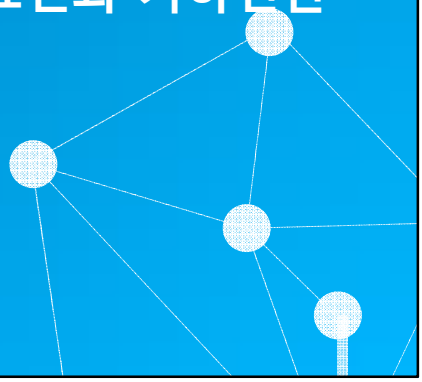
04 CHAPTER

물체의 표현과 기하변환



4장. 학습 내용

- 3차원 물체의 표현 방법
- 기본적인 수학
- 2차원 기하변환
- 3차원 기하변환
- 3차원 복합변환
- 기하 변환의 구현

2

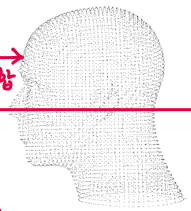
4.1 3차원 물체의 표현 방법

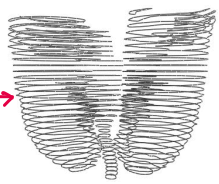
- 3차원 데이터의 다양한 유형
- 3차원 물체의 표면 기반 표현 (Boundary Surface Representation)
- 표면 모델의 관리 방법

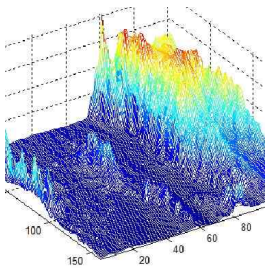
3

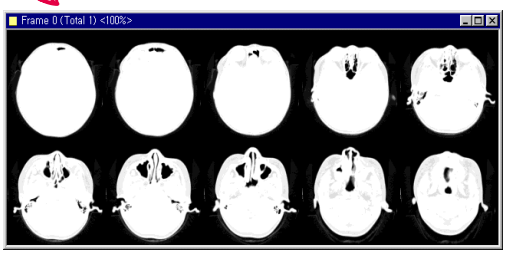
3차원 데이터의 다양한 유형

- 3차원 데이터 취득 형태
  - 비정렬 3차원 측정점
  - 등고선
  - DTED(고도 데이터)
  - 단층 영상 데이터



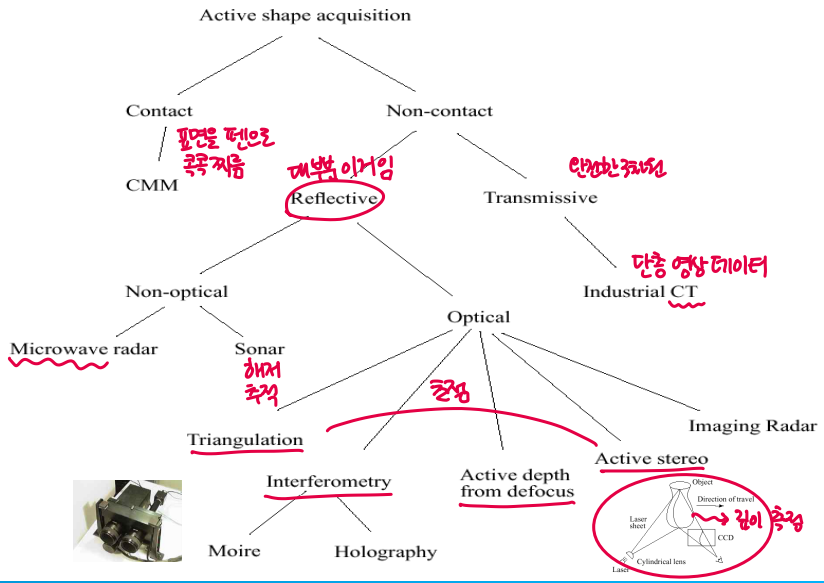






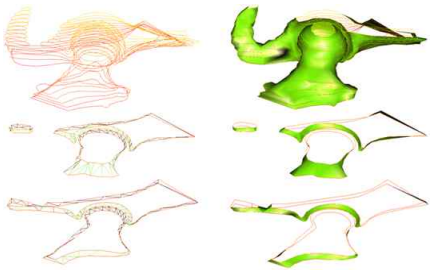
4

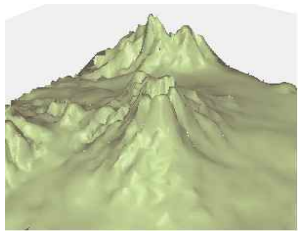
Taxonomy of 3D Scanning Methods

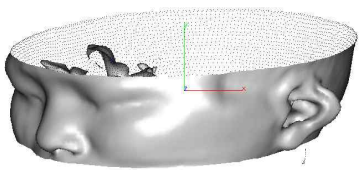


3차원 물체의 표면기반 표현

- 표면기반(surface-based) 표현



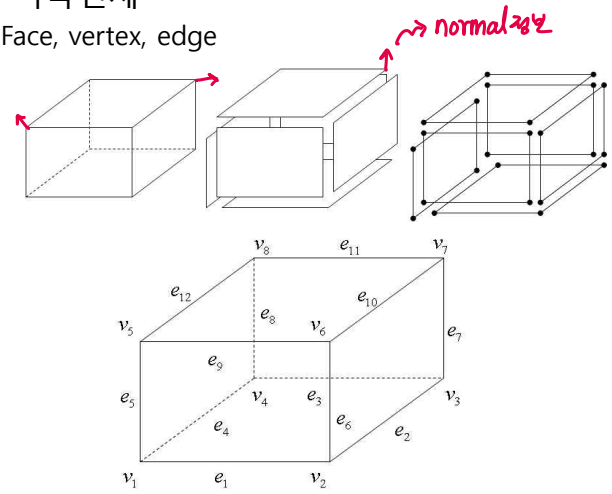




  - 경계면 표현
  - 메쉬(mesh): surface mesh, polygon mesh, surface polygon
  - 주로 삼각형 사용(Why?)
  - 다른 방법들: Quadratic surfaces, Superquadrics, Spline

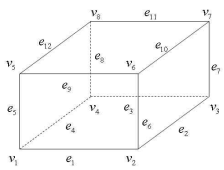
표면 모델의 관리 방법

- 예) 직육면체
  - Face, vertex, edge



(1) A vertex-based model

- 정점 정보 + 면 정보 (어려움, 비효율적)

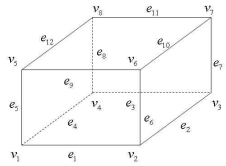


vertex	coordinates	face	vertices
v_1	$x_1\ y_1\ z_1$ – n	f_1	$v_1\ v_2\ v_3\ v_4$
v_2	$x_2\ y_2\ z_2$ – n	f_2	$v_6\ v_2\ v_1\ v_5$
v_3	$x_3\ y_3\ z_3$ ·	f_3	$v_7\ v_3\ v_2\ v_6$
v_4	$x_4\ y_4\ z_4$ ·	f_4	$v_8\ v_4\ v_3\ v_7$
v_5	$x_5\ y_5\ z_5$ ·	f_5	$v_5\ v_1\ v_4\ v_6$
v_6	$x_6\ y_6\ z_6$ ·	f_6	$v_8\ v_7\ v_6\ v_5$
v_7	$x_7\ y_7\ z_7$ ·		
v_8	$x_8\ y_8\ z_8$ ·		

[Lab 4-1] 정점기반 표현을 위한 자료구조를 구상하라. 3차원 공간상의 정점(Vertex)와 이를 이용한 메쉬(Mesh) 클래스를 설계하라.

(2) An edge-based model

- 에지 정보 위주 (정점+면에 에지 정보 추가)



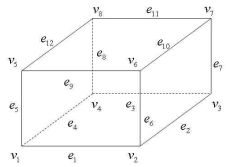
edge	vertices
e_1	v_1 v_2
e_2	v_2 v_3
e_3	v_3 v_4
e_4	v_4 v_1
e_5	v_1 v_5
e_6	v_2 v_6
e_7	v_3 v_7
e_8	v_4 v_8
e_9	v_5 v_6
e_{10}	v_6 v_7
e_{11}	v_7 v_8
e_{12}	v_8 v_5

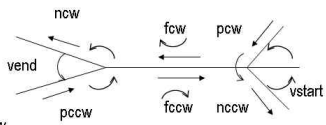
vertex	coordinates
v_1	x_1 y_1 z_1
v_2	x_2 y_2 z_2
v_3	x_3 y_3 z_3
v_4	x_4 y_4 z_4
v_5	x_5 y_5 z_5
v_6	x_6 y_6 z_6
v_7	x_7 y_7 z_7
v_8	x_8 y_8 z_8

face	edges
f_1	e_1 e_2 e_3 e_4
f_2	e_9 e_6 e_1 e_5
f_3	e_{10} e_7 e_2 e_6
f_4	e_{11} e_8 e_3 e_7
f_5	e_{12} e_5 e_4 e_8
f_6	e_{12} e_{11} e_{10} e_9

(3) Winged-edge data structure (Baumgart)

- 메쉬 편집용 → 더 복잡한 자료구조





edge	vstart	vend	fcw	fccw	ncw	pcw	nccw	pccw
e_1	v_1	v_2	f_1	f_2	e_2	e_4	e_5	e_6
e_2	v_2	v_3	f_1	f_3	e_3	e_1	e_6	e_7
e_3	v_3	v_4	f_1	f_4	e_4	e_2	e_7	e_8
e_4	v_4	v_1	f_1	f_5	e_1	e_3	e_8	e_5
e_5	v_1	v_5	f_2	f_5	e_9	e_1	e_4	e_{12}
e_6	v_2	v_6	f_3	f_2	e_{10}	e_2	e_1	e_9
e_7	v_3	v_7	f_4	f_3	e_{11}	e_3	e_2	e_{10}
e_8	v_4	v_8	f_5	f_4	e_{12}	e_4	e_3	e_{11}
e_9	v_5	v_6	f_2	f_6	e_6	e_5	e_{12}	e_{10}
e_{10}	v_6	v_7	f_3	f_6	e_7	e_6	e_9	e_{11}
e_{11}	v_7	v_8	f_4	f_6	e_8	e_7	e_{10}	e_{12}
e_{12}	v_8	v_5	f_5	f_6	e_5	e_8	e_{11}	e_9

vertex	first edge	coordinates
v_1	e_1	x_1 y_1 z_1
v_2	e_2	x_2 y_2 z_2
v_3	e_3	x_3 y_3 z_3
v_4	e_4	x_4 y_4 z_4
v_5	e_9	x_5 y_5 z_5
v_6	e_{10}	x_6 y_6 z_6
v_7	e_{11}	x_7 y_7 z_7
v_8	e_{12}	x_8 y_8 z_8

face	first edge
f_1	e_1
f_2	e_9
f_3	e_6
f_4	e_7
f_5	e_{12}
f_6	e_9

4.2 기본적인 수학

- 벡터 공간
- 어파인 공간
- 좌표계
- 동차좌표

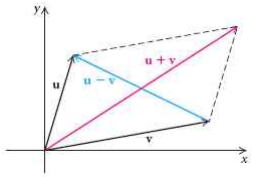
벡터 공간(Vector Space)

- 스칼라(scalar): 크기는 있고 방향은 없는 양
- 벡터(vector): 크기와 방향을 가짐
- 벡터 공간
  - 주어진 벡터로부터 파생되는 모든 벡터의 집합

- Scalar 연산
  - 덧셈, 뺄셈, 곱셈, 나눗셈, ..
  - 교환 법칙
  - 결합법칙
  - 역원, 항등원

벡터 연산

- 스칼라와 벡터의 곱셈(나눗셈)
- 벡터와 벡터의 덧셈(뺄셈)
- 역벡터

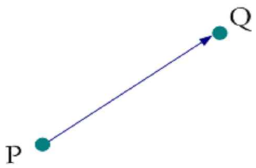


- 기하학에서는 점이 추가로 필요함
- 점: 위치만 있고 크기나 방향이 없음
  - 벡터 공간에서는 표현할 방법이 없음
  - → 공간을 확장 → 어파인 공간

13

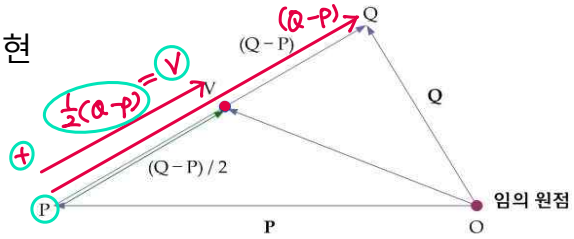
어파인 공간(Affine Space)

- 벡터 공간은 위치의 의미가 없음 → 기하 표현에 무리
- 점과 벡터를 동족처럼 취급함으로써 벡터공간을 확장
  - 벡터의 크기, 방향 + 위치
- 어파인 연산
  - 벡터와 벡터의 덧셈/뺄셈
  - 스칼라와 벡터의 곱셈/나눗셈
  - 점과 벡터의 덧셈/뺄셈 → 추가
  - 점과 점의 뺄셈 → 추가
- 점(P,Q,R)과 벡터(u,v,w)의 연산
  - 점 – 점 = 벡터 → v = P - Q
  - 점 + 벡터 = 점 → Q = P + v
  - 벡터+벡터 = 벡터 → w = u + v
  - 점+점 ? 자명X



14

- 선분 표현



  - V = P + (1/2)(Q - P)

일반화 ⇒ V = P + t (Q - P) = (1 - t)P + (t)Q (0 ≤ t ≤ 1)

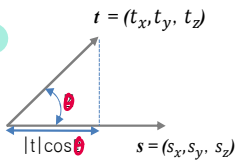
대표적

- Affine Sum 아핀 sum
  - 점의 계수 합이 1이 되는 경우
  - 점의 덧셈은 각 점들 앞의 계수 합이 1일 때에 한해서만 허용됨.
  - Ex) Parametric Equation

벡터의 내적과 외적

1) 벡터의 내적
   - 후면 제거나 조명과 음영 계산 등에서 다양하게 사용

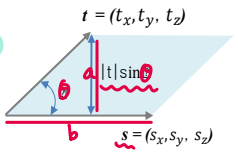
$$s \cdot t = |s||t|\cos\theta = s_x t_x + s_y t_y + s_z t_z$$



2) 벡터의 외적 = 새로운 벡터
   - 표면의 법선 등 다양하게 사용

$$s \times t = |s||t|\sin\theta\, n$$
$$= (s_y t_z - s_z t_y, -s_x t_z + s_z t_x, s_x t_y - s_y t_x)$$

$$\begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

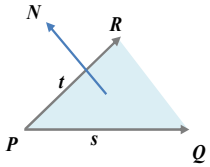


axb = (삼각형) X 2

(3) 평면 법선 벡터 계산 방법

- 3차원 점들의 순서가 중요
- OpenGL: 오른손 법칙

$$s = (Q_x - P_x, Q_y - P_y, Q_z - P_z)$$
$$t = (R_x - P_x, R_y - P_y, R_z - P_z)$$
$$N = s \times t$$



시험

[Quiz 4-1] 세 정점이 순서대로 주어졌을 때 이들에 의해 정의되는 삼각형의 표면 법선 벡터를 계산하라.

[Quiz 4-2] 세 정점이 순서대로 주어졌을 때 이들에 의해 정의되는 삼각형의 면적을 계산하라.

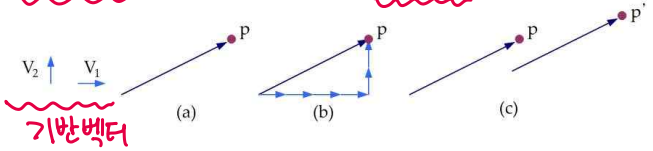
17

좌표계 (Coordinate System)

- 다양한 좌표계
  - 직교 좌표계 : Cartesian Coordinate System, Rectangular CS
  - 원기둥 좌표계: Cylindrical Coordinate System
  - 원구 좌표계: Spherical Coordinate System
  - 직교좌표계가 가장 많이 사용 됨
- 기반 벡터(basis vector)

  자신들의 합성으로 인해 모든 벡터를 표시할 수 있는 벡터
  - 어떤 벡터가 가능한가? 선형 독립(linear independent)
    - 벡터 v = 4 V1 + 2 V2 + V3:

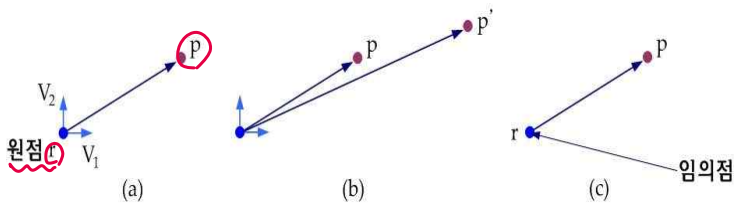
    ★ 서로 방향이 같으면 X
  - 3차원 공간에서는 일반적으로 x, y, z축으로 기반벡터



18

좌표계

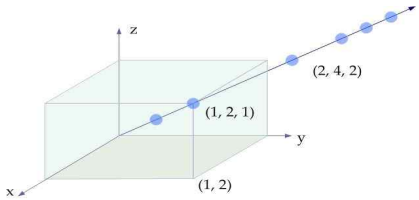
- 좌표계
  - 원점과 기반벡터로 구성되는 프레임
    - Ex. 3차원 좌표계 = (r, V1, V2, V3)
  - 원점: 어파인 공간에서 기반벡터 시작점을 일치시킨 곳
  - 점 p = r + 4V1 + 2V2 + V3: 원점이 필요



동차좌표(Homogeneous Coordinate System)

- 3차원 벡터와 점을 달리 표현할 수 있도록 함
  - v = 4 V1 + 2 V2 + V3
  - P = r + 4V1 + 2V2 + V3
- 차원을 하나 올리는 방법 (동차좌표)
  - v = 4 V1 + 2 V2 + V3 + 0 r = (4, 2, 1, 0): 벡터 0이면 벡터
  - P = 4V1 + 2V2 + V3 + 1 r = (4, 2, 1, 1): 점 0이 아니면 점
- 네 번째 항목이 0이면 벡터를 나타내고, 0이 아니면 점
- 동차좌표 (x, y, z, w) → 3차원 좌표 (x/w, y/w, z/w)

- 예) 3차원 점 (1, 2, 1)
  - 4차원 동차좌표로 사상 : (1,2,1, 1) = (2,4,2, 2) = (3,6,3, 3) = ...



- 점과 벡터의 표현 방법: 행렬
  ① 행 행렬
  ② 열 행렬

행 행렬: $[x, y, z, 1]$

열 행렬: $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

4.3 2차원 기하 변환

- 이동(translation)
- 회전(rotation)
- 신축(scaling)
- 복합 변환
- 동차좌표계를 이용한 변환
- 기타 변환: 반사, 밀림

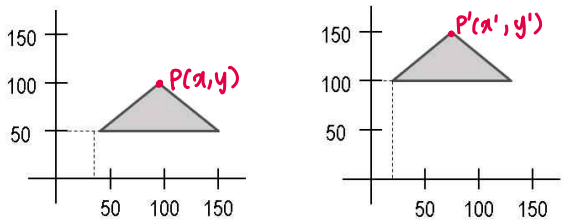
이동(translation)

- 형태불변(rigid body) 변환

Just 이동

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x+t_x \\ y+t_y \end{bmatrix}, P = \begin{bmatrix} x \\ y \end{bmatrix}, T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

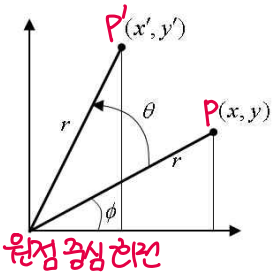
$$P' = P + T$$



회전(rotation)

- 형태불변(rigid body) 변환

Just 한번



원점 중심 회전

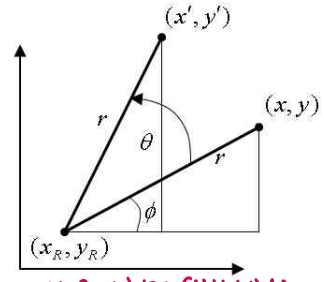
$$P = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos\phi \\ r\sin\phi \end{bmatrix}$$

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r\cos(\phi+\theta) \\ r\sin(\phi+\theta) \end{bmatrix} = \begin{bmatrix} r\cos\phi\cos\theta - r\sin\phi\sin\theta \\ r\cos\phi\sin\theta + r\sin\phi\cos\theta \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

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

회전 변환행렬 R, P: 열행렬

- 피벗을 중심으로 한 2D 회전변환



피벗 중심회전 (복합변환)

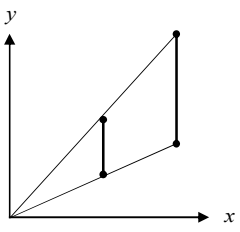
신축(scaling)

"형태 변함"

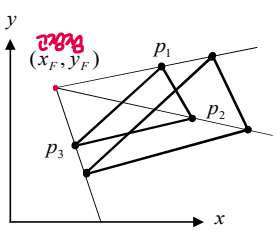
- 형태 불변 변환이 아님

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} xS_x \\ yS_y \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = SP$$

곱행렬

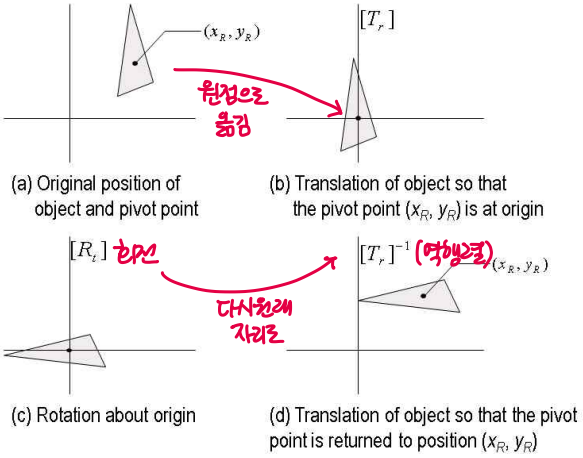


- 고정점을 중심으로 한 2D 신축변환

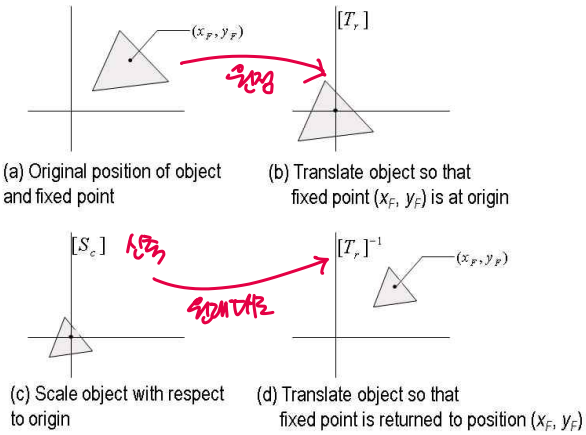


2차원 복합변환

- 피벗을 중심으로 한 회전



- 고정점을 중심으로 한 신축변환



동차좌표계를 이용한 변환

- 변환 형태가 다름
  - 회전, 신축: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
  - 이동: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$
- 복합 변환에서의 비효율

$$P' = M_n M_{n-1} \cdots M_2 M_1 P = MP$$

이때, $M = M_n M_{n-1} \cdots M_2 M_1$

- How??? ➔ 동차 좌표계에서의 변환으로 처리

- 동차 좌표계 사용
  - 2차원 점: (x, y, 1)
  - 변환 형태가 모두 동일

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

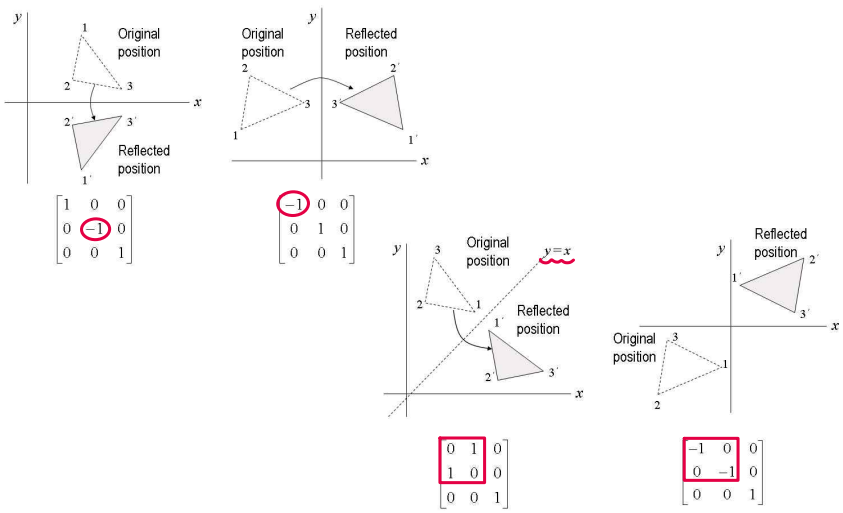
rotation
scaling
translation
perspective transform

- 이동: $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+t_x \\ y+t_y \\ 1 \end{bmatrix}$ (P′ = T P)
- 회전: $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- 신축: $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

- 동차 좌표계 사용 장점
  - 복합변환에서 각각의 변환을 순서적으로 적용할 필요가 없음
  - 복합변환 전체를 하나의 변환행렬로 미리 계산해 놓음
  - 이를 한꺼번에 적용 → 처리시간 크게 개선

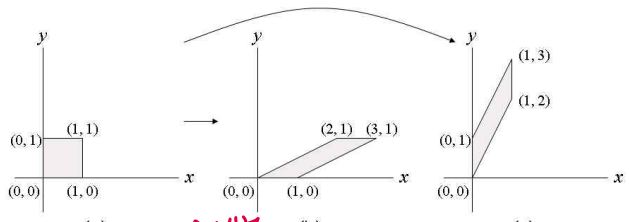
- 개별적 복합 변환: $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = [T_n] \cdots [T_3][T_2][T_1] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- 통합된 복합 변환: $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = [T] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad [T] = [T_n] \cdots [T_3][T_2][T_1]$

기타 변환: 반사 (reflection)



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

기타 변환: 밀림 (shearing)



$SH_x = 2,\ x = 1$

$y' = SH_x x + y$

$x + \frac{y}{SH_x} = \frac{y'}{SH_x}$

$x = \frac{y' - y}{SH_x}$

$1 = \frac{y' - y}{2}$

$\therefore y' = y + 2$

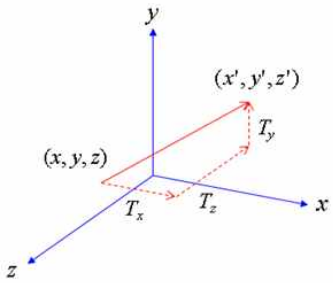
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ SH_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (c)$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & SH_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (b)$$

4.4 3차원 기하 변환

- 이동(translation)
- 회전(rotation)
- 신축(scaling)
- 반사
- 밀림
- 기타변환
- 복합 변환

이동(translation)

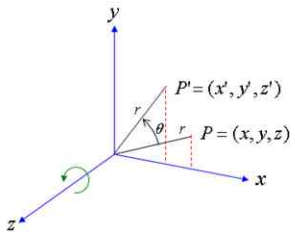


$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = T \cdot P$$

회전(rotation)

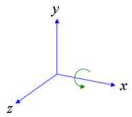
- Rotation about z-axis



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

$$P' = R_z \cdot P$$

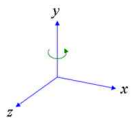
- x-axis



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

$$P' = R_x \cdot P$$

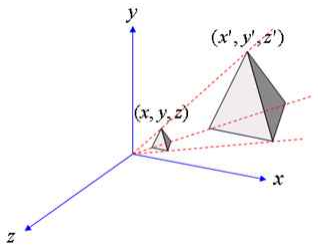
- y-axis



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

$$P' = R_y \cdot P$$

신축(scaling)



$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = S \cdot P$$

37

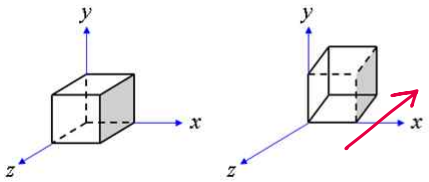
반사 (reflection)

- xy plane : (x, y, z) → (x, y, -z)
  - right-hand coord. <=> left-hand coord.
- yz plane : (x, y, z) → (-x, y, z)
- zx plane : (x, y, z) → (x, -y, z)

- arbitrary plane에 의한 반사는 어떻게 할까?

38

밀림 (shearing)



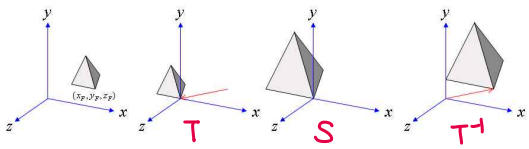
$$P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P$$

조만일명

$$P' = Sh \cdot P$$

3차원 복합변환

- Scaling with respect to a selected fixed point

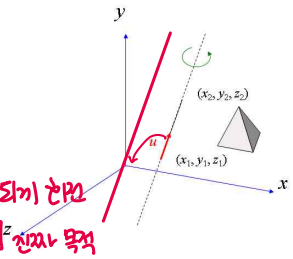


$$P' = [T^{-1}][S][T]P, \text{ where}$$

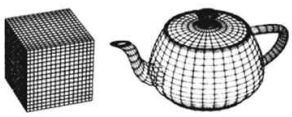
$$T = \begin{bmatrix} 1 & 0 & 0 & x_F \\ 0 & 1 & 0 & y_F \\ 0 & 0 & 1 & z_F \\ 0 & 0 & 0 & 1 \end{bmatrix}, S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{and } T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -x_F \\ 0 & 1 & 0 & -y_F \\ 0 & 0 & 1 & -z_F \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

오류 오류

- Rotation about an arbitrary axis (임의의 축)
  - 1. Translate the object so that the rotation axis passes through the origin. (원점 지나도록 변환 [T])
  - 2. Rotate the object so that the rotation axis coincides with one of the coordinate axis. (축자체가 y축과 수평되게 한다)
  - 3. Perform the specified rotation. $[R]_z$ (진짜 목적)
  - 4. R 역변환
  - 5. T 역변환



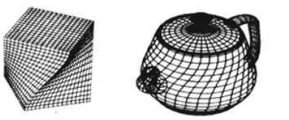
구조왜곡 변환(Structure-Deforming Transformation)



테이퍼링(Tapering): z에 따라 x, y의 크기조절
휨(Bending): 축을 따라 물체가 휨
비틀림(Twisting): z에따라 회전각↑증가

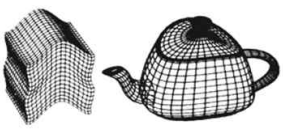
- Tapering

$$x' = r(z) \cdot x$$
$$y' = r(z) \cdot y$$
$$z' = z$$



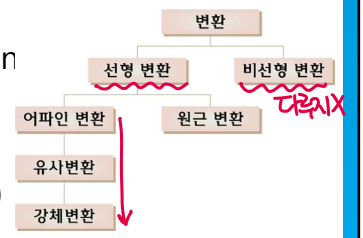
- Twisting

$$x' = x\cos\theta(z) - y\sin\theta(z)$$
$$y' = x\sin\theta(z) - y\cos\theta(z)$$
$$z' = z$$

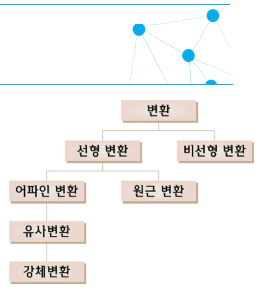


- Bending

그래픽 변환 분류



- 강체변환(Rigid Body Transformation)
  - 이동변환, 회전변환
  - 물체 자체의 모습은 불변
- 유사변환(Similarity Transformation)
  - 강체변환 + 균등 크기조절 변환, 반사변환
  - 물체면 사이의 각이 유지됨.
  - 물체내부 정점간의 거리가 일정한 비율로 유지됨
- 어파인변환(Affine Transformation)
  - 유사변환 + 차등 크기조절 변환, 전단변환
  - 물체의 타입이 유지
    - 직선은 직선으로, 다각형은 다각형으로, 곡면은 곡면으로
    - 평행선이 보존
    - 변환행렬의 마지막 행이 항상 (0, 0, 0, 1)



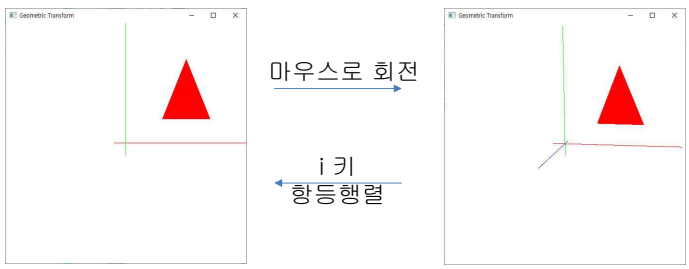
- 원근변환(Perspective Transformation)
  - 평행선이 만남.
  - 직선이 직선으로 유지
  - 변환행렬의 마지막 행이 (0, 0, 0, 1) 아님.
- 선형변환(Linear Transformation)
  - 어파인 변환 + 원근 변환
  - 선형 조합(Linear Combination)으로 표시되는 변환
  - x' = ax + by + cz에서 x'는 x, y, z 라는 변수를 각각 상수 배 한 것을 더한 것이다.
  - 예: $x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta = x\cos\theta - y\sin\theta$

    $x' = x + Shy \cdot y$

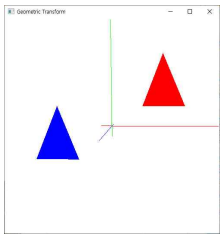
행렬과 변환

4.5 기하 변환의 구현

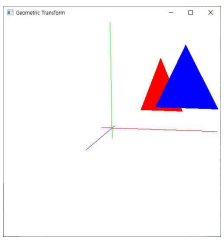
- 자신만의 기하변환 라이브러리 만들기
  - 다양한 변환 테스트
  - 행렬 적용 이해
  - GLUT 활용
- 구현 예



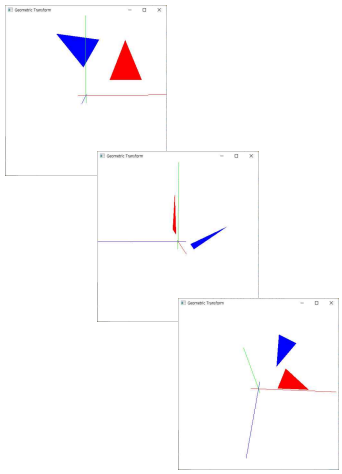
- 이동(t)



- 신축(s)

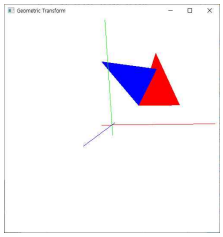


- 회전(z, x, y)

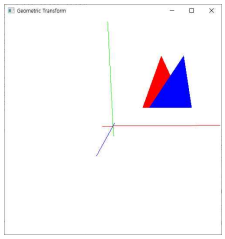


47

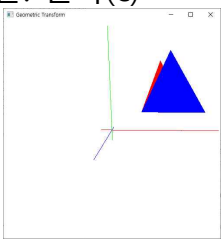
- 복합변환: 회전(Z)

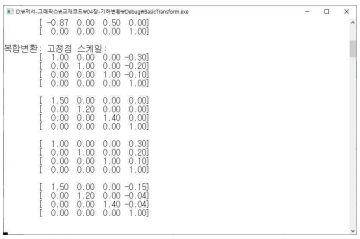


- 밀림(h)



- 복합변환: 신축(c)





```
[ -0.87  0.00  0.50  0.00]
[  0.00  0.00  0.00  1.00]

복합변환: 고정점 스케일:
[  1.00  0.00  0.00 -0.30]
[  0.00  1.00  0.00 -0.20]
[  0.00  0.00  1.00 -0.10]
[  0.00  0.00  0.00  1.00]

[  1.50  0.00  0.00  0.00]
[  0.00  1.20  0.00  0.00]
[  0.00  0.00  1.40  0.00]
[  0.00  0.00  0.00  1.00]

[  1.00  0.00  0.00  0.30]
[  0.00  1.00  0.00  0.20]
[  0.00  0.00  1.00  0.10]
[  0.00  0.00  0.00  1.00]

[  1.50  0.00  0.00 -0.15]
[  0.00  1.20  0.00 -0.04]
[  0.00  0.00  1.40 -0.04]
[  0.00  0.00  0.00  1.00]
```

48

자신만의 기하변환 라이브러리 만들기

- OpenGL에서는 기하변환을 위한 여러 가지 함수 제공
- 직접 만들어 보자.
  - 변환행렬의 생성: glkMatSet()
  - 변환행렬을 곱하기: glkMatMult(double* m1, double m2);
  - 행렬을 화면에 출력: glkMatPrint(double *m)
  - 변환행렬 생성
    - 항등행렬: glkMatIdentity(double* m)
    - 이동: glkMatTrans(double* m, double tx, double ty, double tz)
    - 신축: glkMatScale(double* m, double sx, double sy, double sz)
    - 회전(X,Y,Z축 중심): glkMatRotateZ(double* m, double a)
    - 밀림(X,Y,Z축 방향): glkMatShearX(double* m, double dy, double dz)
  - 정점 변환처리: glkTransform(double* m, double* p, double* q)
  - 기타 그리기 함수들: 선분 그리기, 삼각형 그리기, 좌표축 그리기

49

감사합니다!

50