parallel projection can be represented by the following transformation:

$$x_p = x_v$$
$$y_p = y_v$$

or by the matrix:

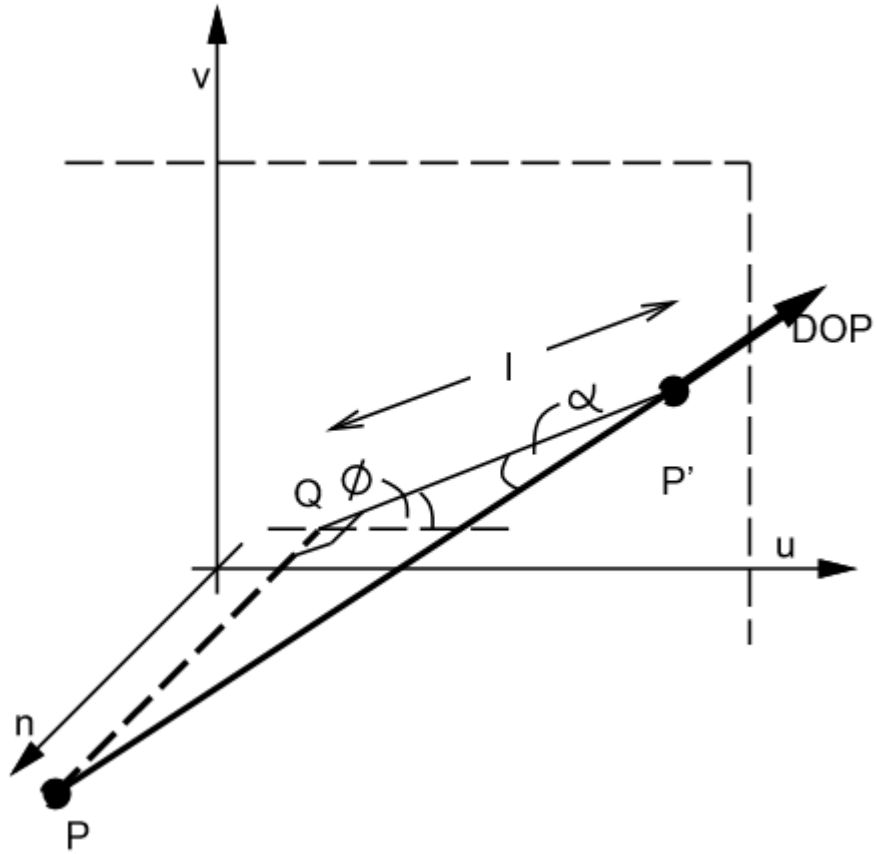
$$J_{FrontElevation} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parallel Projections- Orthographic projections

- If the projection is orthographic and the plane of projection intersects with more than one world coordinate axis, then the resulting view is termed ***axonometric***.
  - One useful axonometric projection is the isometric projection, which is produced when the projection plane intersects each principal axis at the same distance from the origin.
  - The projected edges of an isometric view of a cube aligned along the world co-ordinate axes have equal length.

Parallel Projections- Oblique projections

- Oblique parallel projections are produced when the projectors are not perpendicular to the plane of projection.
- The direction of projection (DOP) may be given by two angles: α and φ.
  - α is defined as the angle each projector makes with the plane of projection, and
  - φ as the angle the horizontal makes with the line that joins the projection with the corresponding **orthographic** projection.
    - φ is usually chosen to be **30**◦ or **45**◦.



Parallel Projections -Parallel projection transformation

- Consider the point P with view coordinates (x_v, y_v, z_v) projected to the point P' with coordinates (x_p, y_p).
- Let Q be the orthographic projection of P on the plane, and l be the length of line QP'.
- Then $l = \frac{z_v}{\tan \alpha}$ and the coordinates of P' are:

$$\begin{aligned} x_p &= x_v + l\cos\phi \\ &= x_v + z_v\frac{\cos\phi}{\tan\alpha} \\ y_p &= y_v + l\sin\phi \\ &= y_v + z_v\frac{\sin\phi}{\tan\alpha} \end{aligned}$$

- Oblique parallel projections can thus be obtained using the transformation matrix:

$$J_{Oblique} = \begin{bmatrix} 1 & 0 & \frac{\cos\phi}{\tan\alpha} & 0 \\ 0 & 1 & \frac{\sin\phi}{\tan\alpha} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parallel Projections -Parallel projection transformation

- In the case of an orthographic projection, we have $\varphi = 0$ and $\alpha = 90$, giving us the transformation:

$$J_{Orthogonal} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Note that in the orthogonal case we simply discard the z coordinate.
- Note also that when using the matrix representations of these transformations we still end up with a vector representing a point in 3D space, albeit with a z value of 0.

Perspective Projections

- Let us consider a point P with viewing coordinates (x_v, y_v, z_v).
- The point is projected to P' with coordinates (x_p, y_p).
- We will consider the special case where the center of projection lies on the $z_v - axis$ at the point $(0,0, c_z)$.
- The projection is given by the transformation:

$$x_p = x_v \frac{-c_z}{z_v - c_z} \qquad y_p = y_v \frac{-c_z}{z_v - c_z}$$

$$= x_v \frac{1}{-\frac{z_v}{c_z} + 1} \qquad = y_v \frac{1}{-\frac{z_v}{c_z} + 1}$$

Perspective Projections

- Multiplying by a value dependent on z_v causes the size of objects to vary depending on their distance from the origin.
- Note that a value $z_v = c_z$, will cause problems as the projected point will be at infinity i.e. objects are infinitely large at the view plane.
- Normally we will clip points for which $z_v \leq c_z$.
- We can write the transformation in matrix form:

$$J_{Perspective} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{c_z} & 1 \end{bmatrix}$$

- This gives us the coordinate $[x_v \; y_v \; 0 \; 1 - \frac{z_v}{y_z}]^T$, which when homogenized gives us the point $[\frac{x_v}{-\frac{z_v}{c_z}+1} \; \frac{x_v}{-\frac{z_v}{c_z}+1} \; 0 \; 1]^T$.

Mapping into a 2D Viewport

- Having constructed a normalized view volume in the shape of a regular parallelepiped, a 2D image can be constructed by projecting the volume orthographically.
- In the process of doing this, various other operations may be carried out:
  - Invisible edges and surfaces may be eliminated, and coloring and shading algorithms may be applied to produce a more realistic image.
  - Techniques for rendering are applied on the final image.

End of CH5!!!

- **Thank You!!!**

Chapter 6- Computer Animation

Computer Animation

1. Raster Methods for Computer Animation
2. Design of Animation Sequences
3. Traditional Animation Techniques
4. Computer-Animation Languages
5. OpenGL Animation Procedures

Computer Animation

- Computer-graphics methods are now commonly used to produce animations for a variety of applications, including **entertainment** (motion pictures and cartoons), **advertising, scientific** and **engineering studies**, and **training** and **education**.
- The term **computer animation** generally refers to any time sequence of visual changes in a picture.
- In addition to changing object positions using translations or rotations, a computer-generated animation could display time variations in object size, color, transparency, or surface texture.

Computer Animation

- Two basic methods for constructing a motion sequence are **real-time animation** and **frame-by-frame animation.**

- In a **real-time** computer-animation, each stage of the sequence is viewed as it is created.
- Thus the animation must be generated at a rate that is compatible with the constraints of the refresh rate.

- For a **frame-by-frame** animation, each frame of the motion is separately generated and stored.
- Later, the frames can be recorded on film, or they can be displayed consecutively on a video monitor in “real-time playback” mode.

- Simple animation displays are generally produced in real time, while more complex animations are constructed more slowly, frame by frame.

Raster Methods for Computer Animation

- Most of the time, we can create simple animation sequences in our programs using real-time methods.
- In general, though, we can produce an animation sequence on a raster-scan system one frame at a time, so that each completed frame could be saved in a file for later viewing.
- The animation can then be viewed by cycling through the completed frame sequence, or the frames could be transferred to film.

Raster Methods for Computer Animation

- If we want to generate an animation in real time, however, we need to produce the motion frames quickly enough so that a continuous motion sequence is displayed.
- For a complex scene, one frame of the animation could take most of the refresh cycle time to construct.
- In that case, objects generated first would be displayed for most of the frame refresh time, but objects generated toward the end of the refresh cycle would disappear almost as soon as they were displayed.

Raster Methods for Computer Animation

- For very complex animations, the frame construction time could be greater than the time to refresh the screen, which can lead to erratic motion and fractured frame displays.

- Because the screen display is generated from successively modified pixel values in the refresh buffer, we can take advantage of some of the characteristics of the raster screen-refresh process to produce motion sequences quickly.

Double Buffering

- One method for producing a real-time animation with a raster system is to employ two refresh buffers.
- Initially, we create a frame for the animation in one of the buffers. Then, while the screen is being refreshed from that buffer, we construct the next frame in the other buffer.
- When that frame is complete, we switch the roles of the two buffers so that the refresh routines use the second buffer during the process of creating the next frame in the first buffer.

Double Buffering

- This alternating buffer process continues throughout the animation.
- Graphics libraries that permit such operations typically have one function for activating the double buffering routines and another function for interchanging the roles of the two buffers.
- The most straight forward implementation is to switch the two buffers at the end of the current refresh cycle, during the vertical retrace of the electron beam.
- If a program can complete the construction of a frame within the time of a refresh cycle, say $\frac{1}{60}$ of a second, each motion sequence is displayed in synchronization with the screen refresh rate.

Double Buffering

- However, if the time to construct a frame is longer than the refresh time, the current frame is displayed for two or more refresh cycles while the next animation frame is being generated.

- For example, if the screen refresh rate is 60 frames per second and it takes $\frac{1}{50}$ of a second to construct an animation frame, each frame is displayed on the screen twice and the animation rate is only 30 frames each second.

- Similarly, if the frame construction time is $\frac{1}{25}$ of a second, the animation frame rate is reduced to 20 frames per second because each frame is displayed three times.

Double Buffering

- Irregular animation frame rates can occur with double buffering when the frame construction time is very nearly equal to an integer multiple of the screen refresh time.

- A san example of this, if the screen refresh rate is 60 frames per second, then an erratic animation frame rate is possible when the frame construction time is very close to $\frac{1}{60}$ of a second, or $\frac{2}{60}$ of a second, or $\frac{3}{60}$ of a second, and so forth.

Double Buffering

- Because of slight variations in the implementation time for the routines that generate the primitives and their attributes, some frames could take a little more time to construct and some a little less time.
- Thus, the animation frame rate can change abruptly and erratically.
- One way to compensate for this effect is to add a small time delay to the program.
- Another possibility is to alter the motion or scene description to shorten the frame construction time.

Operations

- We can also generate real-time raster animations for limited applications using block transfers of a rectangular array of pixel values.
- This animation technique is often used in game-playing programs.
- A simple method for translating an object from one location to another in the *xy* plane is to transfer the group of pixel values that define the shape of the object to the new location.
- Two-dimensional rotations in multiples of 90° are also simple to perform.

Operations

- Sequences of raster operations can be executed to produce realtime animation for either two-dimensional or three-dimensional objects.
- We can also animate objects along two-dimensional motion paths using **colortable transformations.**
- Here we predefine the object at successive positions along the motion path and set the successive blocks of pixel values to color-table entries.
- The pixels at the first position of the object are set to a foreground color, and the pixels at the other object positions are set to the background color.

Operations

- The animation is then accomplished by changing the color-table values so that the object color at successive positions along the animation path becomes the foreground color as the preceding position is set to the background color (Figure 1).

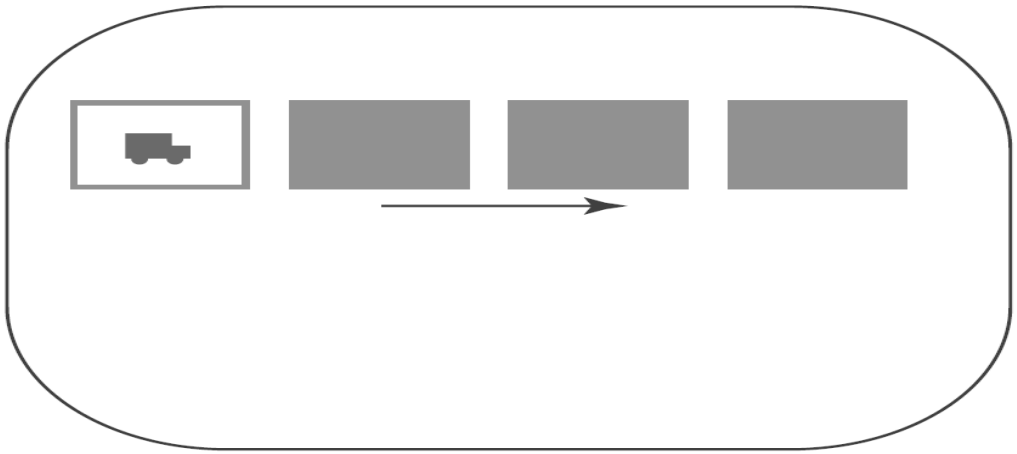


FIGURE 1
Real-time raster color-table animation.

Design of Animation Sequences

- Constructing an animation sequence can be a complicated task, particularly when it involves a story line and multiple objects, each of which can move in a different way.

- A basic approach is to design such animation sequences using the following development stages:
  - Storyboard layout
  - Object definitions
  - Key-frame specifications
  - Generation of in-between frames

Design of Animation Sequences

- The **storyboard** is an outline of the action. It defines the motion sequence as a set of basic events that are to take place.

- Depending on the type of animation to be produced, the storyboard could consist of a set of rough sketches, along with a brief description of the movements, or it could just be a list of the basic ideas for the action.

- Originally, the set of motion sketches was attached to a large board that was used to present an overall view of the animation project. Hence, the name “storyboard.”

Design of Animation Sequences

- An **object definition** is given for each participant in the action.
- Objects can be defined in terms of basic shapes, such as polygons or spline surfaces.
- In addition, a description is often given of the movements that are to be performed by each character or object in the story.

Design of Animation Sequences

- A **key frame** is a detailed drawing of the scene at a certain time in the animation sequence.
- Within each key frame, each object (or character) is positioned according to the time for that frame.
- Some key frames are chosen at extreme positions in the action; others are spaced so that the time interval between key frames is not too great.
- More key frames are specified for intricate motions than for simple, slowly varying motions.
- Development of the key frames is generally the responsibility of the senior animators, and often a separate animator is assigned to each character in the animation.

Design of Animation Sequences

- **In-betweens** are the intermediate frames between the key frames.
- The total number of frames, and hence the total number of in-betweens, needed for an animation is determined by the display media that is to be used.
- Film requires 24 frames per second, and graphics terminals are refreshed at the rate of 60 or more frames per second.
- Typically, time intervals for the motion are set up so that there are from three to five in-betweens for each pair of key frames.
- Depending on the speed specified for the motion, some key frames could be duplicated.
- As an example, a 1-minute film sequence with no duplication requires a total of 1,440 frames. If five in-betweens are required for each pair of key frames, then 288 key frames would need to be developed.

Design of Animation Sequences

- There are several other tasks that may be required, depending on the application.
- These additional tasks include motion verification, editing, and the production and synchronization of a soundtrack.
- Many of the functions needed to produce general animations are now computer-generated.

Traditional Animation Techniques

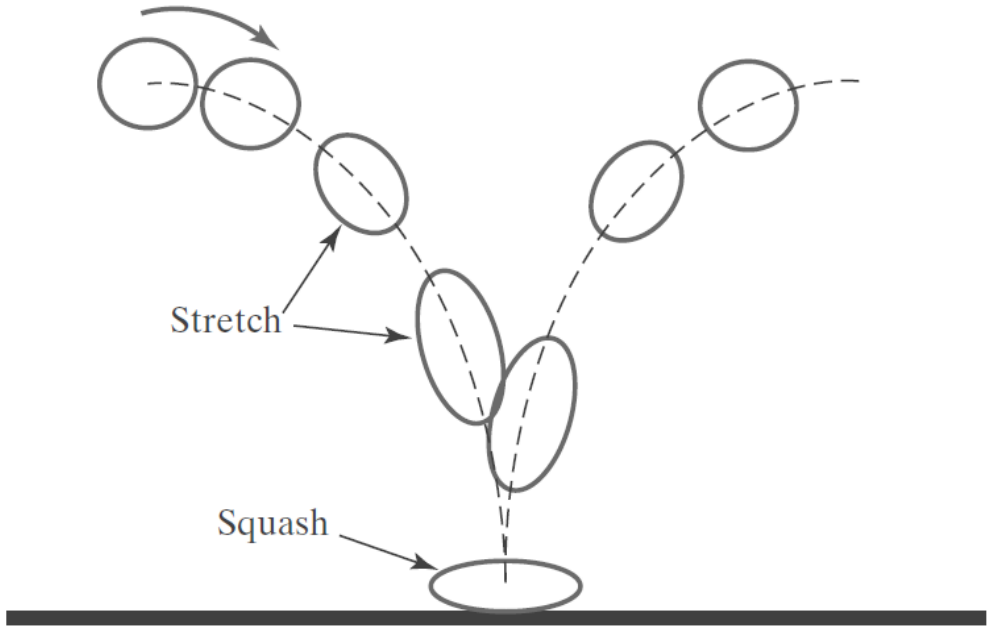


FIGURE 4
A bouncing-ball illustration of the "squash and stretch" technique for emphasizing object acceleration.

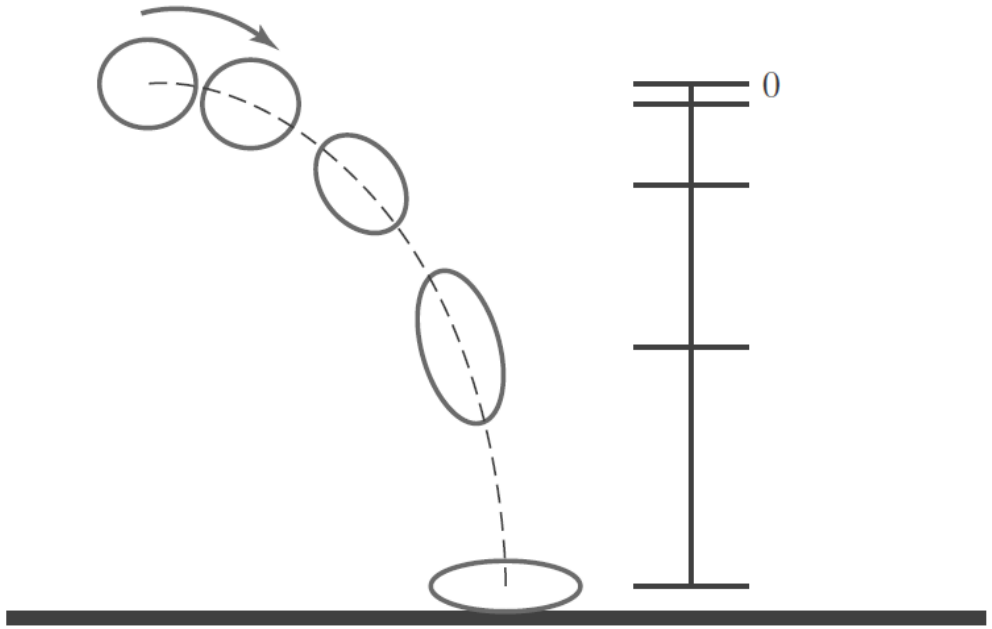


FIGURE 5
The position changes between motion frames for a bouncing ball increase as the speed of the ball increases.

Computer-Animation Languages

- We can develop routines to design and control animation sequences within a general-purpose programming language, such as C, C++, Lisp, or Fortran, but several specialized animation languages have been developed.

- These languages typically include a graphics editor, a key-frame generator, an in-between generator, and standard graphics routines.

- The graphics editor allows an animator to design and modify object shapes, using spline surfaces, constructive solid geometry methods, or other representation schemes.

OpenGL Animation Procedures

- Double buffering operations can be specified

End of Ch6

- Thank You!