LECTURE - I

Computer Graphics with OpenGL - Hearn and Baker

Computer Graphics : Principles and Practice by Foley, Van Dam

TUTORIAL - I

LECTURE - II

World representation → Convert to Graphics primitives → Graphics API → Framebuffer Image

Graphics is concerned with appearance of the 3D world to a camera.
∴ Only outer surfaces of objects important, not interiors.
Hence, only 1D or 2D primitives used.
Points : 2D or 3D
Lines : specified using end points
Triangles / polygons : specified using vertices.

Attributes :- Colour, point width, line width, line style, fill, fill pattern, line (two end points), etc.

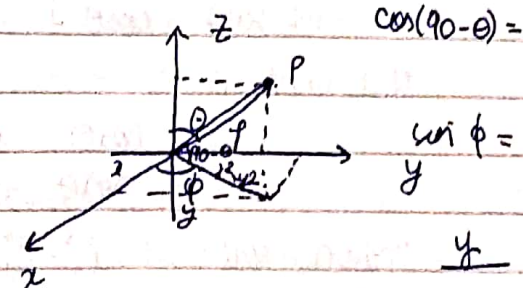
Points undergo transformations - translations, rotation, scaling and shearing.

→ 3D Coordinates : Linear or polar

$r = \sqrt{x^2+y^2+z^2}$

$\cos\theta = \frac{z}{r} \Rightarrow \theta = \cos^{-1}\left(\frac{z}{r}\right)$

$\tan\phi = \frac{y}{x} \Rightarrow \phi = \tan^{-1}\left(\frac{y}{x}\right)$



$\therefore z = r\cos\theta,\ y = r\sin\theta\sin\phi,\ x = r\sin\theta\cos\phi$

→ Translation

$P = (x, y, z)$, translated by $(a, b, c) = T$

$\therefore P' = (x+a, y+b, z+c) = P+T$

→ Scaling

Scale along x, y, z by s, u, t.

$\therefore x' = sx,\ y' = uy,\ z' = tz$

$\therefore P' = SP$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & t \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

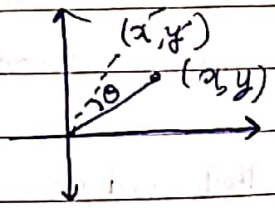
Invariants :- Parallelism, ratios of length in any direction (angles also for uniform scaling).

→ Rotation (we consider only CCW rotation as +ve)

2D:

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

$P' = R_\theta P$



$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Note that,

$$R^{-1} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = R^T$$

Orthonormal: If $\vec{C_i} \cdot \vec{C_j} = 0$ if $i \neq j$ and 1 if $i = j$ $\forall$ colvrs $\vec{C_i}, \vec{C_j}$

$\therefore R$ is orthonormal.

Invariants : distances, angles, parallelism.

3D:

Rotation can be about any axis, L.

About L, distances from L remain constant.

Each point moves in a circle, on a plane ⊥ L, with centre on L.

Now, about z-axis, if we rotate by θ, then matrix is

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This is an orthonormal, length preserving transformation.

Similarly,

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

→ (??)

It is orthonormal and norm preserving.

→ Shearing

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & x_y & x_z \\ y_x & 1 & y_z \\ z_x & z_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Atleast One of $x_y, x_z, y_x, y_z, z_x, z_y$ is non-zero.

Rectangles can become llgms, but not general quadrilaterals.

Invariants = ⊗ Parrallelism, ratios of length in any direction.

→ Reflection
- -ve entries denote reflection.

Out of all the transforms, only translation cannot be represented as a matrix vector product in 3-D

→ HOMOGENOUS COORDINATES
A non-zero scale factor ω is added to each coordinate.
e.g. a 2-D point is represented as $\begin{bmatrix} x \\ y \\ \omega \end{bmatrix}$

$$\begin{pmatrix} x \\ y \\ \omega \end{pmatrix} \equiv \left(\frac{x}{\omega}, \frac{y}{\omega}\right)$$

Simplest ω is $\omega = 1$.

$$\therefore \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a\ 20_x \\ 20_y \\ 20 \end{bmatrix} \equiv \begin{bmatrix} 20,000x \\ 20,000y \\ 20,000 \end{bmatrix}$$ (as actual point is $\left(\frac{x}{\omega}, \frac{y}{\omega}\right)$)

In such a system, translation can be written as a matrix product.

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

$\Rightarrow P' = TP$

∴ In homogenous coordinate, ~~any translation~~ linear transformations can be written as a matrix product.

e.g. Rotation followed by translation followed by scaling followed by another rotation :- $P' = R_2 S T R_1 P$

In 3-D matrix has 4 dimension $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \equiv \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$ (excluding zero)

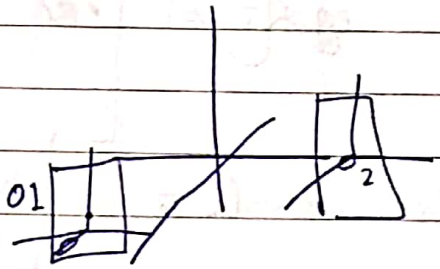
For 2D, all transforms are 3×3 ie, and last row is [0 0 1]
Similarly, in 3D, most transforms are 4×4 and last row is [0 0 0 1]

Manipulation of last rows allow us to represent projection.

In homogenous coordinate, $\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$ represents the points at ∞. These are used in projective geometry.

TUTORIAL-II

Object reference coordinate (ORC)
World reference coordinate (WRC)



ORC
↓ M (matrix of transform b/w ORC and WRC)
WRC
↓ View (view reference matrix)
Camera

LECTURE-III

Homogenous Coordinates

In 3-D, we use homogenous coordinates, so rotation matrices are 4D matrices.

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation : $T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$

In general, most transformations are not commutative.

eg. $RT = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & a\cos\theta - b\sin\theta \\ \sin\theta & \cos\theta & & \end{bmatrix}$

$$R = \begin{bmatrix} R' & 0 \\ 0 & 1 \end{bmatrix}, \quad T = \begin{bmatrix} I & t' \\ 0 & 1 \end{bmatrix}, \quad R' = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\vec{t'} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$\therefore RT = \begin{bmatrix} R' & R't \\ 0 & 1 \end{bmatrix}$$

$$TR = \begin{bmatrix} R' & t' \\ 0 & 1 \end{bmatrix}$$

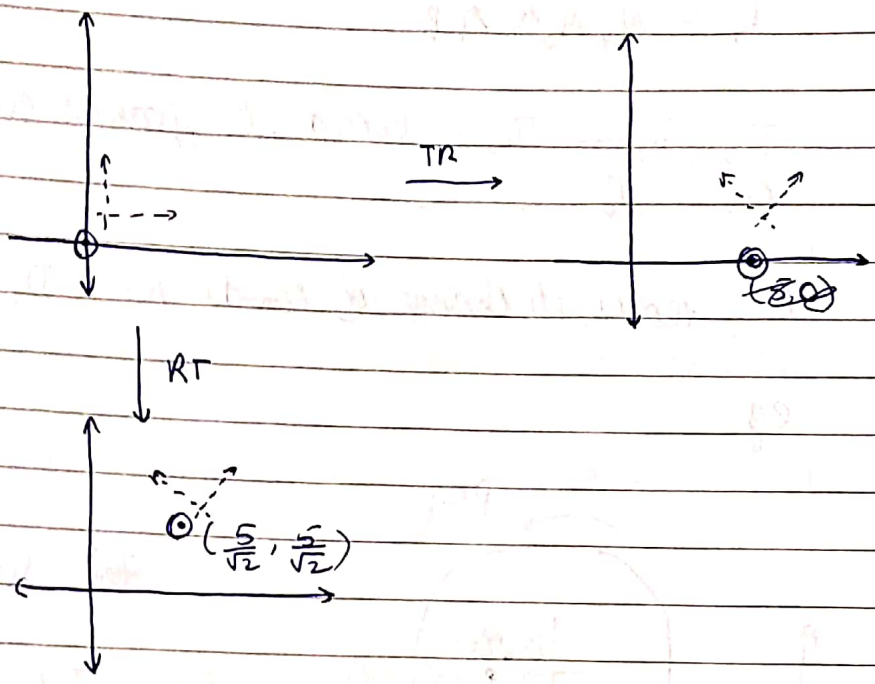
$\therefore RT \neq TR$

If $RT = TR$, then
1. $R' = I$
2. $t' = 0$
3. $R't' = t'$ i.e. 1 is an eigenvalue of R'

i.e. translation is along axis of rotation

In general, translations are commutative ($T_1 T_2 = T_2 T_1$)
Scaling is commutative ($S_1 S_2 = S_2 S_1$)
But rotations along different axes are not commutative

TR, RT :-



In frame of reference, the frame undergoes opposite transformation.

→ frame (rotation is done locally)

$P' = TRP$

← point

Local world and global world coordinate system:

In place or local transformations :- we transform the object about the centroid or any origin of local coordinate system. e.g. for ~~so~~ local rotation, we can think of it as translating ~~to origin~~ local origin to global origin, rotating, then translating back.

∴ $T(c)$ M $T(-c)$, where c is position vector of local origin.

→ Points and Frames

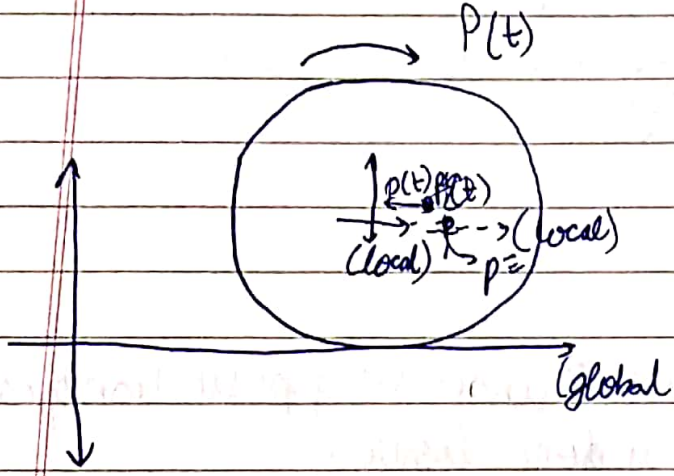
Frames - local / transient coordinate systems.

$P_4 = M_4 M_3 M_2 M_1 B$

$\Pi_4, \Pi_3 \dots \Pi_0$:- coordinates frames corresponding to $P_4, \dots P_0$.

M_i represents change of coods from Π_{i-1} to Π_i (??)

e.g.



Find $P(t)$

$P(t) = T_1 R T_2 (p(t)) P^{\equiv}$

$\therefore P(t) = T_1(t) \; @R(\theta(t)) T_2(p(t)) P^{\equiv}$

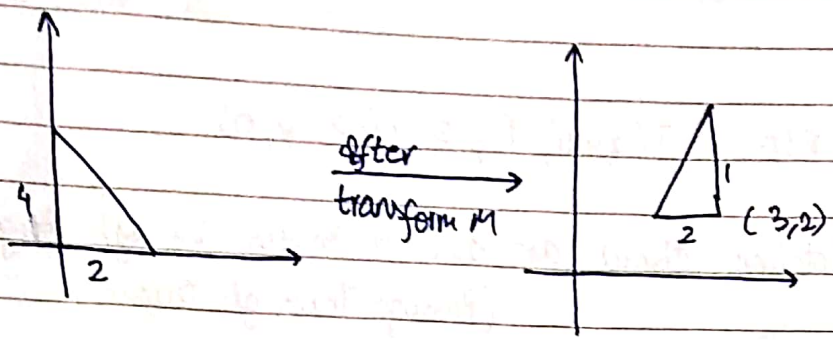
$T_1(t) = T(r\theta(t), r) = T(r\omega t, r)$

$R(\theta(t)) = R_z(\omega t)$

$T_2(t) = T(\theta, 0, 1) \; T(p(t), 0) = T(vt, 0)$

$P^{\equiv} = [0, 0, 1]^T$ (in local frame, head has coods (0,0))

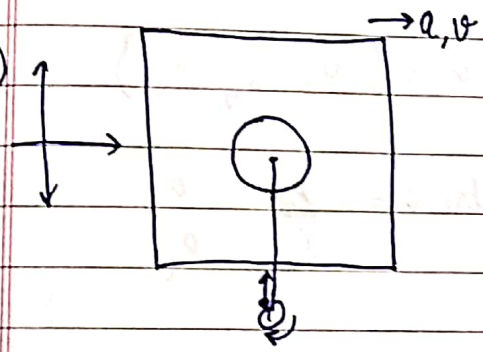
LECTURE-IV



Transformation in order = scaling, rotation then translation.

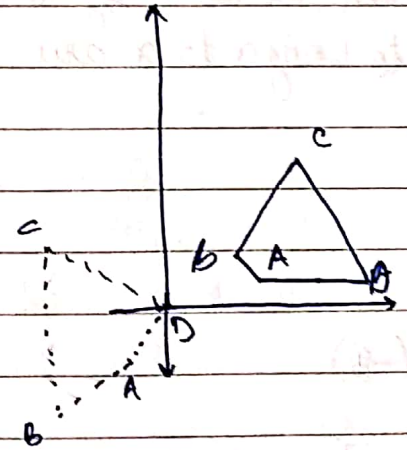
$\therefore M = T(3,2)\, R(90^\circ)\, S\left(\frac{1}{2}, \frac{1}{2}\right) \equiv T(3,2)\, S\left(\frac{1}{2}, \frac{1}{2}\right) R(90^\circ)$

Q.)



Model motion of ant.

→ A Transformation Problem



Shift D to origin then rotate s.t BC is ∥ to y-axis.

The rotation matrix can be found as

$$\begin{bmatrix} \text{—} \\ -\frac{\vec{BC}}{\|\vec{BC}\|} \end{bmatrix} \rightarrow \text{row} \perp \text{to } \vec{BC}$$

$$= \begin{bmatrix} u_y & -u_x \\ -u_x & u_y \end{bmatrix} \text{ or } \begin{bmatrix} -u_y & u_x \\ u_x & u_y \end{bmatrix}$$

↳ Rotation about an axis parallel to z, through $(x, y, 0)$.

Translate $(x, y, 0)$ to $(0,0,0)$, rotate about z, then translate back

$$\therefore R(\theta) = T(x,y,0)\, R_z(\theta)\, T(-x,-y,0)$$

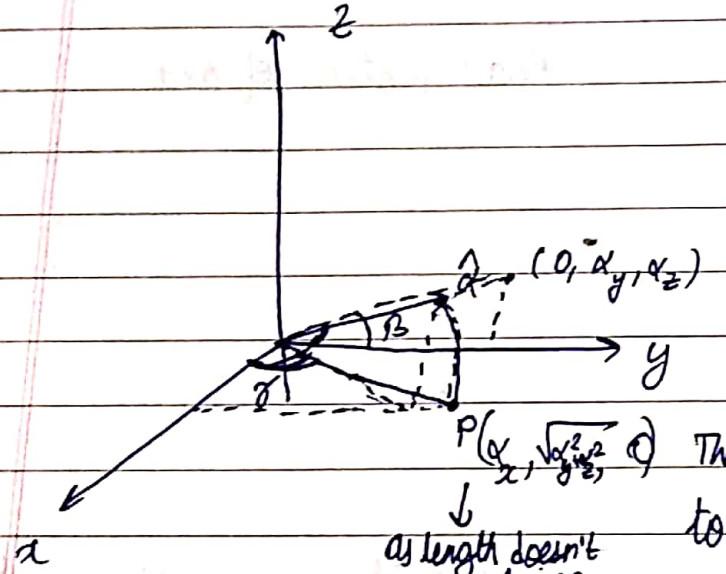
↳ 3D Rotation about an axis ~~Passing Through Origin~~ (Passing Through Origin)

$R_\alpha(\theta) = ?$

3-step process :- ~~Rotate~~ Align α to $\hat{x}$.
Rotate about x by θ.
Rotate back.

To $\quad \therefore R_\alpha(\theta) = R_{\alpha x}^{-1}\, R_x(\theta)\, R_{\alpha x}$

$\alpha = (\alpha_x, \alpha_y, \alpha_z)$



$$\tan\beta = \frac{\alpha_z}{\alpha_y}$$

∴ To project α on xy plane, we rotate by $-\beta$ along x-axis. Then, we rotate by $-\gamma$ along z-axis to bring α to x-axis.

↓ as length doesn't change

$$\tan\gamma = \frac{\sqrt{\alpha_y^2 + \alpha_z^2}}{\alpha_x}$$

$$\therefore R_{\alpha x} = R_z(-\gamma)\, R_x(-\beta)$$

$$\Rightarrow R_{\alpha x}^{-1} = [R_x(-\beta)]^{-1} [R_z(-\gamma)]^{-1} = R_x(\beta)\, R_z(\gamma)$$

$\sec\gamma =$ $\quad\quad \sqrt{2} = \tan\gamma$

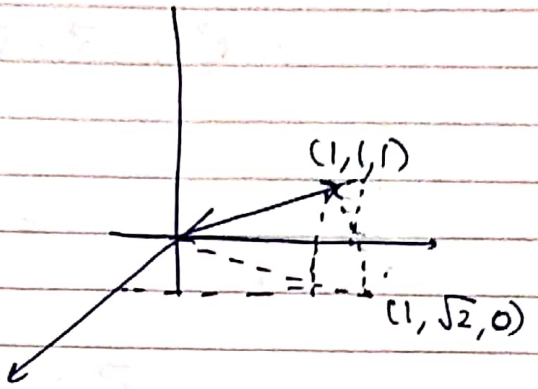
Alternatively, we can find out $R_{\hat{d}x}$ by using the fact that $\hat{d}$ will be the vector that becomes x-axis.

e.g. rotation about $\hat{d} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$\therefore \tan\beta = \frac{1}{1} \Rightarrow \beta = \frac{\pi}{4}$

$\tan\gamma = \frac{\sqrt{2}}{1}$

$\Rightarrow \gamma = \tan^{-1}(\sqrt{2})$



$\therefore R_x(-\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$, $R_z(-\tan^{-1}(\sqrt{2})) = \begin{bmatrix} 1/\sqrt{3} & 2/\sqrt{6} & 0 \\ -2/\sqrt{6} & 1/\sqrt{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\therefore R_{\hat{d}x} = R_z(-\gamma)\, R_x(-\beta) = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ -2/\sqrt{6} & 1/\sqrt{6} & 1/\sqrt{6} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

~~Or, we can find~~

Method - II :-

We directly know the first row, as it is the axis which rotates onto the x-axis i.e it is $\hat{d}$-axis.

$\therefore R_{\hat{d}x} = \begin{bmatrix} — & r_1 & — \\ — & r_2 & — \\ — & r_3 & — \end{bmatrix}$, $r_1 = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \Rightarrow = \vec{u}$

~~Also~~ Now, find $\vec{v}$ not $\parallel$ to $\vec{u}$.

$\therefore \vec{r_2} = \frac{\vec{r_1} \times \vec{v}}{\|\vec{r_1} \times \vec{v}\|}$ and $\Rightarrow R_{\hat{d}x} = \begin{bmatrix} \vec{d} \\ \vec{d} \times \vec{v} \\ \vec{r_1} \times \vec{r_2} \end{bmatrix}$

$\vec{r_3} = \frac{\vec{r_1} \times \vec{r_2}}{\|\vec{r_1} \times \vec{r_2}\|}$

~~$R_{\hat{d}x}$ is not~~

Is $R_{\hat{d}x}$ unique ? ↳ No

$$\therefore \quad \vec{a} = \frac{1}{\sqrt{3}}[1, 1, 1]^T = \vec{r_1}$$

Let $\vec{v} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$$\therefore \vec{r_2} = \vec{a} \times \vec{v} = \frac{1}{\sqrt{3}} \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{vmatrix}$$

$$= \frac{1}{\sqrt{3}} \left(-\hat{j}(-1) + \hat{k}(-1) \right)$$

$$\vec{r_2} = \frac{1}{\sqrt{3}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \Rightarrow \hat{r}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$\vec{r_3} = \vec{r_1} \times \vec{r_2} = \frac{1}{\sqrt{3}} \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 1 & 1 & 1 \\ 0 & 1 & -1 \end{vmatrix}$$

$$= \hat{i}(-2) - \hat{j}(-1) + \hat{k}(1)$$

$$= \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}$$

$$\therefore \hat{r}_3 = \frac{1}{\sqrt{6}} \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}$$

$$\therefore R_{ax} = \begin{bmatrix} \hat{r}_1^T \\ \hat{r}_2^T \\ \hat{r}_3^T \end{bmatrix}$$

↳ Rotation About Arbitrary Axis & Point

We translate point to origin, then rotate about α then translate back.

$$\therefore M(\theta) = T(x,y,z)\, R_{\alpha}(\theta)\, T(-x,-y,-z)$$

$$= T^{-1} R_{\alpha}(\theta) T$$

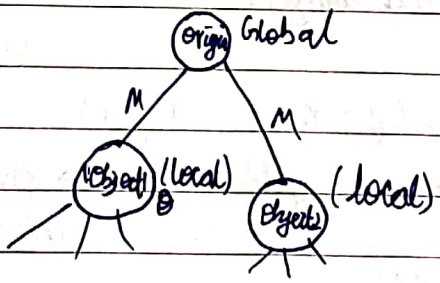
$$= T^{-1} R_{\alpha x}^{-1} R_{x}(\theta) R_{\alpha x} T$$

→ TRANSFORMING LINES

Linear transforms conserve collinearity therefore we just transform the end points, and then connect them with a line.

LECTURE - IV

→ Modelling



Different coordinates :- Object coords, world coords, camera coords, etc.

Object reference :- ORC (object reference coordinate)
World coordinates (WC)
View reference coordinate (VRC)
Normalized coordinates (NC)

$P_{VRC} = V \; M \; P_{ORC}$

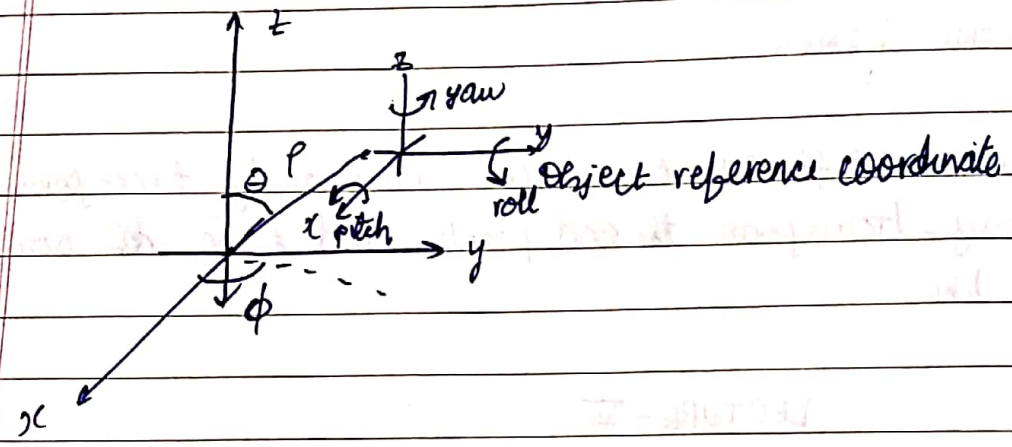
P_{ORC} → in ORC

M → transform to WC

V → transforms to VRC

↳ modelling :- ORC to WC.

In polar coordinates :-



~~steps :- start with both axes aligned~~

T_r

$\therefore M = T(\rho, \theta, \phi) \cdot R_z(y) \; R_x(p) \; R_y(r)$

yaw pitch roll → preserve order.
(otherwise, rotations are not proper)

$= T(\rho, \theta, \phi) \; R_z(-\phi)$

OR

Now,

$T(\rho, \theta, \phi) = R_z(-\phi) \; R_y(\theta) \; T(0, 0, \rho) \; R_y(\theta - \theta) \; R_z(\phi)$

Rotate back → translate along z by ρ → align z to $\vec{r}$ i.e translation direction

→ Hierarchy of Transformations

A hierarchy of transformations needed to setup the world and camera.

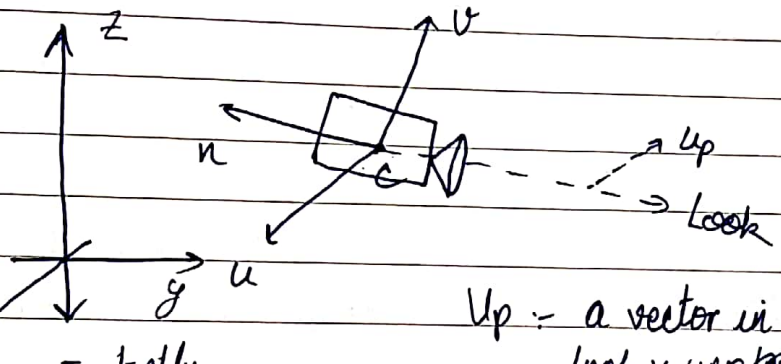
OpenGL transforms ~~local coordinate~~ ORC directly to normalized projection coordinates.

PVM :- Projection, View, Modelling

→ View orientation or Viewing

Transform WC to VRC.

VRC ⊙ u, v, n :- Camera center, look point and up vector specified in world coordinates.



Up :- a vector in n, v plane

look x up gives u, $n \times u$ gives v

Initially u :- aligned with x, v :- aligned with y, n :- aligned with z

∴ Translate origin to C(x, y, z)

Rotate to align (-z-axis to look vector. ⊙ $(-\hat{L})$

Rotate ⊙ to align y-axis to up.

Note that rotation matrix can be obtained by using the fact that columns represent the ⊙ vectors which allign to the axes.

∴ $\begin{bmatrix} u & n \times u = v & n \end{bmatrix} = R$

$A = T(x,y,z)\ R$

A transforms VRC to WC.

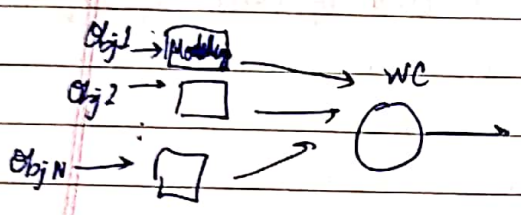
$\therefore P_{WC} = A\,P_{VRC}$

$\Rightarrow P_{VRC} = A^{-1} P_{WC}$

$= R^T T(-x,-y,-z)\ P_{WC}$

Scene has one viewing matrix, but each object has a modelling matrix.

LECTURE-III



PROJECTIONS

Projection involves projectors starting from 3D points and hitting the 2D projection plane, forming the image of the point.
IRL, projectors are photons.

↳ Parallel Projection :- All projectors parallel ie have same direction of projection. (DOP)

↳ Perspective Projection :- All projectors pass through a point, called the centre of projection e.g. pinhole.
↳ (COP).

Parallel projection :- Perspective projection with COP at infinity.

→ Parallel Projection

⇒ Orthographic Projection :- Projection plane ⊥ to D.O.P.
e.g. if DOP is parallel to the axes : plan, elevation, side elevation.
If P.P intersect all axes at equal distance : isometric projection.
Here lengths ∥ PP are preserved. Good approx. for camera with long focal length.

⇒ Oblique Projection :- DOP makes some angle with PP.
e.g. Cavalier : 45° angles b/w DOP and PP. Length along depth axis preserved.

Cabinet : $\tan^{-1}(2)$ angle b/w DOP and PP. Length along depth axis halved.

Orthographic projection equation :-

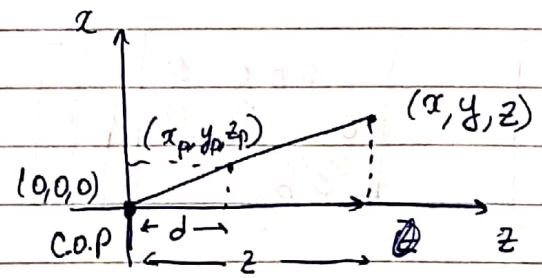
$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ Z \\ w \end{bmatrix}$$

→ Perspective Projections

Can be characterized by number of vanishing points (projection of points at infinity).
Depends on the number of

There can be 1-point, 2-point, 3-point projections.

Geometry :-



$$\frac{x_p}{d} = \frac{x}{z} \Rightarrow x_p = \frac{x}{\frac{z}{d}}$$

$$\frac{y_p}{d} = \frac{y}{z} \Rightarrow y_p = \frac{y}{z_d}$$

$$z_p = d \Rightarrow z_p = d$$

$$\frac{x_p}{z_p} = \frac{1}{z}$$

$\therefore x_p = \frac{x}{\omega}, \; y_p = \frac{y}{\omega}, \; z_p = \frac{z}{\omega}$, where $\omega = \frac{z}{d}$

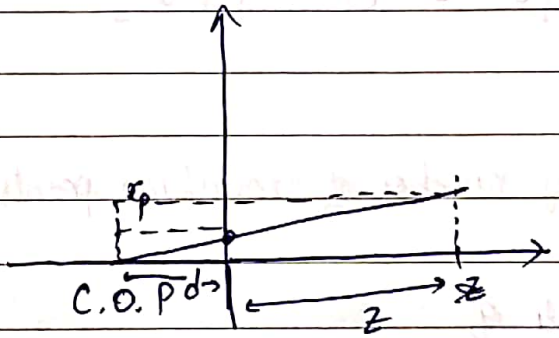
In homogenous coordinates

$$\begin{bmatrix} x_p/\omega \\ y_p/\omega \\ z_p/\omega \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x_p \\ y_p \\ z_p \\ \omega \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix} \Rightarrow \begin{bmatrix} x_p \\ y_p \\ z_p \\ \frac{z}{d} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} x_p \\ y_p \\ z_p \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Coordinates scaled down proportional to the depth of z values.

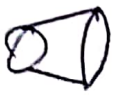
other scenario :- when c.o.p at $(0, 0, -d)$



$$\frac{x_p}{d} = \frac{x}{z+d} \Rightarrow = \frac{x}{1+\frac{z}{d}}, \quad z_p = 0, \; y_p = \frac{y}{1+\frac{z}{d}}$$

$$\therefore \begin{bmatrix} x_p \\ y_p \\ z_p \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Orthographic projection matrix is a special case when $d \to \infty$.



→ Volume of visibility

Cameras have finite fields of view in horizontal and vertical directions.

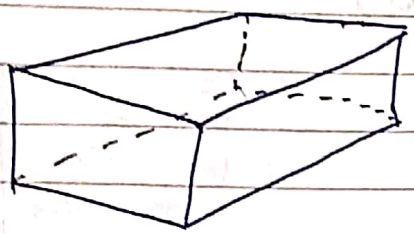
∴ The shape of visible space is ~~cylinder~~ cube for orthographic projections and cones for perspective projections.

View volume is taken as rectangular specified by 6 planes.

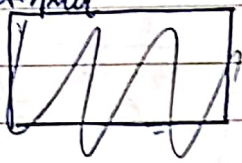
View volume is volume of visible space.

↓

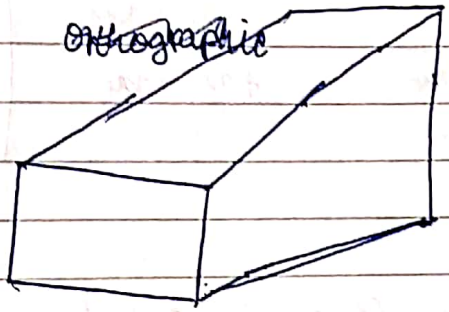
left, right, top, bottom, near, far.



→ Orthographic



~~Orthographic~~

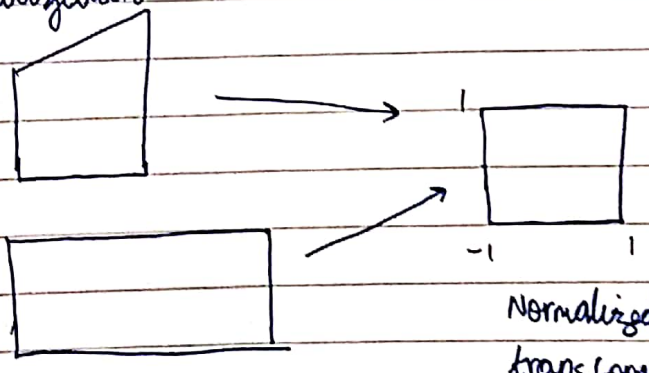


→ Perspective

An ideal pin-hole camera has ∞ focal length

Finite focal length cameras simulate real world

Normalization :-

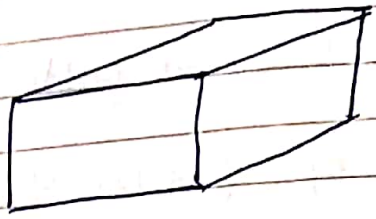


Normalized transform.

x→ ↓ y

-1, +

Canonical view volume :-



$-1 \le x \le 1$

$-1 \le y \le 1$

$-1 \le z \le 1$

For orthographic view volume, we can reach the canonical view volume through appropriate scaling.

∴ orthographic normalizing matrix :-

$$\begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\left(\frac{right+left}{right-left}\right) \\ 0 & \frac{2}{top-bottom} & 0 & -\left(\frac{top+bottom}{top-bottom}\right) \\ 0 & 0 & \frac{2}{far-near} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

∴ $(0, 0, -near)$ maps to $(0, 0, 1)$ and $(0, 0, -far)$ maps to $(0, 0, -1)$

We drop z and use (x, y) as the (normalized) window coordinates.