EECE 478

Linear Algebra and
3D Geometry

Learning Objectives

- Linear algebra in 3D
  - Define scalars, points, vectors, lines, planes
  - Manipulate to test geometric properties
- Coordinate systems
  - Use homogeneous coordinates
  - Create coordinate transforms
  - Distinguish rigid body, angle-preserving and affine transforms

Learning Objectives

- Represent transforms in homogeneous coordinates
  - Rotation, translation and scaling
  - Combine to move fixed points or rotation axes
  - Quaternion rotations
- Manipulate transform matrices in OpenGL

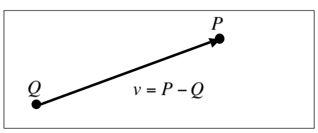
Vocabulary

- Scalar
- Point
- Line
- Vector
- Plane
- Dot product
- Cross product
- Normal
- Coordinate system
- Frame
- Homogeneous coordinates
- Rotation
- Translation
- Scaling
- Quaternions

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Geometric Objects

- Fixed *Point* in space
- *Scalar* is a real number
- *Vector* has direction and magnitude



©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Defined Operations

- Point + Vector = Point
- Point - Point = Vector
- Scalar * Vector = Vector
- Vector + Vector = Vector
- Vector - Vector = Vector
- Vector • Vector = Scalar (dot)
- Vector × Vector = Vector (cross)

 UBC Dept. of ECE

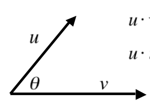
Inner Product

- Linear measure of a vector space
- Constraints:
  - $<u,v>$ is a scalar
  - $<\alpha u,v> = \alpha<u,v>$
  - $<u,v+w> = <u,v> + <u,w>$
  - $<v,v> \geq 0$ (*magnitude* of v or $||v|| = \sqrt{<v,v>}$)
  - $<v,v> = 0$ only if $v = 0$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Dot Product

- Scalar combination of vector lengths and internal angle
- Perpendicular vectors have inner product of 0

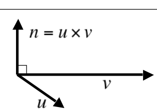


$$u \cdot v = |u||v|\cos(\theta)$$
$$u \cdot u = |u|^2$$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Cross Product

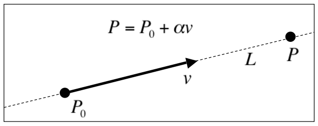
Outer product of vectors
- Cross product n is perpendicular to u and v
- Right hand coordinate system
- n is *normal* to plane of u,v



Line

- Line L passes through point P_0 with direction v
  - L is *all* points P

$$P = P_0 + \alpha v$$

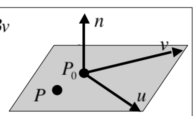


©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Plane

- Plane T passes through point P_0 with directions u and v
  - T is *all* points P
  - n is *normal* vector

$$P = P_0 + \alpha u + \beta v$$
$$n = u \times v$$
$$0 = (P - P_0) \cdot n$$



©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Coordinate System

- Any three non-coplanar vectors v_1, v_2, v_3 define a *coordinate system (vector space)*
  - *any* vector w is *uniquely* defined as a linear combination of *basis* vectors v_1, v_2, v_3

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$
$$w = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Frame

- Basis vectors v_1, v_2, v_3 and an origin P_0 define a *frame*
  – *any* point P is *uniquely* defined by a vector w added to the origin P_0

$$P = P_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

$$P = P_0 + \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Representation

Given a frame: (P_0, v_1, v_2, v_3)

– *representation* of point P is $(\alpha_1, \alpha_2, \alpha_3)$

– *representation* of vector w is $(\alpha_1, \alpha_2, \alpha_3)$

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

$$P = P_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

Cartesian Frame

Orthonormal basis

– basis vectors mutually perpendicular

– basis vectors all have magnitude 1

Euclidean basis: $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$

$$e_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

$$e_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$$

$$e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

Homogeneous Coordinates

- Don't want to confuse points and vectors
  - representations should be different
- *N.B.* Points refer to origin, vectors don't

Solution:
- Use a 4-dimensional coordinate system

Homogeneous Point

Four component *homogeneous coordinate*
- First three components refer to basis
- Fourth component refers to origin

$$P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0$$

$$P = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

Homogeneous Vector

Don't include origin
- Fourth component is zero

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

$$w = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

Change of Frame(1)

Express frame F_2 in coordinates of F_1

$$
\begin{aligned}
F_1 &= (P_0, v_1, v_2, v_3) \\
F_2 &= (Q_0, u_1, u_2, u_3) \\
\\
u_1 &= \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3 \\
u_2 &= \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3 \\
u_3 &= \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3 \\
Q_0 &= \gamma_{41} v_1 + \gamma_{42} v_2 + \gamma_{43} v_3 + P_0
\end{aligned}
$$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Change of Frame(2)

The same in matrix form

$$
\begin{aligned}
F_1 &= (P_0, v_1, v_2, v_3) \\
F_2 &= (Q_0, u_1, u_2, u_3)
\end{aligned}
$$

$$
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}
$$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Change of Frame(3)

Consider homogeneous coordinates

– $\mathbf{a}_1$ in F_1 and $\mathbf{a}_2$ in F_2

$$
\mathbf{a}_2^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathbf{a}_2^T \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \mathbf{a}_1^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}
$$

so,

$$
\mathbf{a}_1 = \mathbf{M}^T \mathbf{a}_2 \ \text{ and } \mathbf{a}_2 = \left(\mathbf{M}^T\right)^{-1} \mathbf{a}_1
$$

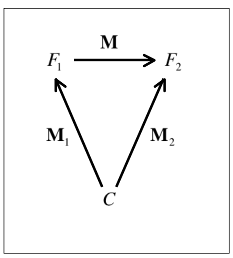
Change of Frame(4)

The change of frame *transformation* $\mathbf{M}^T$ transforms coordinates from F_2 to F_1

$$\mathbf{M}^T = \begin{bmatrix} \gamma_{11} & \gamma_{21} & \gamma_{31} & \gamma_{41} \\ \gamma_{12} & \gamma_{22} & \gamma_{32} & \gamma_{42} \\ \gamma_{13} & \gamma_{23} & \gamma_{33} & \gamma_{43} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Cartesian Frame



$$F_1 = \mathbf{M}_1\mathbf{C} \text{ and } F_2 = \mathbf{M}_2\mathbf{C}$$
$$F_2 = \mathbf{M}F_1 = \mathbf{M}\mathbf{M}_1\mathbf{C} = \mathbf{M}_2\mathbf{C}$$
$$\Rightarrow \mathbf{M}\mathbf{M}_1 = \mathbf{M}_2$$
$$\Rightarrow \mathbf{M} = \mathbf{M}_2\mathbf{M}_1^{-1}$$
$$\Rightarrow \mathbf{M}^T = \left(\mathbf{M}_1^{-1}\right)^T \mathbf{M}_2^T$$
and
$$a_1 = M^T a_2, \quad a_2 = \left(M^T\right)^{-1} a_2$$
$$a = M_2^T a_2, \quad a = M_1^T a_1$$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Affine Transformation

A *transformation* is a function from vertices (point or vector) to vertices

– transformation is *linear* or *affine* if and only if

$$f(\alpha u + \beta v) = \alpha f(u) + \beta f(v)$$
or
$$f(p + \beta v) = f(p) + \beta f(v)$$

⇒ need only transform endpoints

Canonical Transformations

- Translation *rigid body*
- Rotation *rigid body*
- Scaling *angle preserving*
- Shear

Represented by 4x4 matrix $\mathbf{M}$ in homogeneous coordinates

Translation(1)

$\mathrm{T}(\mathbf{d})$: Displace all points $\mathbf{p}$ by vector $\mathbf{d}$ to $\mathbf{p}'$

$$\begin{aligned}
\mathbf{p} &= \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \\
\mathbf{p}' &= \begin{bmatrix} x' & y' & z' & 1 \end{bmatrix}^T \\
\mathbf{d} &= \begin{bmatrix} \alpha_x & \alpha_y & \alpha_z & 0 \end{bmatrix}^T \\
\mathbf{p}' &= \mathbf{p} + \mathbf{d} \\
&= \begin{bmatrix} x+\alpha_x & y+\alpha_y & z+\alpha_z & 1 \end{bmatrix}^T
\end{aligned}$$

Translation(2)

In matrix form we can express this as:
- $\mathrm{T}(\alpha_x, \alpha_y, \alpha_z)$ is the *translation matrix*

$$\mathbf{p}' = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p} = \mathrm{T}(\alpha_x, \alpha_y, \alpha_z)\mathbf{p}$$

Translation(3)

$T(\alpha_x,\alpha_y,\alpha_z)^{-1}$: Simple inverses are important

$$T(\alpha_x,\alpha_y,\alpha_z)^{-1} = T(-\alpha_x,-\alpha_y,-\alpha_z) = \begin{bmatrix} 1 & 0 & 0 & -\alpha_x \\ 0 & 1 & 0 & -\alpha_y \\ 0 & 0 & 1 & -\alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling

$S(\beta_x,\beta_y,\beta_z)$: Stretch every vertex away from origin

$$\mathbf{p}' = \begin{bmatrix} \beta_x x & \beta_y y & \beta_z z & 1 \end{bmatrix}^T = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}$$

Scaling

$S(\beta_x,\beta_y,\beta_z)^{-1}$: Simple inverse

$$S(\beta_x,\beta_y,\beta_z)^{-1} = S(\frac{1}{\beta_x},\frac{1}{\beta_y},\frac{1}{\beta_z}) = \begin{bmatrix} \frac{1}{\beta_x} & 0 & 0 & 0 \\ 0 & \frac{1}{\beta_y} & 0 & 0 \\ 0 & 0 & \frac{1}{\beta_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation

$\mathrm{R}(\theta_x, \theta_y, \theta_z)$: Rotate by θ around each of x, y and z axes.

$$\mathrm{R}_x(\theta_x) = \mathrm{R}(\theta_x, 0, 0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

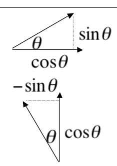
©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

2D Rotation(1)

$\mathrm{R}(\theta)$: Rotate by θ

$$\mathrm{R}(\theta)\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

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

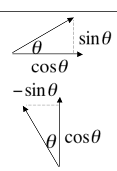


©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

2D Rotation(2)

$\mathrm{R}(\theta)$: Simple matrix derived from rotation of basis vectors

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



3D Rotation: Z Axis

$R_z(\theta)$: 2D rotation of (x,y)

– z axis is invariant

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

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

3D Rotation: Y Axis

$R_y(\theta)$: 2D rotation of (x,z)

– y axis is invariant

$$R_y(\theta_y) = R(0,\theta_y,0) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

3D Rotation

$R(\theta_x, \theta_y, \theta_z)$: General 3D rotation

– any rotation is combination of 3 axes

$$R(\theta_x,\theta_y,\theta_z) = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)$$
$$R^{-1}(\theta) = R(-\theta) = R^T(\theta)$$

Quaternions

- Generalization of complex numbers
  - $i^2 = j^2 = k^2 = ijk = -1$
  - $q = a + bi + cj + dk$
  - $q = (a,v)$
  - $q_1 q_2 = (a_1 a_2 - v_1 \bullet v_2,\ a_1 v_2 + a_2 v_1 + v_1 \times v_2)$
  - $q = (a,v) = (\cos(\theta/2),\ n \sin(\theta/2))$
    - n is unit vector
    - q is a rotation by θ about n

Concatenation

Transformations don’t occur in isolation

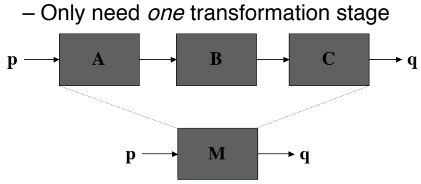
- Linearity means matrices can be combined

$$
\begin{aligned}
\mathbf{p}' &= \mathbf{A}\mathbf{p} \\
\mathbf{p}'' &= \mathbf{B}\mathbf{p}' \\
\mathbf{q} &= \mathbf{C}\mathbf{p}'' \\
&= \mathbf{C}(\mathbf{B}(\mathbf{A}\mathbf{p})) \\
&= (\mathbf{CBA})\mathbf{p} \\
&= \mathbf{M}\mathbf{p}
\end{aligned}
$$

Transform Pipeline

Transformations don’t occur in isolation

- Linearity means matrices can be combined
- Only need *one* transformation stage



Compound Transformations

Linear combination of translation, rotation, scaling and shear can produce any affine transformation.

$$\mathbf{M} = \mathbf{TRSH}$$

Rotation About Point

Strategy (*origin is fixed point for rotation*):

1. Move point p to origin
2. Rotate
3. Move point p back

$$\mathbf{M} = \mathbf{T}(p)\mathbf{R}(\theta)\mathbf{T}(-p)$$

Instance Transformation

Consider a prototype object

- fixed size, position and orientation
- usable as model if we can
  - make it desired size
  - change to desired orientation
  - move to desired position

$$\mathbf{M} = \mathbf{TRS}$$

OpenGL Transformations

OpenGL has *current transformation matrix* (CTM)

- global variable
- aka GL_MODELVIEW matrix
- set/modified by functions

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Transformation Operations

glLoadMatrix(M)	$\mathbf{C} \leftarrow \mathbf{M}$
glMultMatrix(M)	$\mathbf{C} \leftarrow \mathbf{CM}$
glLoadIdentity()	$\mathbf{C} \leftarrow \mathbf{I}$
glRotatef(θ,vx,vy,vz)	$\mathbf{C} \leftarrow \mathbf{CR}(\theta, vx, vy, vz)$
glTranslatef(tx,ty,tz)	$\mathbf{C} \leftarrow \mathbf{CT}(tx, ty, tz)$
glScalef(sx,sy,sz)	$\mathbf{C} \leftarrow \mathbf{CS}(sx, sy, sz)$

©2005, Lee Iverson <leei@ece.ubc.ca> UBC Dept. of ECE

Order of Transformations

- OpenGL operations post-multiply
  - last transformation called is first applied
  - sometimes useful to *save* matrix
- Matrix context saved by push/pop
  - `glPushMatrix()` – push CTM onto stack
  - `glPopMatrix()` – pop CTM off stack