Computer Graphics

유주한

컴퓨터AI학부
동아대학교
2025년 01학기

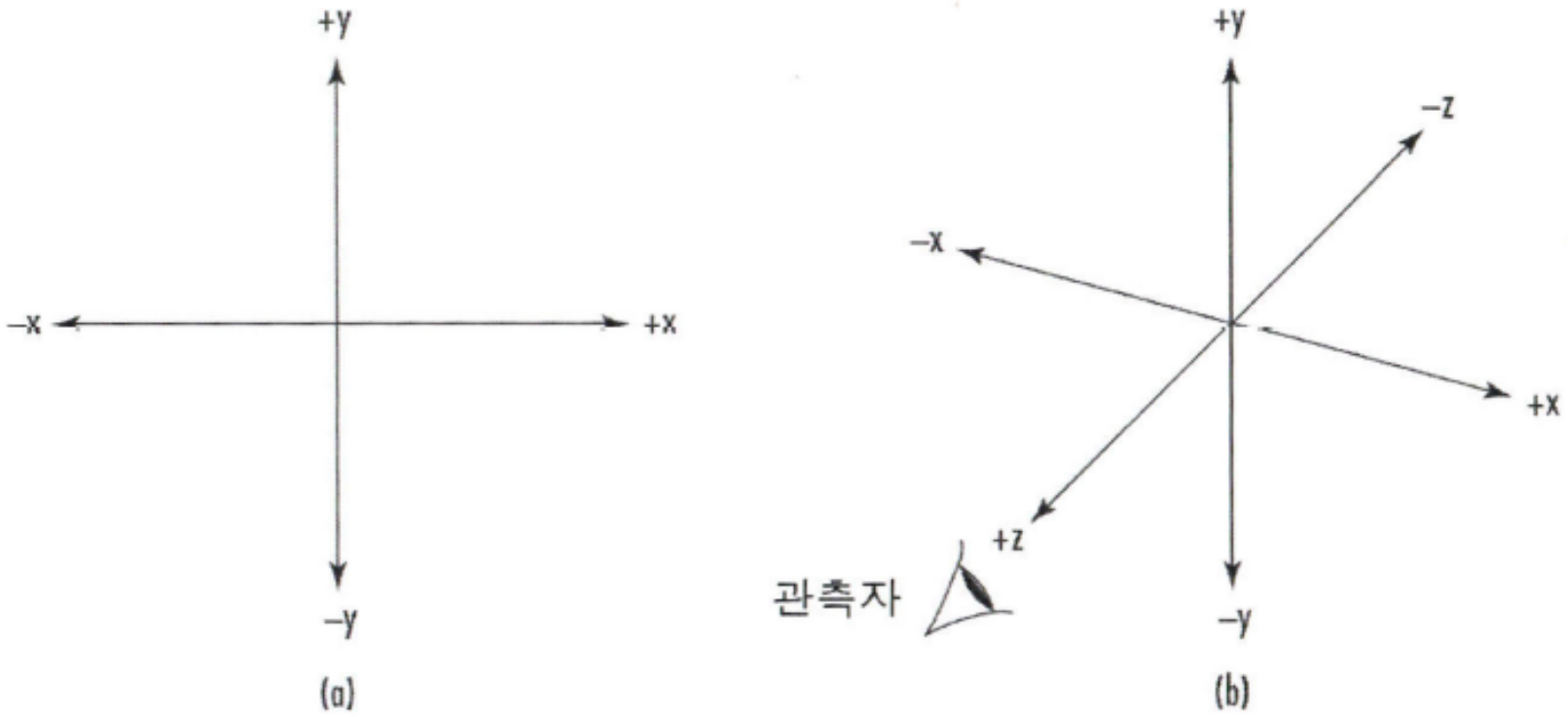
지오메트리 (Geometry)

- 기하학
- 공간에 존재하는 point, line, plan, surface, angle, curvature 등의 특성을 다룸

- 개념 요소들
  - measurement
    - ✓ 길이, 면적, 부피 등 수치적 특성
  - transformation
    - ✓ translation, rotation, reflection, dilation들 어떤 shape이나 solid의 위치나 방향이 바뀌는 것
  - property
    - ✓ 일치, 닮음, 대칭, 평행, 직교 등 어떤 shape이나 solid 사이의 관계

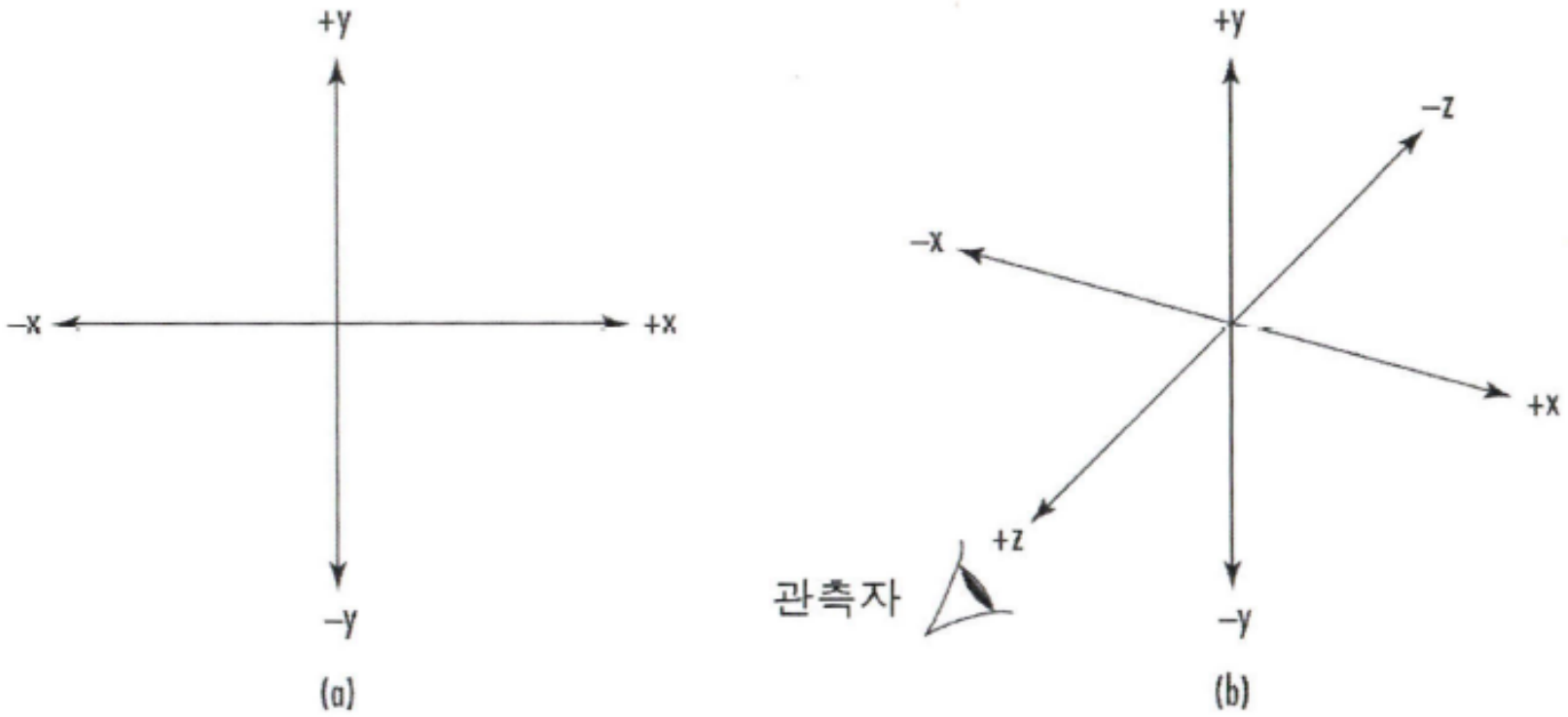
지오메트리 변환

- eye coordinate(시각 좌표)
  - 관측자의 시야를 기준으로 만들어지는 좌표계로, 모든 변환에 영향을 받지 않는 절대적인 화면 좌표
  - 일종의 고정된 가상 좌표계



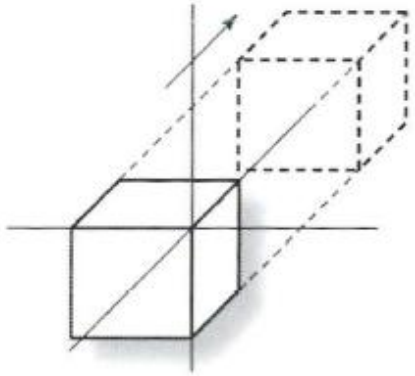
지오메트리 변환

- viewing transformation(관측 변환)
  - 장면의 내용을 어떤위치에서 관측하는지를 결정하는 것
  - 표준적으로는 (0,0,0) 지점에서 음의 z축 방향으로 바라보는 방향

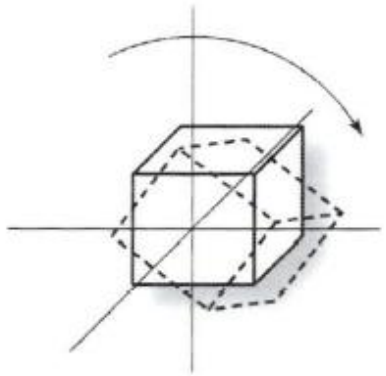


지오메트리 변환

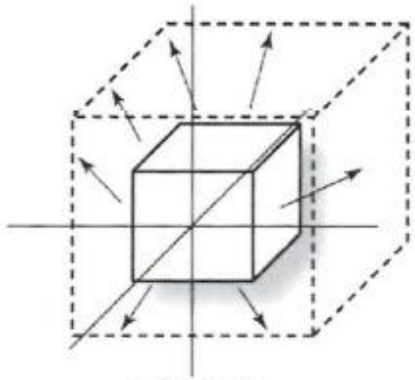
- modeling transformation
  - 특정한 물체를 제어하는 과정으로, 물체를 특정한 위치로 옮기고, 회전시키고, 크기를 바꾸는 것
  - translation
  - rotation
  - scaling



평행 이동



회전



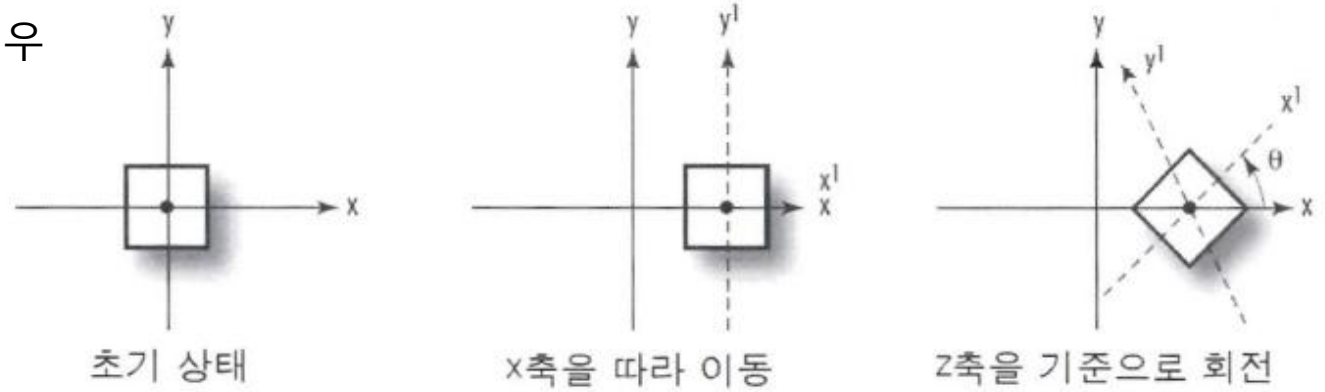
크기 변경

지오메트리 변환

- modeling transformation
  - 동일한 이동과 회전을 순서를 바꾸어서 하면 결과가 달라짐
    - ✓ z축을 기준으로 회전을 먼저 한 다음 x축을 따라 이동시킨 경우

(a)

- ✓ 이동을 먼저 하고 회전시킨 경우

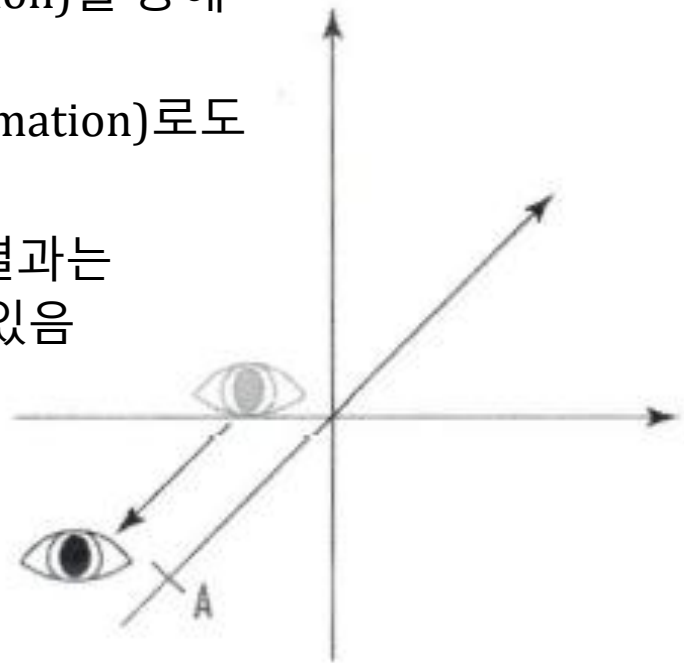


(b)

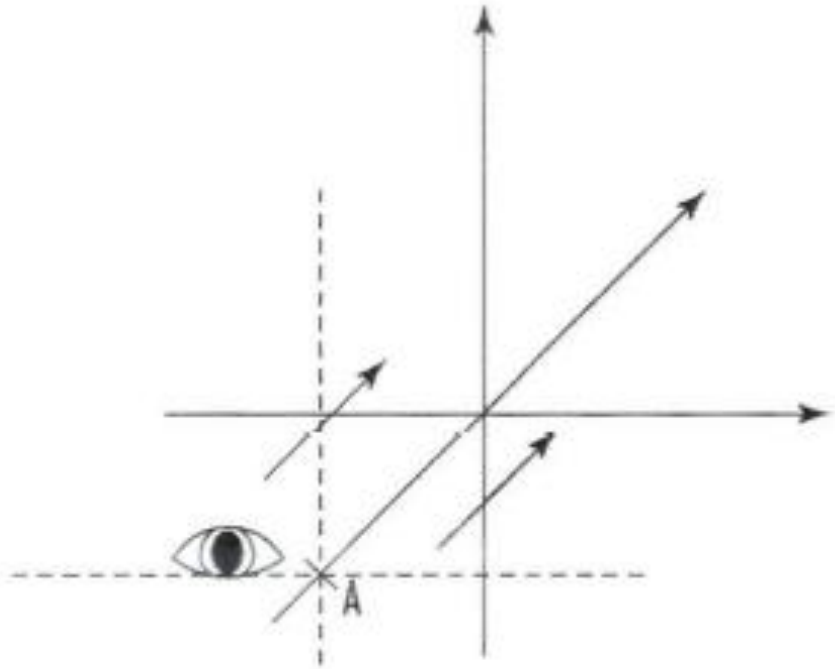
지오메트리 변환

- [1] 모델뷰 변환 (modelview transformation)
  - **모델뷰(modelview) = 관측(viewing) + 모델링(modeling)**
  - 두 변환의 효과가 같으므로 합해서 모델뷰 변환이라고 함

- 모델뷰의 이중성
  - 관측 변환(viewing transformation)을 통해 만들어질 수 있는 결과는 모델링 변화(modeling transformation)로도 만들 수 있음
  - 모델링변환으로 만들 수 있는 결과는 관측 변환을 통해서도 만들 수 있음



관측자가 이동



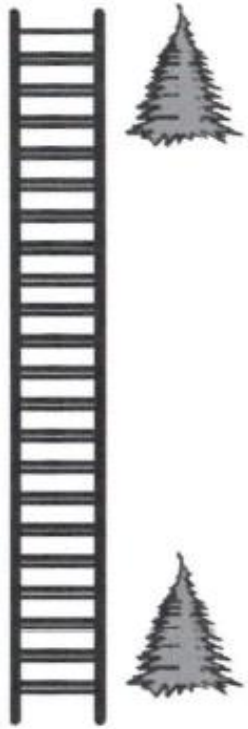
좌표계가 이동

지오메트리 변환

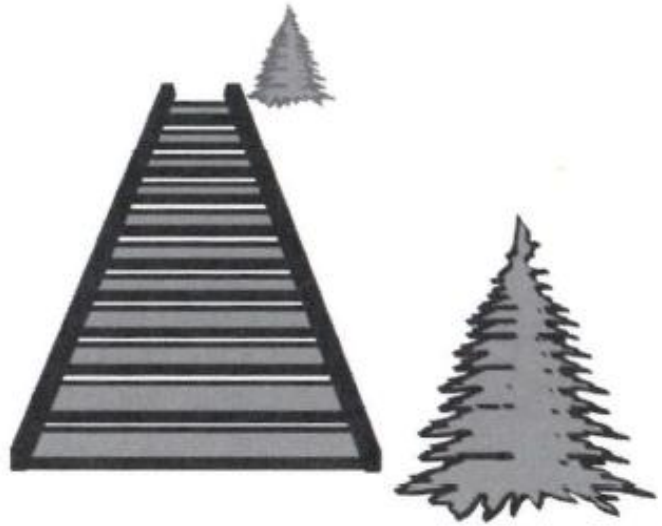
- [2] 투영 변환(projection transfonnation)
  - 모델뷰 변환이 모두 끝난 시점에 이루어지 는 과정
  - **관측 공간**과 **절단면(clipping plane)** 설정
  - **완성된 장면이 화면에 어떻게 보일지를 결정하는 단계**

- 직교 투영 (orthographic projection)
  - 멀리 있거나 가까이 있어도 동일한 크기로 화면에 투영

- 원근 투영(perspective projection)
  - 가까이 있는 물체는 크게, 멀리 있는 물체는 작게 투영



모든 물체의 크기가 같음



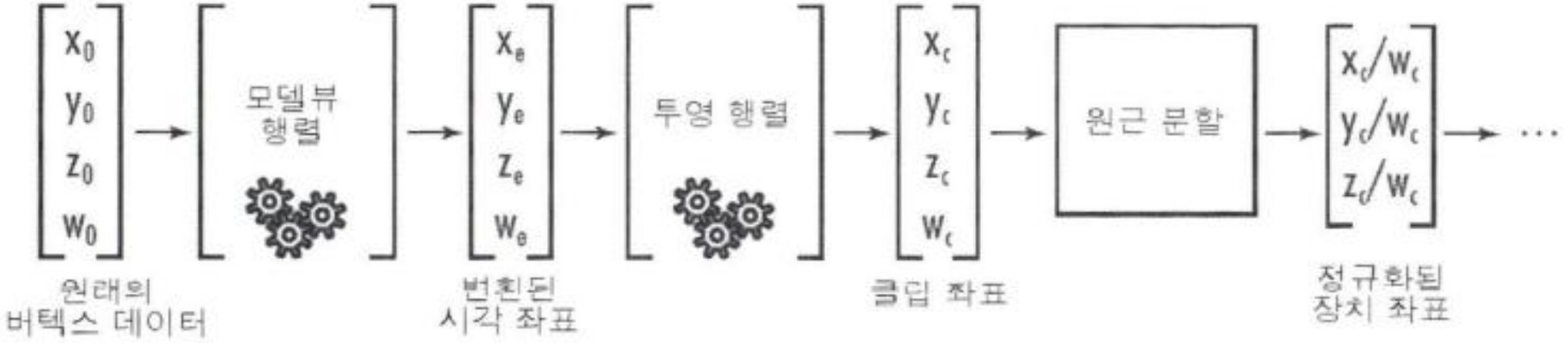
거리에 따라 물체의 크기가 달라짐

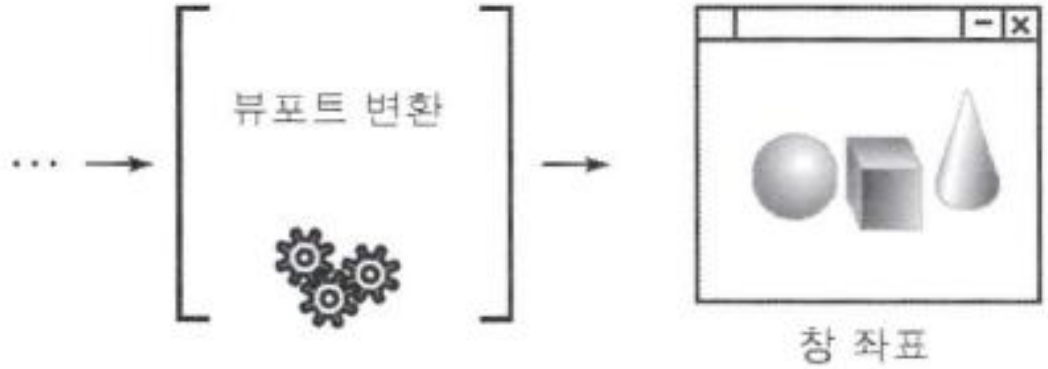
지오메트리 변환

- [3] 뷰포트 변환
  - 모델뷰 변환과 투영 변환이 끝나면 화면에 보여질 2D 이미지가 만들어지는데,
    이를 화면 내의 창에 맞게 변환하는 과정
  - 색상 버퍼와 창의 픽셀 간의 1:1 대응이 일반적이지만 특정한 경우에는 변경이 이루어지기도 함
    - ✓ *고해상도 디스플레이 (예: Retina Display)
      - 논리적 창 크기는 800×600이지만, 실제 디스플레이 픽셀은 1600×1200 (2배 스케일)
      - 하나의 OpenGL 픽셀이 **2×2 물리 픽셀**에 대응됨.
    - ✓ 안티앨리어싱
      - 한 픽셀에 대해 여러 샘플을 계산하여 평균 색상을 사용.
      - 색상 버퍼의 샘플 수 ≠ 출력 픽셀 수
      - 부드러운 경계는 얻지만 1:1 대응은 아님.
    - ✓ 다중 모니터 또는 창 크기 조절
    - ✓ 가상 해상도 (UI 스케일링)

지오메트리 변환

- 변환 파이프라인
  - 일단 버텍스는 1X4 크기의 행렬로 만들어지는데, 네 번째 값인 w는 크기 조정에 사용되는 배율로, 표준값은 1이며 수정할 일은 거의 없음(Homogeneous coordinate)





지오메트리 변환

- 3D 그래픽의 수학적 핵심인 행렬
  - 지금까지 설명한 변환들은 행렬 연산을 통해서 이루어짐
  - 변환 과정을 수학적으로 표현하면 두 행렬의 곱 나타낼 수 있음

지오메트리 변환

- Homogeneous coordinates (동차 좌표)
  - 기하학적인 변환, 특히 **이동(translation)**까지,
    행렬 **곱**으로 표현할 수 있게 해주는 좌표 표현 방식

일반적인 데카르트 좌표계 (Cartesian coordinates)

- 2D: (x, y)
- 3D: (x, y, z)

Homogeneous coordinates로 확장

- 2D: $(x, y) \rightarrow (x, y, 1)$
- 3D: $(x, y, z) \rightarrow (x, y, z, 1)$

> 마지막 원소 w는 **스케일 계수**. 동차 좌표 (x, y, w)는 실제 좌표 $(x/w, y/w)$를 의미

이동 변환의 경우는 사실 행렬 곱셈이 아닌 스칼라 덧셈으로도 새로운 위치의 계산이 가능
하지만 여러 작업이 결합된 복잡한 변환의 경우, 행렬 변환을 사용하는 것이 보다 바람직한 방법

지오메트리 변환

- Homogeneous coordinates (동차 좌표)
  - 기하학적인 변환, 특히 이동(translation)까지,
    행렬 **곱**으로 표현할 수 있게 해주는 좌표 표현 방식

1. 데카르트 좌표계

회전과 확대는 행렬 곱으로 가능하지만, 이동은 덧셈으로만 가능.

1. **확대 행렬 (2×, 3×)**:

$$S = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

2. **회전 행렬 (90도 반시계)**:

$$R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

3. **적용** (확대 → 회전):

곱 연산 $$R \cdot S \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$$

4. **이동**: 벡터 덧셈

합 연산 $$\begin{bmatrix} -3 \\ 4 \end{bmatrix} + \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \end{bmatrix}$$

2. 동차 좌표계

모든 변환을 3×3 행렬 곱으로 표현 가능.

1. **확대 행렬 (2×, 3×)**:

$$S = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. **회전 행렬 (90도 반시계)**:

$$R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. **이동 행렬 (+3, +2)**:

$$T = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

4. **총 변환 행렬**:

곱 연산 $$M = T \cdot R \cdot S$$

- $S \cdot \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$
- $R \cdot \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \\ 1 \end{bmatrix}$
- $T \cdot \begin{bmatrix} -3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 1 \end{bmatrix}$

지오메트리 변환 (모델뷰 행렬)

- 모델뷰 행렬
  - 좌표계 변환을 통해 물체의 위치와 방향을 결정하기 위한 4 X4 행렬
  - 화면에 그려질 기본 모델을 구성하는 각 vertex들은 하나의 열로 구성된 행렬로 만들어지고 모델뷰 행렬과 곱해지고, 새로운 좌표계의 vertex로 변형

$$\begin{bmatrix} X & Y & Z & W \end{bmatrix} \begin{bmatrix} & 4 \times 4 & \\ & M & \end{bmatrix} = \begin{bmatrix} X_e & Y_e & Z_e & W_e \end{bmatrix}$$

✓ 추가적인 값인 w까지 모두 4개의 값을 가진 버텍스 행렬이 모델뷰 행렬과 곱해지고, 그 결과로 다시 같은 형태의 버텍스 행렬이 만들어 짐

✓ w는 크기 변환에 관련된 요소(scaling factor)로, 이 값을 변경할 일은 거의 없을지도...

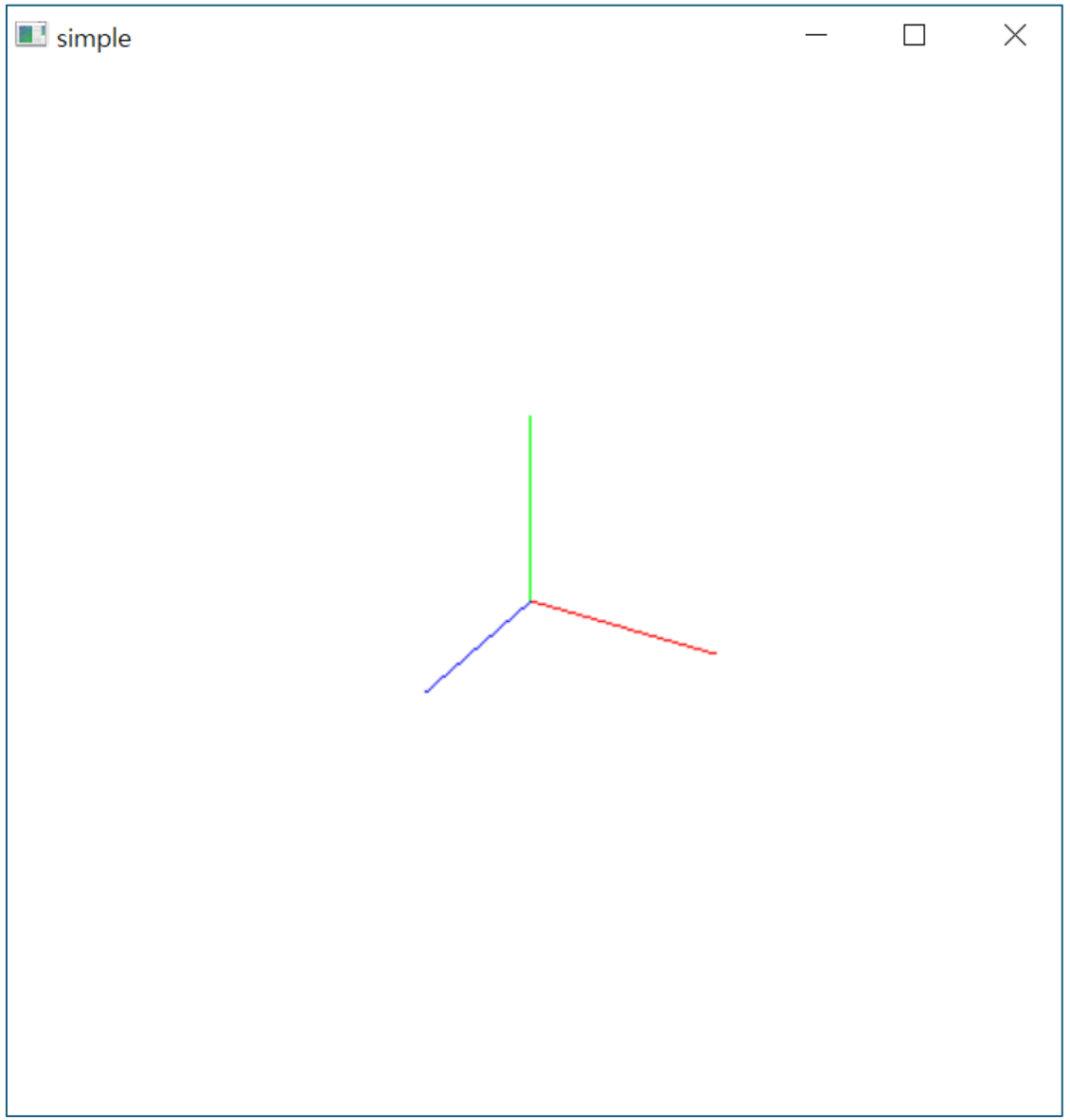
잠깐, vertex

- OpenGL은 "좌표"가 아니라 구조화된 그래픽 단위로 처리해야 하므로 point보다 vertex를 사용
  - 수학이나 물리에서는 point = 위치만 존재
  - OpenGL에서는 위치 + 그래픽 정보 → 그래서 vertex로 구분

항목	Point (점)	Vertex (버텍스)
의미	공간상의 위치	도형의 구성 요소 (예: 삼각형의 꼭짓점)
정보	보통 좌표만 (x,y,z)(x, y, z)	좌표 + 색상, 법선, 텍스처 등 다양한 속성
사용 맥락	수학, 물리, 일반 공간 개념	그래픽스, 메시(mesh), 폴리곤 모델
예시	(1, 2, 3): 단순한 점	(1, 2, 3, 색, 노멀, 텍스처 좌표): 삼각형의 꼭짓점 등

지오메트리 변환 (4_1_a)

- “4_1_a” code는 다음과 같은 view 방향을 가지고 있음



지오메트리 변환 (4_1)

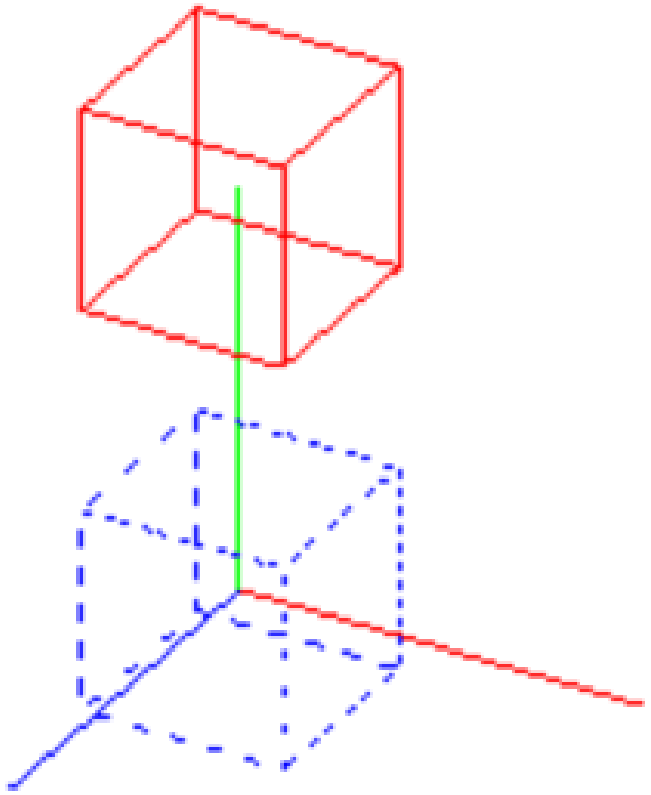
Try

4_1_a code를 기반으로 작성하시오

- 이동
  - 예) GLUT 라이브러리의 glutWireCube 함수를 사용하여 육면체를 그리고, 아래 출력을 만드시오

```
glutWireCube(10.0f);
```

✓ **원점**을 중심으로 하여 10 크기만큼의 변 길이를 가지는 육면체



- **y축 양의 방향으로 10 만큼 이동하는 변환 행렬 설정하여 그리기**
- **모델뷰 기준으로 R,T가 적용되게 함**

✓ **glMatrixMode(GL_MODELVIEW);**

```
glTranslatef(GLfloat x, GLfloat y, GLfloat z);
```

✓ x, y, z 방향으로 이동할 거리를 인자로 받아 적절한 행렬을 만들고, 곱함

- 왼쪽 그림의 점선 큐브의 중심은 (0,0,0) 임