

6장. 모델변환과 시점변환

👤 학습목표

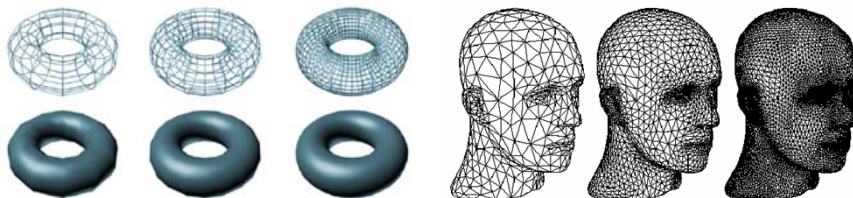
- 경계면 표현(B-Rep) / 물체 내부표현
- 어파인 공간을 정의하는 이유를 이해한다.
- 동차좌표를 정의하는 이유를 이해한다.
- 이동, 회전, 크기조절 등의 기하변환과 변환행렬을 이해한다.
- 모델 좌표계, 전역 좌표계, 시점 좌표계의 차이점을 이해한다.
- 좌표계 변환과 변환 행렬과의 관계를 이해한다.
- 기하변환 순서와 함수호출 순서의 상관관계를 이해한다.
- Point3D 클래스를 이해하고 활용한다.
- Translate/Rotate/Scale 메뉴를 프로그램에 넣고 구현한다.
- 자신만의 Robot을 만들어 본다.
- 다양한 방법으로 시점을 변환해 본다.

1

6.1 좌표계-3차원 물체표현

👤 경계면 표현(Boundary Surface Representation)

- 메쉬(Mesh), 표면 메쉬(Surface Mesh), 다각형 메쉬(Polygon Mesh), 표면 다각형(Surface Polygon), 다각형(Polygon)
- 사각형 메쉬(Rectangular Mesh): 평면 보장 못함.
- 삼각형 메쉬(Triangular Mesh): 평면 보장. 2배의 드로잉 속도



[그림 6-2] 120, 300, 1000 개의 다각형 표면

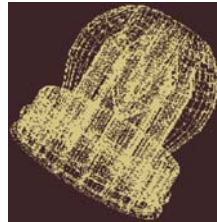
[그림 6-5] 삼각형 메쉬

2

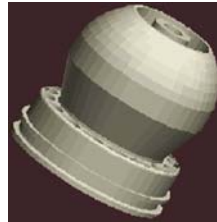
와이어 프레임과 솔리드 렌더링

Wireframe Rendering / Solid Rendering

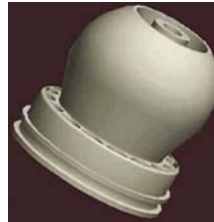
- 와이어 프레임: 드로잉 속도가 빠름
- 가끔씩 솔리드 렌더링으로 외형 확인
 - `glutWireTeapot()`
 - `glutSolidTeapot()`



[그림 6-6] 설계 I



[그림 6-7] 설계 II



[그림 6-8] 설계 III

3

Vector공간 / Affine 공간

Vector: 크기와 방향

법칙들 : pp.240



[그림 6-11] 벡터 II

$V = Q - P$ // 점 - 점

$Q = V + P$ // 벡터 + 점

어파인 공간(Affine Space)

- 점과 벡터를 동족처럼 취급함으로써 벡터공간을 확장

어파인 연산

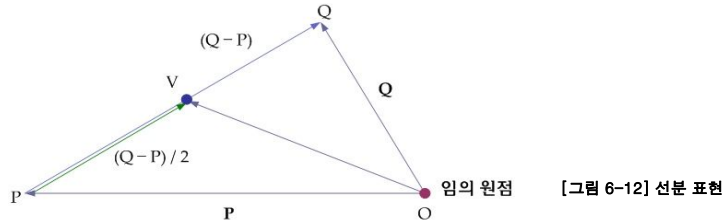
- 벡터와 벡터의 덧셈(뺄셈)
- 스칼라와 벡터의 곱셈(나눗셈)
- 점과 벡터의 덧셈(뺄셈)

- Scalar / Vector ?
- Vector space ?
- Inner Product ?
- Cross Product ?

4

Affine 공간

선분표현



👤 $V = P + (1/2)(Q - P)$

👤 $V = P + t(Q - P) = (1 - t)P + (t)Q \quad (0 \leq t \leq 1)$

Affine Sum

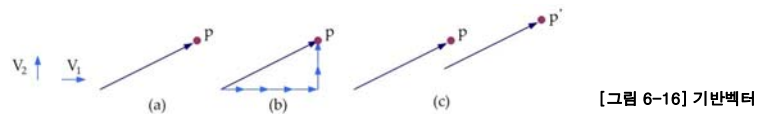
- 점의 계수 합이 1이 되는 경우
- 점의 덧셈은 각 점들 앞의 계수 합이 1일 때에 한해서만 허용됨.
- Ex) Parametric Equation

5

좌표축과 좌표계

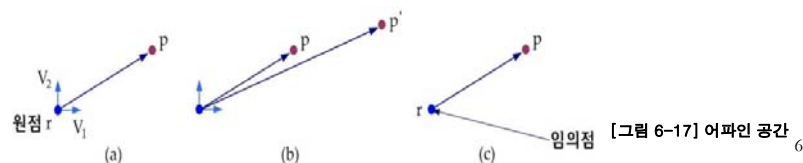
기반벡터 (Basis Vector)

- 벡터 $p =$ 벡터 p'
- 벡터 $v = 4V_1 + 2V_2 + V_3$:
 - 방향동일, 기반벡터만으로 표시가능
 - → 어떤 벡터가 가능한가? Linear independence



좌표계

- 원점과 기반벡터로 구성되는 프레임
 - Ex. 3차원 좌표계 = (r, V_1, V_2, V_3)
- 원점: 어파인 공간에서 기반벡터 시작점을 일치시킨 곳
- 점 $p = r + 4V_1 + 2V_2 + V_3$: 원점이 필요



동차좌표(Homogeneous Coordinate System)

👤 벡터와 점의 표현이 다름

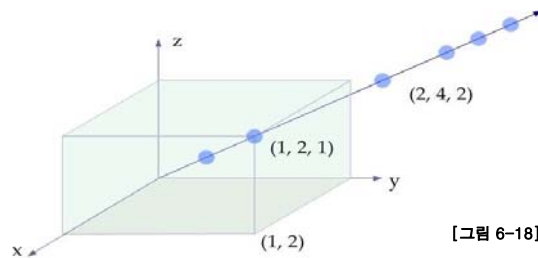
- $v = 4V_1 + 2V_2 + V_3$
- $P = r + 4V_1 + 2V_2 + V_3$

👤 차원을 하나 올리면 동일 방법으로 표현

- $v = 4V_1 + 2V_2 + V_3 + 0r = (4, 2, 1, 0)$: 벡터
- $P = 4V_1 + 2V_2 + V_3 + 1r = (4, 2, 1, 1)$: 점

👤 3차원 점 (1, 2, 1)

- 4차원 동차좌표로 사상
- 동차좌표 $(1, 2, 1, 1) = (2, 4, 2, 2) = (3, 6, 3, 3) = \dots$
- 동차좌표 $(x, y, z, w) \Rightarrow$ 3차원 좌표 $(x/w, y/w, z/w)$



→ 동차좌표가 왜 필요한가?

[그림 6-18] 동차좌표

7

6.2기하변환- Geometric Transformation

👤 물체 변환 또는 좌표계 변환의 기본

👤 행렬로 표현됨

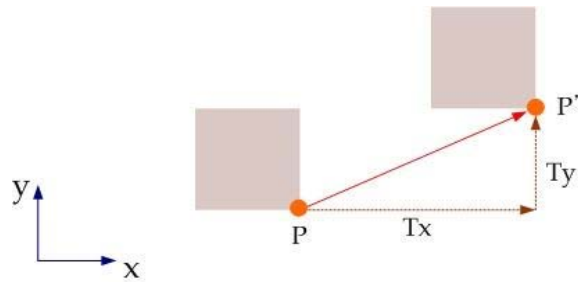
👤 이동, 회전, 크기조절 등

8

2차원 이동(Translation)

$$\begin{matrix} \text{P} \end{matrix} \mathbf{x}' = 1 \times \mathbf{x} + 0 \times \mathbf{y} + T_x \times 1$$

$$\begin{matrix} \text{P} \end{matrix} \mathbf{y}' = 0 \times \mathbf{x} + 1 \times \mathbf{y} + T_y \times 1$$



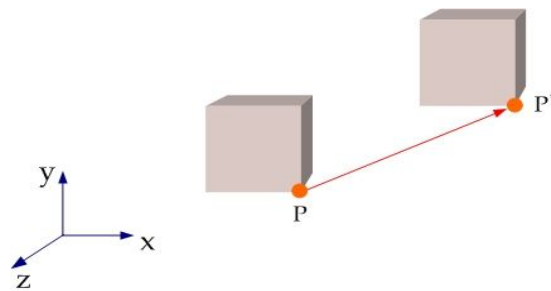
[그림 6-19] 2차원 이동

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (6.12)$$

→ 동차좌표를 안쓰면?

9

3차원 이동



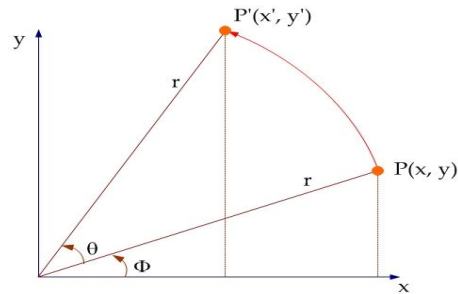
[그림 6-19] 2차원 이동

$$P' = T \cdot P \quad (6.13)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (6.14)$$

10

2차원 회전(Rotation)



[그림 6-21] 2차원 회전

$$x' = r \cos(\phi + \theta) = r \cos \phi \cos \theta - r \sin \phi \sin \theta = x \cos \theta - y \sin \theta \quad (6.15)$$

$$y' = r \sin(\phi + \theta) = r \cos \phi \sin \theta + r \sin \phi \cos \theta = x \sin \theta + y \cos \theta \quad (6.16)$$

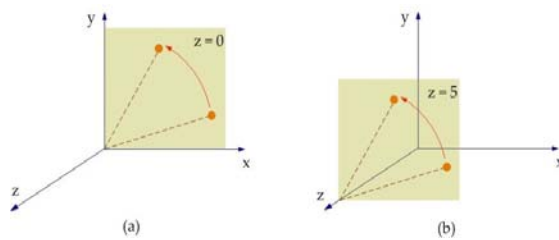
$$P' = R \cdot P \quad (6.17)$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (6.18)$$

11

3차원 회전

- 회전축 기준의 회전에 정의
- 반시계 방향의 회전각



[그림 6-22] 3차원 회전



[그림 6-24] 반시계 방향

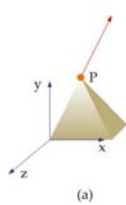
$$P' = R_z(\theta) \cdot P \quad (6.19)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (6.20)$$

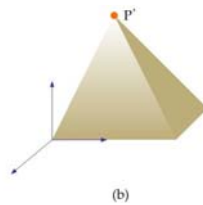
12

크기조절(Scaling)

☞ 균등 크기조절(Uniform Scaling) vs. 차등 크기조절(Non-Uniform Scaling)

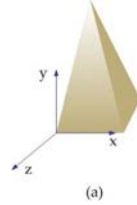


(a)

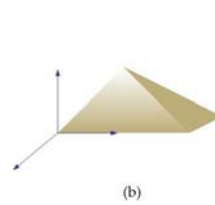


(b)

[그림 6-25] 균등 크기조절



(a)



(b)

[그림 6-26] 차등 크기조절

$$P' = S \cdot P$$

(6.25)

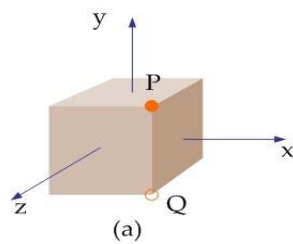
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

(6.26)

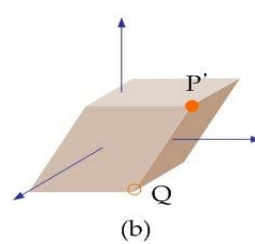
13

전단(Shearing)

☞ 예: x-y 평면에서의 전단



(a)



(b)

[그림 6-27] 전단

$$x' = x + Sh_y \cdot y$$

(6.27)

$$y' = y + Sh_x \cdot x$$

(6.28)

$$P' = Sh \cdot P$$

(6.29)

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & Sh_y & 1 & 0 \\ Sh_x & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

(6.30)

14

복합변환(Composite Transformation)

👤 크기조절(S1) 후, 결과 물체를 회전(R1)한 후,
다시 크기조절(S2)

- $P' = S2 \cdot R1 \cdot S1 \cdot P$
- 행렬곱셈의 순서에 유의
- $P' = C \cdot P$ 복합행렬 C는 한번만 계산. 모든 정점에 적용

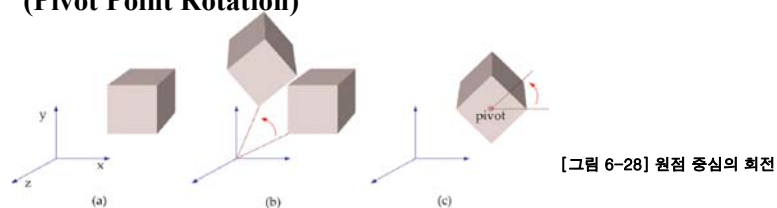
➔ 임의의 점을 지나가는 어떤 벡터를 중심으로 theta만큼 회전?

➔ 임의의 점을 중심으로 어떤 벡터 방향으로 scale만큼 확대?

15

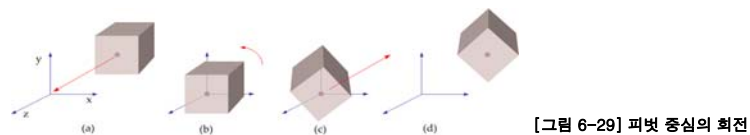
복합변환

👤 원점 기준 회전(Origin Rotation) versus 중심점 기준 회전(Pivot Point Rotation)



👤 중심점 기준 회전

- 피벗이 좌표계 원점에 일치하도록 물체를 이동한다.
- 물체를 원점 기준으로 축 주위로 회전한다.
- 회전된 물체를 ①번에서 이동한 방향의 반대 방향으로 이동한다.



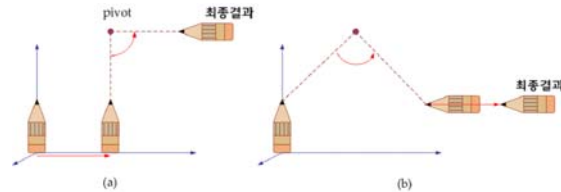
$$C = T(X_p, Y_p, Z_p) \cdot R_z(\theta) \cdot T(-X_p, -Y_p, -Z_p) \quad (6.31)$$

16

이동 후 회전과 회전 후 이동

교환법칙이 성립하지 않음.

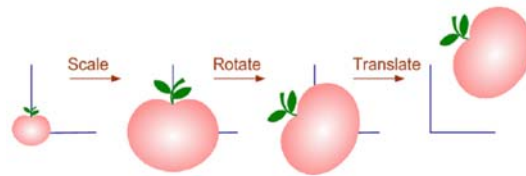
• $R \cdot T$ 와 $T \cdot R$ 은 일반적으로 서로 다른 결과



[그림 6-30] 이동 후 회전과 회전 후 이동 비교

물체 인스턴스

• $C = T \cdot R \cdot S$

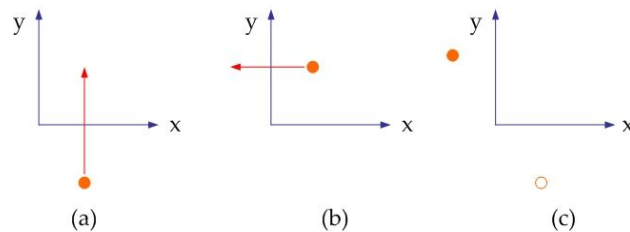


[그림 6-31] 크기조절/회전 및 이동

→ `glutWireCube()`를 이용해 테스트?

17

반사(Reflection)



[그림 6-32] 반사변환

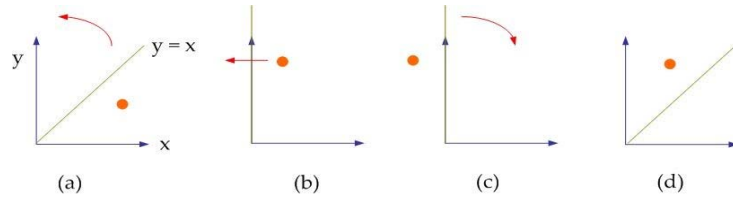
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$C = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.34)$$

18

y=x 기준의 반사

복합변환



[그림 6-33] y = x 축 기준의 반사 변환

$$\begin{pmatrix} \cos(-45^\circ) & -\sin(-45^\circ) & 0 \\ \sin(-45^\circ) & \cos(-45^\circ) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} \cos(45^\circ) & -\sin(45^\circ) & 0 \\ \sin(45^\circ) & \cos(45^\circ) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

19

구조왜곡 변환(Structure-Deforming Transformation)

테이퍼링(Tapering)

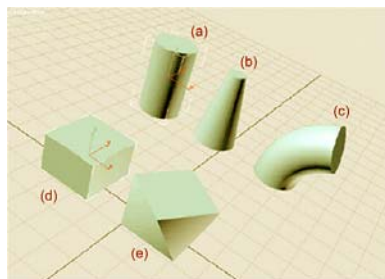
- z에 따라 x, y의 크기조절

휨(Bending)

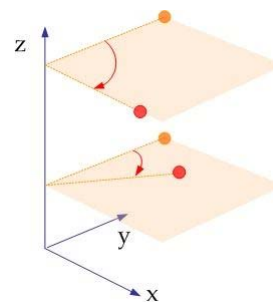
- 축을 따라 물체가 휨

비틀림(Twisting)

- z에 따라 회전각 증가



[그림 6-35] 구조왜곡 변환



[그림 6-36] 비틀림

20

그래픽 변환

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 $(0, 0, 1)$



[그림 6-38] 그래픽 변환 분류

21

그래픽 변환

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
- $x' = ax + by + cz$ 에서 x' 는 x, y, z 라는 변수를 각각 상수 배 한 것을 더한 것이다.
- 예: $x' = r \cos(\phi + \theta) = r \cos \phi \cos \theta - r \sin \phi \sin \theta = x \cos \theta - y \sin \theta$

$$x' = x + Shy \cdot y$$



[그림 6-38] 그래픽 변환 분류

22

행렬과 변환



[그림6-39] 행렬과 변환

23

6.3 지엘의 모델변환- 모델 좌표계

모델링

- 물체를 설계 = 물체 정점을 정의

좌표계 (눈금)단위

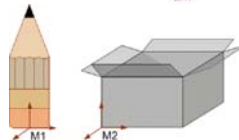
- 임의로 설정
- 물체공간(Object Space): 부동소수 정밀도



[그림 6-40] 모델링 기본단위

좌표계 원점 및 축방향

- 설계상의 편의
- 물체마다 서로 다름



[그림 6-41] 모델 좌표계

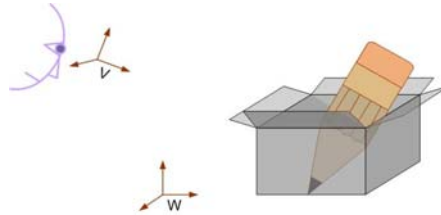
모델 좌표계(MCS, Modeling Coordinate System) 또는 지역 좌표계 (LCS, Local Coordinate System)

24

전역 좌표계, 시점 좌표계

👤 장면

- 여러 물체가 존재 = 여러 지역 좌표계가 존재
- 일률적으로 어우를 수 있는 기준 좌표계
 - 전역 좌표계(WCS, **World Coordinate System**)
- 임의 위치에 선정



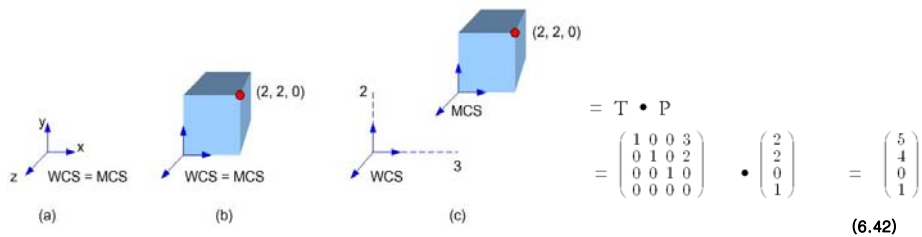
[그림 6-42] 전역 좌표계

👤 시점

- 바라보는 위치에 따라 장면은 달라보임
- 시점 좌표계(VCS, **View Coordinate System**)

25

변환행렬의 의미



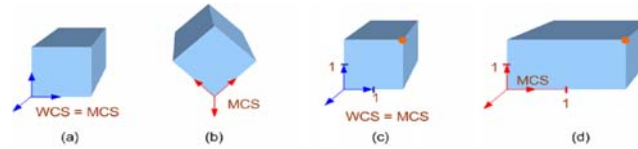
[그림 6-43] 모델 좌표계의 분리

👤 Ex. 이동변환

- 기본적으로 WCS=MCS
- 일반적 관점
 - 변환행렬 T는 WCS 기준으로 물체 정점을 (3, 2, 0)만큼 이동함을 의미.
- 지엘의 관점
 - 변환과 동시에 WCS와 MCS가 분리됨
 - 변환 후에도 **MCS 기준의 정점 좌표는 불변**
 - 좌표계의 이동으로 간주.
 - 전역 좌표계를 (3, 2, 0)만큼 이동하면 모델 좌표계와 일치.
 - **"전역 좌표계를 모델 좌표계로 일치시키기 위한 것이 변환행렬이다"**

26

변환행렬의 의미



[그림 6-45] 전역 좌표계와 모델 좌표계의 일치

회전

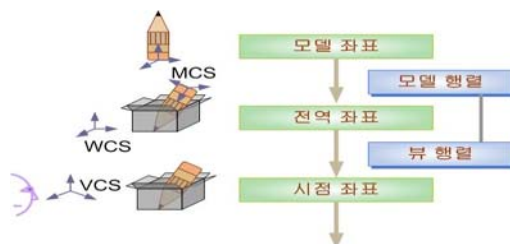
- 물체와 함께 MCS도 회전.
- MCS 기준의 물체 좌표는 불변
- 회전변환 행렬 T 를 (a)의 WCS를 45(b)의 MCS로 일치시키는데 사용. 이후 이를 MCS 기준으로 물체를 렌더링

크기조절: x축으로 2배

- MCS x축 눈금의 절대 길이가 바뀜.
- MCS 기준의 물체 좌표는 불변(Ex. (2, 2, 0))

27

지엘 파이프라인



[그림 6-46] 지엘 파이프라인

모델변환

- 물체에 가해지는 기하변환(이동, 회전, ...)
- 모델행렬로 대변됨
- 모델 좌표에 모델 행렬을 곱하면 전역좌표

시점변환 또는 뷰변환

- 카메라 위치와 방향 설정
- 뷰행렬로 대변됨
- 전역좌표에 뷰행렬을 곱하면 시점좌표

지엘은 모델행렬과 뷰행렬을 모델뷰 행렬 하나로 취급

- 물체를 뒤로 빼나 카메라를 앞으로 미나 마찬가지로

28

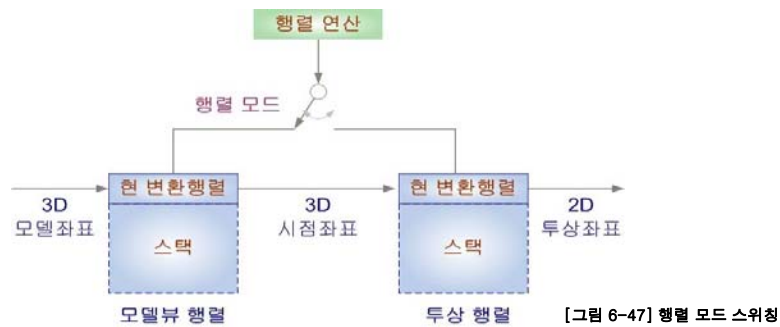
지엘의 모델변환

행렬 모드 설정

- 조작하고자하는 행렬을 선택
- `void glMatrixMode(GLenum mode);`
 - `GL_MODELVIEW`, `GL_PROJECTION`, `GL_TEXTURE`

현 변환행렬(CTM: Current Transformation Matrix)

- 상태변수. 스택 탑에 존재
- 항상 이것이 정점에 곱해짐



29

지엘의 모델변환

초기화

- `void glLoadIdentity();`
- 항등행렬로 초기화
- 초기화 결과

$$(6.43) \quad CTM = I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(6.44)

$$\text{모델 좌표계} = \text{전역 좌표계} = \text{시점 좌표계} \quad P' = CTM \cdot P = I \cdot P = P$$

기하변환을 명시

- `void glTranslatef(GLfloat dx, GLfloat dy, GLfloat dz);`
- `void glScalef(GLfloat sx, GLfloat sy, GLfloat sz);`
- `void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);`

후위곱셈(Post Multiplication)

- 전역 좌표계를 기준으로 하는 모델 좌표계의 변환
- $CTM = CTM \cdot M$
- Ex. `glTranslatef(1, 2, 0)`

$$\begin{aligned} CTM &= CTM \cdot T \\ &= I \cdot T \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (6.47) \end{aligned}$$

30

복합변환

Ex.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glScalef(sx, sy, sz);
glRotatef(theta, vx, vy, vz);
glBegin(GL_POINTS);
    glVertex3f(px, py, pz);
glEnd();
```

함수 종류	지엘 함수	행렬 연산
행렬 초기화	<code>glLoadIdentity()</code>	$CTM = I$
크기 조절	<code>glScalef()</code>	$CTM = CTM \cdot S = I \cdot S$
회전	<code>glRotatef()</code>	$CTM = CTM \cdot R = I \cdot S \cdot R$
정점 선언	<code>glVertex3f()</code>	$P' = CTM \cdot P = I \cdot S \cdot R \cdot P$

[표 6-1] 행렬 조작 함수

크기조절 이후 회전? 회전 이후 크기조절?

31

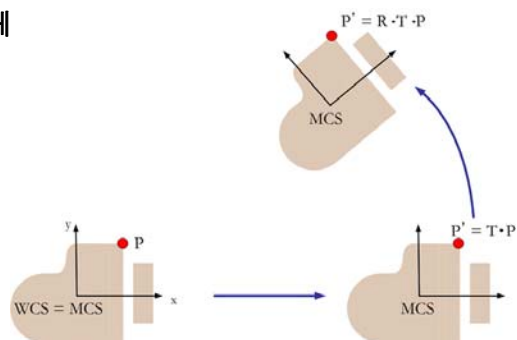
물체 변환에 의한 모델링

코드

- `glMatrixMode(GL_MODELVIEW);`
- `glLoadIdentity();`
- `glRotatef(45, 0.0, 0.0, 1.0);` 물체 변환의 역순
- `glTranslatef(10.0, 0.0, 0.0);`
- `glVertex3f(Px, Py, Pz);`

전역좌표 기준의 물체

- $P' = T \cdot P$
- $P'' = R \cdot P' = R \cdot T \cdot P$



[그림 6-48] 물체 변환에 의한 모델링

32

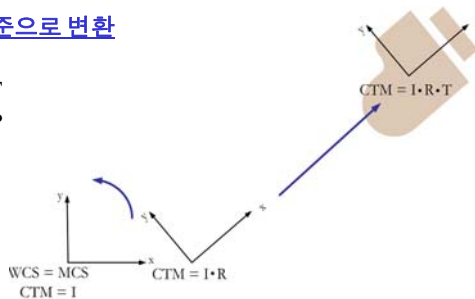
좌표계 변환에 의한 모델링

코드

- `glMatrixMode(GL_MODELVIEW);`
- `glLoadIdentity();`
- `glRotatef(45, 0.0, 0.0, 1.0);` 좌표계 변환과 동일 순서
- `glTranslatef(10.0, 0.0, 0.0);`
- `glVertex3f(Px, Py, Pz);`

모델 좌표계를 변환

- 직전의 모델 좌표계를 기준으로 변환
- $CTM = I \cdot R$
- $CTM = CTM \cdot T = I \cdot R \cdot T$
- $P'' = CTM \cdot P = I \cdot R \cdot T \cdot P$



[그림 6-49] 모델 좌표계 변환에 의한 모델링

33

함수호출 순서



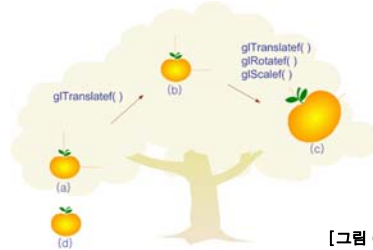
[그림 6-50] 행렬연산 순서



[그림 6-51] 함수호출 순서

34

행렬 스택



[그림 6-57] 오렌지 매달기

호출 함수	현 변환행렬	작업
<code>glMatrixMode(GL_MODELVIEW);</code>	$CTM \leftarrow ModelView \ CTM$	모델뷰 행렬 선택
① <code>glLoadIdentity();</code>	$CTM = I$	초기화
② <code>Draw_Orange();</code>	$P' = CTM \cdot P$	(a)의 오렌지 그리기
③ <code>glTranslatef(4.0, 4.0, 0.0);</code>	$CTM = CTM \cdot T1$	좌표계 이동
④ <code>Draw_Orange();</code>	$P' = CTM \cdot P$	(b)의 오렌지 그리기
⑤ <code>glTranslatef(6.0, -2.0, 0.0);</code>	$CTM = CTM \cdot T2$	좌표계 이동
⑥ <code>glRotatef(45, 0.0, 0.0, 1.0);</code>	$CTM = CTM \cdot R$	좌표계 회전
⑦ <code>glScalef(2.0, 2.0, 2.0);</code>	$CTM = CTM \cdot S$	좌표계 눈금 크기조절
⑧ <code>Draw_Orange();</code>	$P' = CTM \cdot P$	(c)의 오렌지 그리기

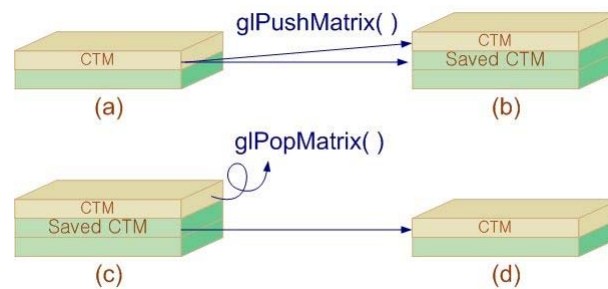
[표 6-2] 오렌지 매달기 프로그램

좌표계를 (d)로 되돌리려면 어떻게 하는가 -> **스택**

35

행렬 스택

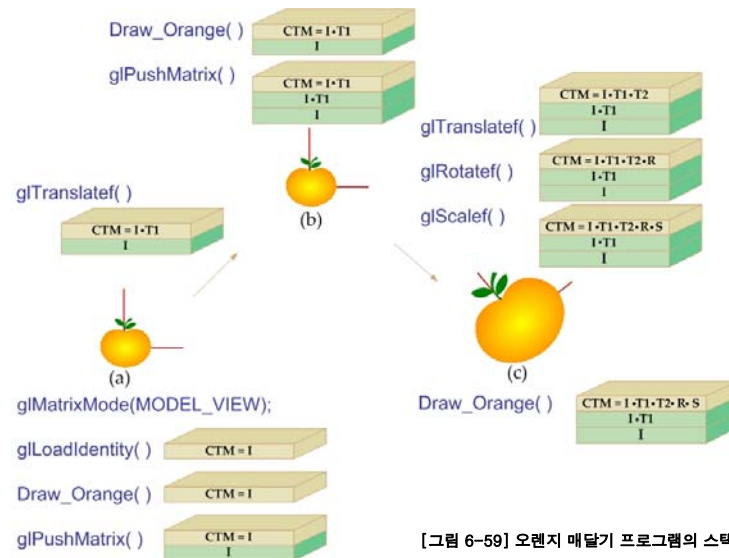
푸시, 팝



[그림 6-58] 지열의 푸시, 팝

36

행렬 스택



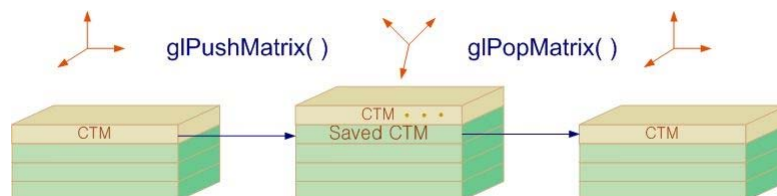
[그림 6-59] 오렌지 매달기 프로그램의 스택 변화

37

행렬 스택

일반적 형태

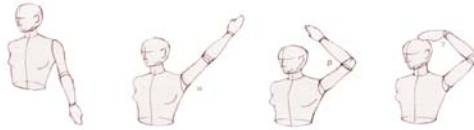
- `glPushMatrix();`
- `glTranslatef();`
- `glRotatef();`
- `glScalef();`
- ...
- `Draw_TransformedObject();`
- `glPopMatrix();`



[그림 6-60] 일반적인 스택 함수 호출

38

계층구조 모델링



[그림 6-61] 계층 [그림 6-62] [그림 6-63] II [그림 6-64] III

```
void drawArm( ){
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    Draw_Body( );
    glPushMatrix( );
        GoToShoulderCoordinates( );
        Draw_UpperArm( );
        glPushMatrix( );
            GoToElbowCoordinates( );
            Draw_LowerArm( );
            glPushMatrix( );
                GoToWristCoordinates( );
                Draw_Hand( );
            glPopMatrix( );
        glPopMatrix( );
    glPopMatrix( );
}
```

전역 좌표계 = 모델 좌표계
 몸체 그리기
 전역 좌표계 저장
 어깨 기준 모델 좌표계
 위 팔 그리기
 어깨 기준 모델 좌표계 저장
 팔꿈치 기준 모델 좌표계
 아래팔 그리기
 팔꿈치 기준 모델 좌표계 저장
 손목 기준 모델 좌표계
 손 그리기
 팔꿈치 좌표계 복원
 어깨 좌표계 복원
 몸체 좌표계 복원

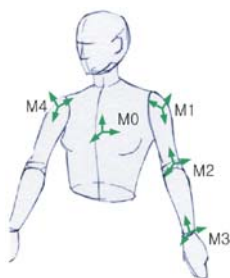
39

계층구조 모델링

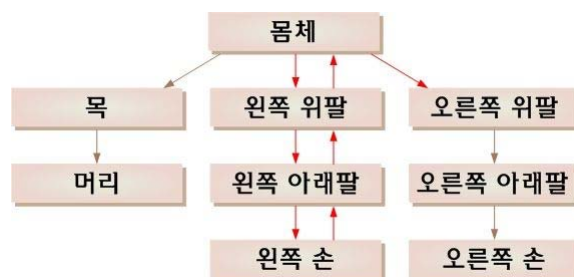
계층구조 트리 관통

- 현 좌표계 저장을 위해서 푸시
- 직전 좌표계 복원을 위해서 팝

→ 태양계 모델링 ? (p.287~)



[그림 6-66] 좌표계



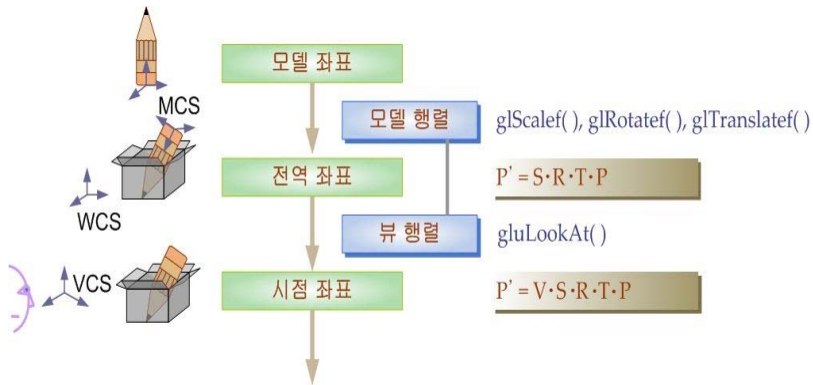
[그림 6-67] 계층구조 트리

- Animation ?
- Forward Kinematics ?
 - Inverse Kinematics ?
 - Motion Capture ?

40

6.4 지엘의 시점 변환- 시점 변환

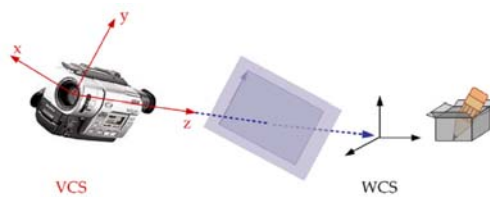
시점 좌표계 (VCS : View Coordinate System)



[그림 6-72] 지엘 파이프라인

41

단순 시스템

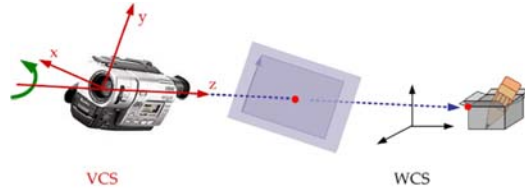


[그림 6-73] 단순한 시점 좌표계

- 카메라 위치 = 시점 좌표계 원점
- 전역 좌표계 원점을 향한 방향이 시점 좌표계의 z축
- z축에 수직으로 서 있는 면= 투상면(Projection Plane, View Plane)
- 투상면 내부에 뷰 윈도우(View Window)= 카메라 필름
- 시점 좌표계 y축 = 뷰 윈도우의 y축과 평행.
- y-z 평면에 수직인 방향으로 x축

42

Renderman 시스템



[그림 6-74] 렌더맨의 시점 좌표계

- 카메라가 바라보는 점 = **초점(Focus, Target)**
- 초점을 향한 방향이 z 축
- 나머지는 단순 시스템과 유사하다.
- z 축을 중심으로 카메라가 회전 가능
 - **롤링(Roll, Rolling)**
 - 초점은 고정된 채 카메라가 시선 방향인 z 축을 기준으로 회전

43

Renderman 시스템

- 시점좌표계, 결과영상, 롤링



[그림 6-75] 시점좌표



[그림 6-76] 결과 영상

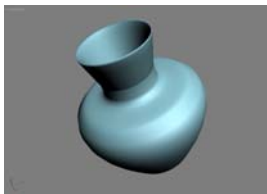
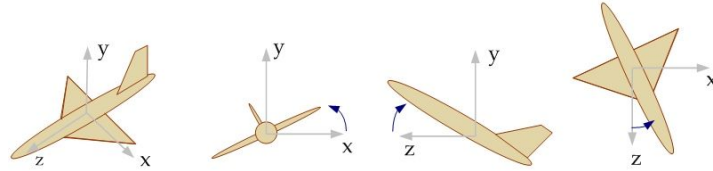


그림 6-77] 롤링

44

비행 시뮬레이션



[그림 6-78] 롤, 피치, 요

Roll = z 축 기준 회전

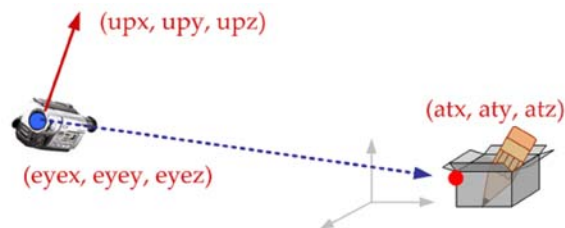
Pitch = x 축 기준 회전

Yaw = y 축 기준 회전


- 각도 변화에 따라서 새로운 축을 x, y, z 축으로 하는 시점 좌표계가 형성
- 조종사의 눈에 보이는 모든 물체는 변화된 새로운 시점 좌표계를 기준으로 변환

45

지엘의 시점 좌표계



[그림 6-80] 지엘의 시점 좌표계

 `void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble atx, GLdouble aty, GLdouble atz, GLdouble upx, GLdouble upy, GLdouble upz);`

 **파라미터**

- 카메라 위치
- 카메라가 바라보는 점, 즉 초점의 위치
- 카메라 기울임(Orientation)

46

시점 좌표, 전역 좌표, 모델 좌표



[그림 6-81] 뷰행렬과 모델행렬

• $P_{WCS} = M \cdot P_{MCS}$

• $P_{VCS} = V \cdot P_{WCS} = V \cdot M \cdot P_{MCS}$

시점변환 함수의 위치

- `glMatrixMode(GL_MODELVIEW);`
- `glLoadIdentity();`
- `gluLookAt(0.2, 0.0, 0.0, 0.0, 0.0, -100.0, 1.0, 1.0, .0);`
- `glRotatef(45, 0.0, 1.0, 0.0);`
- `glutWireCube(1.0);`

I
V
M
P_{MCS}

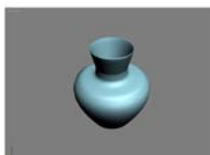
47

틸트, 팬, 롤링, 돌리

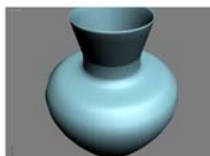


[그림 6-88] 틸트, 팬, 롤, 돌리

돌리



[그림 6-89/90] 원위치, 결과 I



[그림 6-91/92] 돌리, 결과 II

48

실습 내용 (10월 12일)

- 다양한 모델들을 땅 위에 올려놓고 회전시켜 다양한 각도로 볼 수 있도록 함.
 - 바닥(x-z plane)에 격자 표시
 - 좌표축 그리기: X, Y, Z 각각 다른 색으로
 - 5장의 GLUT 모델들을 어떻게 바닥에 올릴지를 설계함
 - Translation/rotation/scaling 및 matrix의 push/pop이용
 - 각 모델들을 그림 (최소 3x3개 이상)
 - 마우스를 이용하여 전체 화면을 다양한 각도로 볼 수 있도록 구현
 - 예: 가로/세로 움직임 → y축/x축 중심의 회전
 - Z축 중심의 회전? 인터페이스 설계

49

실습 내용

- Point3D 클래스를 이해하고 활용한다.
- Translate/Rotate/Scale 메뉴를 프로그램에 넣고 구현한다.
- 자신만의 Robot을 만들어 본다.
- 다양한 방법으로 시점을 변환해 본다.
- GUI
 - 기준좌표계 그리기
 - X-Z 평면에 그리드 그리기
 - 마우스로 Translate/Rotate/Scale 하기 위한 방법은?
 - 고급: 객체를 선택하려면 ?

50