3차원 그래픽스의 뷰잉(관측) 1

Prof. Kim, Soo Kyun

사진 촬영에 필요한 것에 대해 생각해보자

사진 촬영에 필요한 것에 대해 생각해보자

사진 촬영에 필요한 것에 대해 생각해보자

- 무엇이 있을까?

사진 촬영에 필요한 것에 대해 생각해보자

- 무엇이 있을까??
  - 피사체(모델): 사람, 산, 건물 …

사진 촬영에 필요한 것에 대해 생각해보자

- 무엇이 있을까??
  - 피사체(모델): 사람, 산, 건물 …
  - 피사체(모델)들 간의 관계
    - 피사체(모델)의 어떤 모습을 찍을 것인가?
    - 피사체(모델)을 회전할까?
    - 피사체(모델)을 이동할까?

사진 촬영에 필요한 것에 대해 생각해보자

- 무엇이 있을까??
  - 피사체(모델): 사람, 산, 건물 …
  - 피사체(모델)들 간의 관계
    - 피사체(모델)의 어떤 모습을 찍을 것인가?
    - 피사체(모델)을 회전할까?
    - 피사체(모델)을 이동할까?
  - 빛 (조명)
    - 인간의 눈은 빛만 인지하죠!!!!
    - 빛이 어디에 있나?

사진 촬영에 필요한 것에 대해 생각해보자

- 무엇이 있을까??
  - 모델… (사람, 산, 건물 …)
  - 모델들 간의 관계
    - 모델의 어떤 모습을 찍을 것인가?
    - 모델을 회전할까?
    - 모델을 이동할까?
  - 빛 (조명)
    - 인간의 눈은 빛만 인지하죠!!!!
    - 빛이 어디에 있나? (위치)
  - 피사체(모델)들이 입거나, 가지고 있는 색
    - 옷, 신발, 가방, 건물의 색, 두건…

사진 촬영에 필요한 것에 대해 생각해보자

- 무엇이 있을까??
  - 모델… (사람, 산, 건물 …)
  - 모델들 간의 관계
    - 모델의 어떤 모습을 찍을 것인가?
    - 모델을 회전할까?
    - 모델을 이동할까?
  - 빛 (조명)
    - 인간의 눈은 빛만 인지하죠!!!!
    - 빛이 어디에 있나?
  - 피사체(모델)들이 입거나, 가지고 있는 색
    - 옷, 신발, 가방, 건물의 색, 두건…
  - 카메라를 이용한 사진 촬영 (우리가 설정한 것을 담아야죠)
    - 클리핑 (Clipping) : 우리가 담는 카메라는 이 세상 모든 것을 담을 수 없음

렌더링 파이프라인

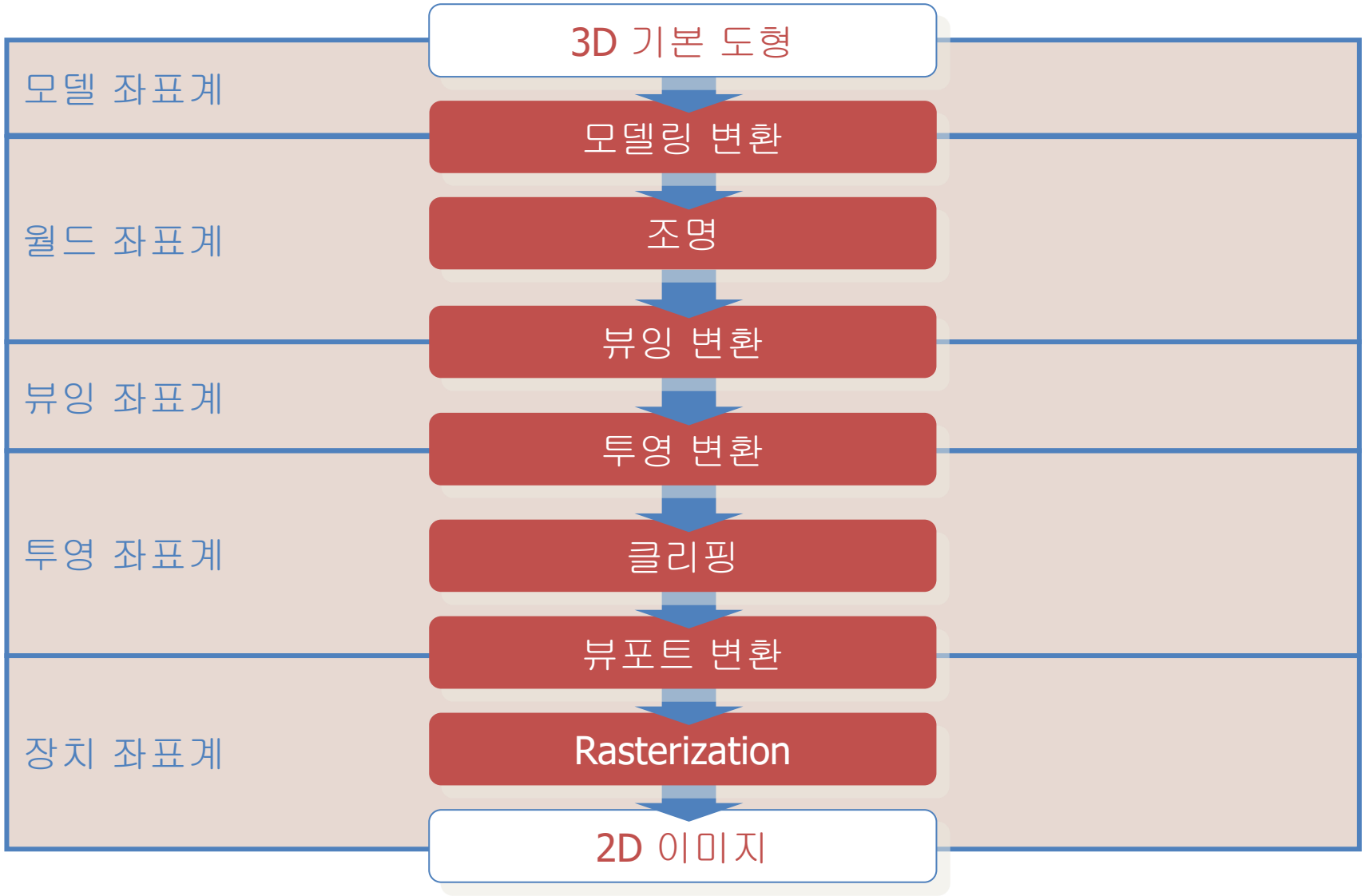
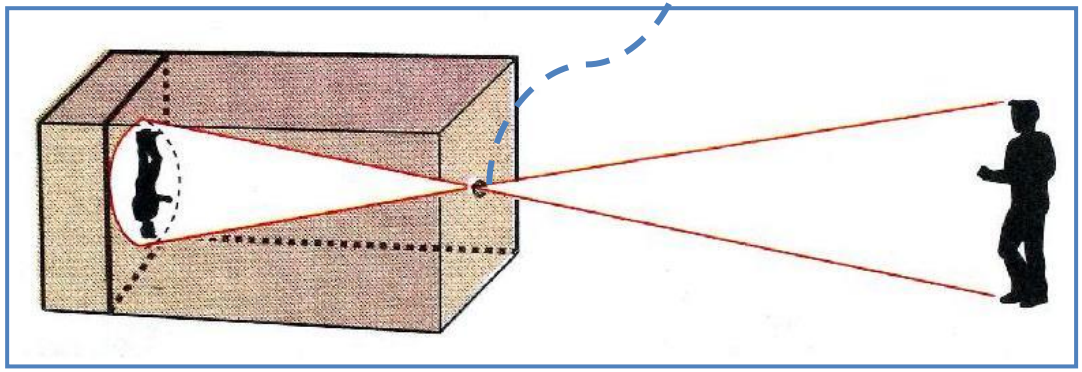

3D 기본 도형
모델 좌표계
모델링 변환
월드 좌표계
조명
뷰잉 변환
뷰잉 좌표계
투영 변환
투영 좌표계
클리핑
뷰포트 변환
장치 좌표계
Rasterization
2D 이미지

사진 촬영에 필요한 것에 대해 생각해보자

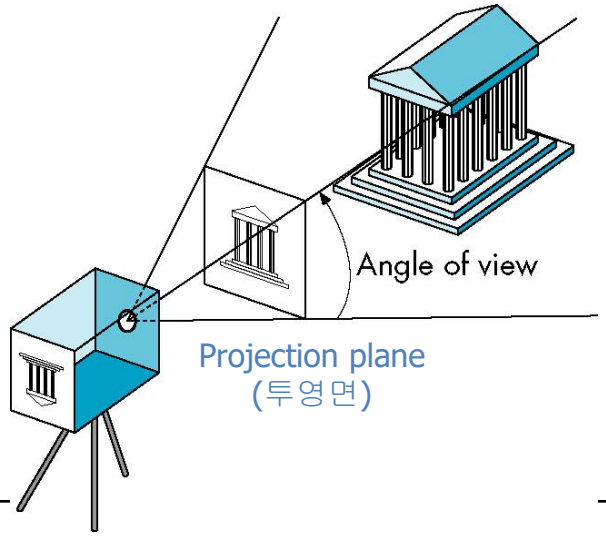
- 무엇이 있을까??
  - 피사체(모델):사람, 산, 건물 …→ 지역좌표 (모델좌표)
  - 피사체(모델)들 간의 관계 → 세계좌표 (월드좌표)
    - 무엇을 기준으로 이동, 회전 할까
  - 빛 → 조명
    - 인간의 눈은 빛만 인지하죠!!!!
  - 피사체(모델)들이 입거나, 가지고 있는 색 → 쉐이딩
    - 옷, 신발, 가방, 건물의 색, 두건…
  - 카메라 (우리가 설정한 것을 담아야죠) →**뷰잉(관측Viewing)**
    - 투영 (Projection) : 카메라 영상은 2차원 → **투영**
    - 클리핑 (Clipping) : 우리가 담는 카메라는 이 세상 모든 것을 담을 수 없음 → **클리핑, 뷰포트**

카메라의 원리와 합성 카메라

- 핀홀 카메라 (Pinhole camera)

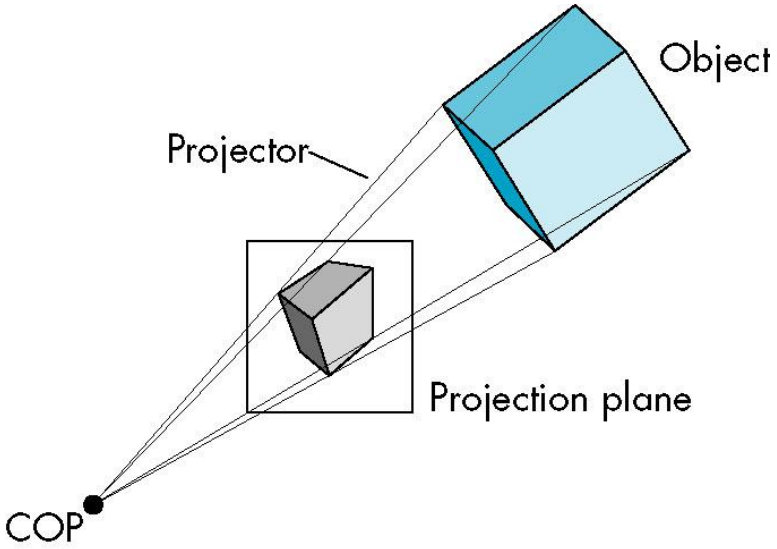


- 합성 카메라(Synthetic camera model)

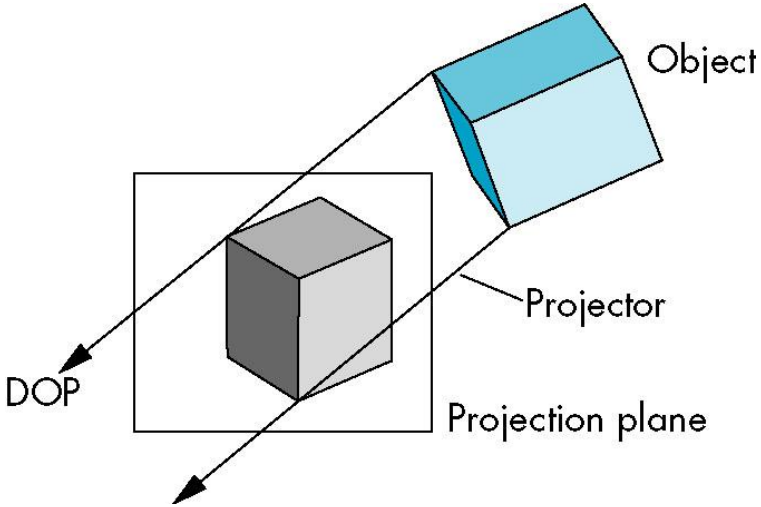


뷰잉(관측)의 기본적인 종류

- 투시 투영
  - 유한 COP (center of projection)
- 평행 투영
  - 무한 COP ➔ DOP (direction of projection)



투시 투영



평행 투영

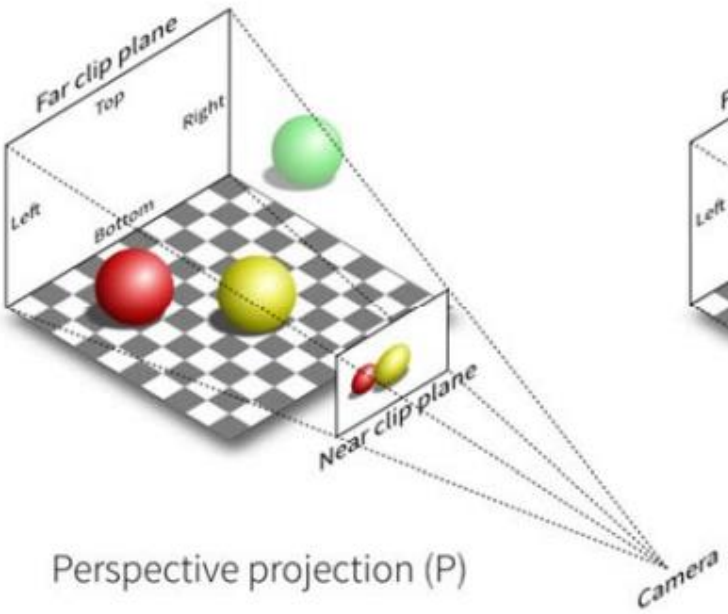
투영과 카메라

PerspectiveCamera

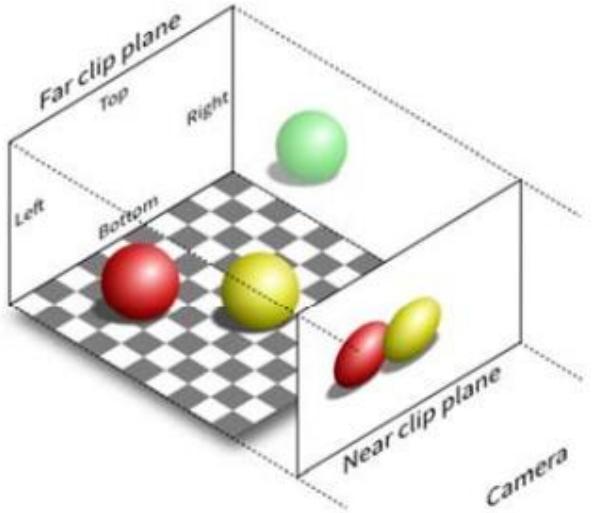
- 사람의 눈이 보는 방식
- 거리감이 존재

OrthographicCamera

- 카메라와의 거리에 관계없이 일정하게 보임
- 2D Scene 또는 UI 요소 렌더링에 사용

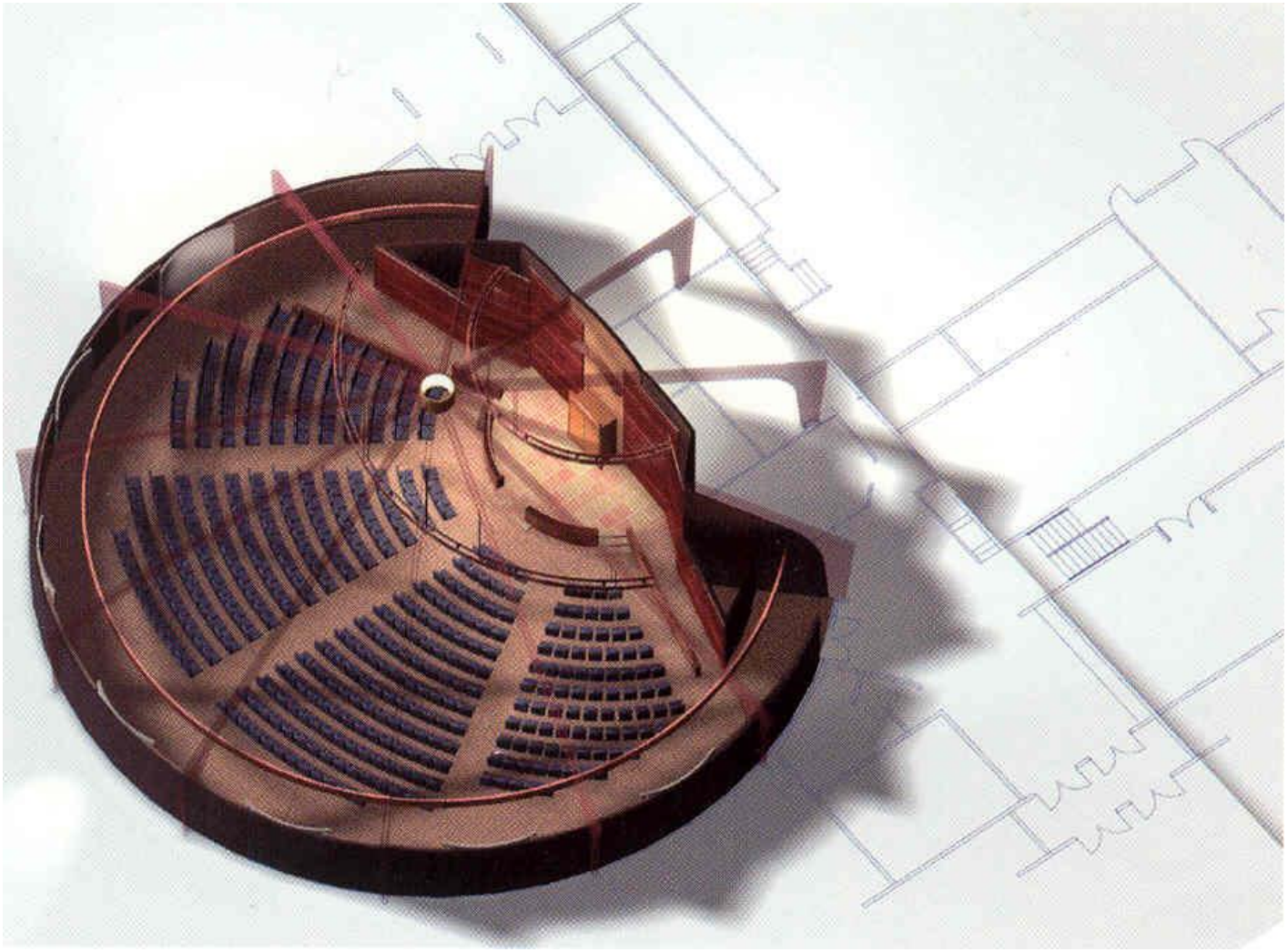


Perspective projection (P)

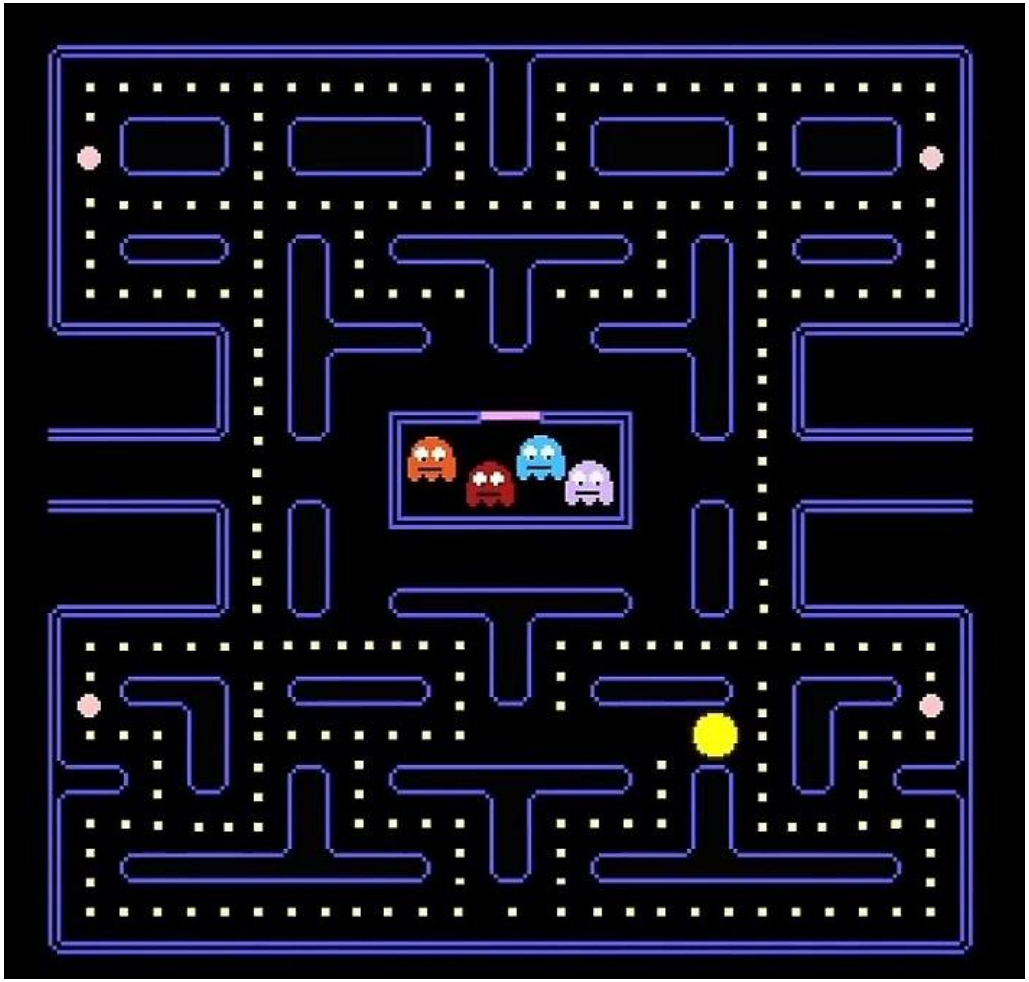


Orthographic projection (O)

평행 투영 (Parallel Projection)



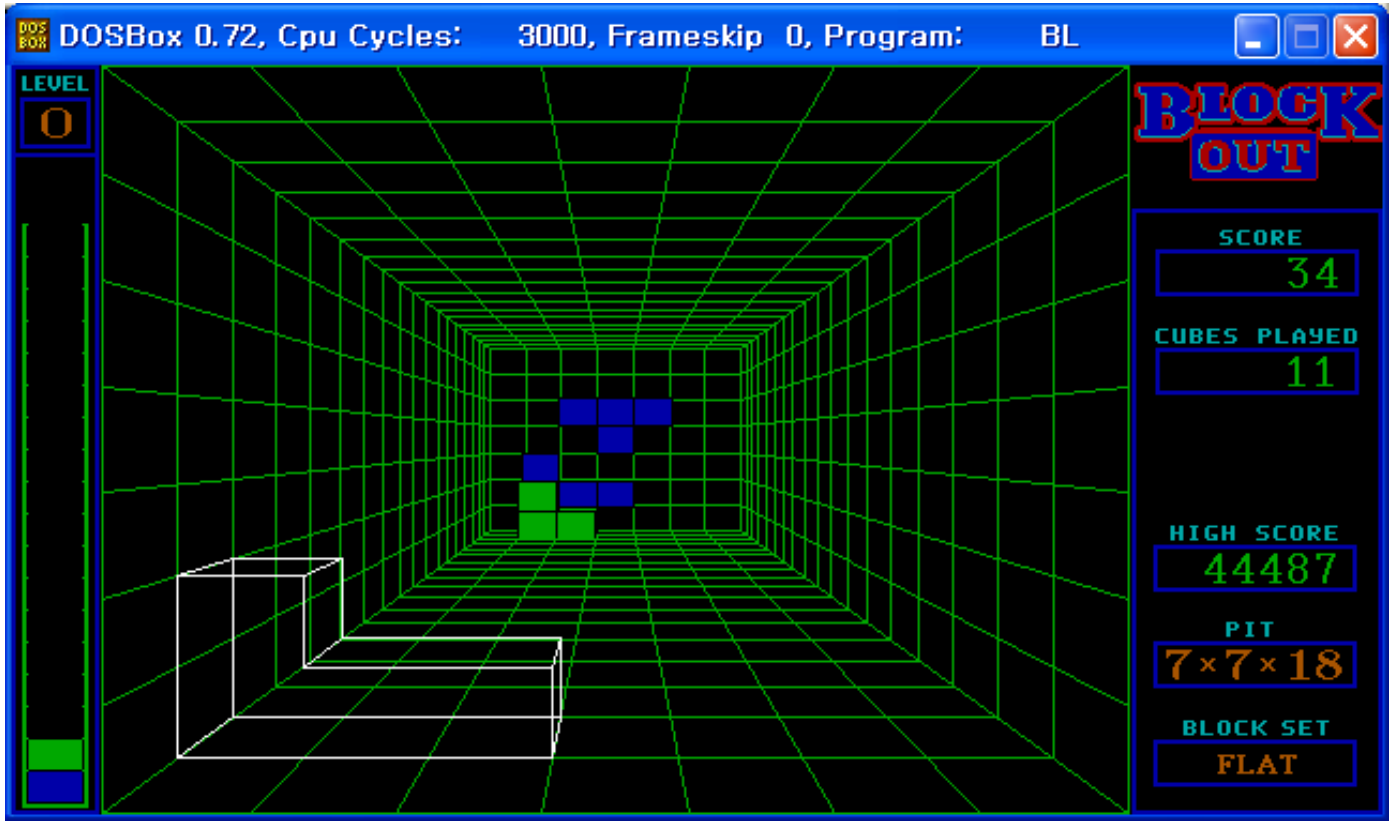
평행 투영 (Parallel Projection)



팩맨

투시 투영 (Perspective Proj.)

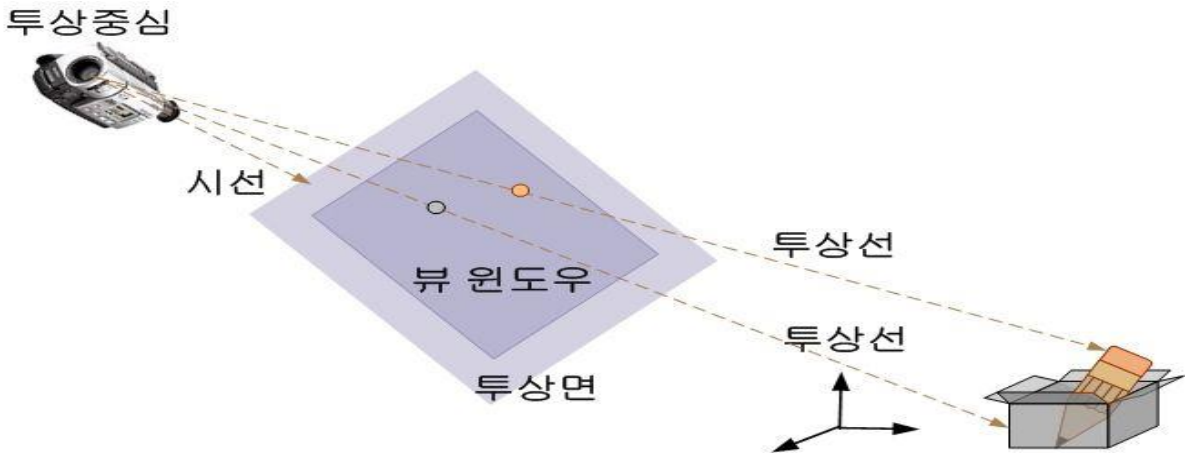
투시 투영 (Perspective Proj.)



3D 테트리스

투영 (Projection)

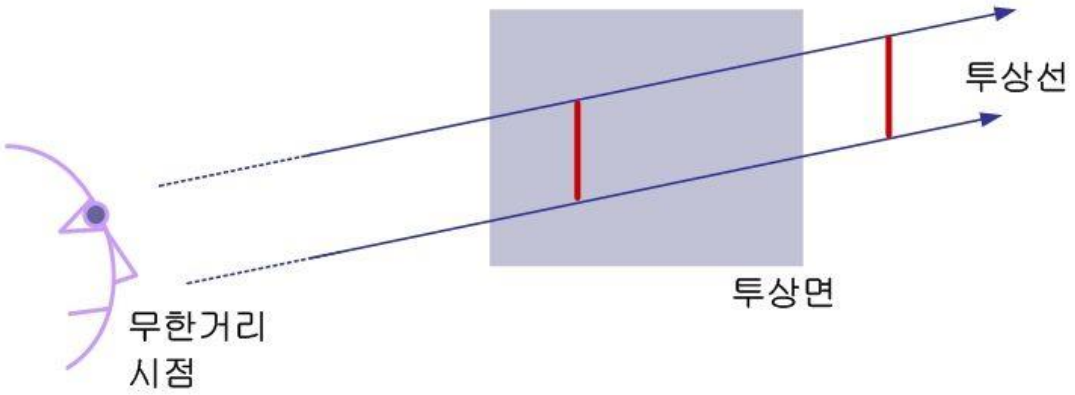
- **투영(Projection) = 관측변환(Viewing Transformation)**
  - 모델 좌표계, 전역 좌표계 (세계 좌표계), 시점 좌표계를 순차적으로 거친 다각형 정점 좌표를 2차원 투영면에 사상(mapping)시키는 과정



투상선(투영선)과 투상면(투영면)

  - 투영면(View Plane, Projection Plane)
  - 관찰자 위치(View Point, Eye Position)
    = 카메라 위치(Camera Position) = 투영중심(COP: Center of Projection) = 시점좌표계 원점(Origin of VCS)
  - 투영선(Projectors): 물체 곳곳을 향함
  - 시선(Line of Sight) : 초점을 향함

투영 (Projection)



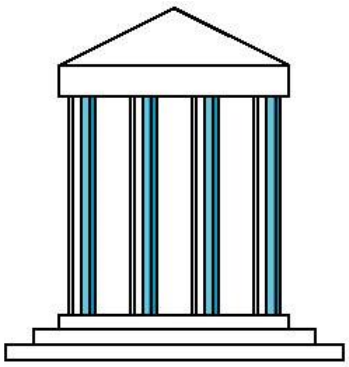
평행투영

- **시점이 물체로부터 무한대의 거리에 있다고 간주**
  - 투영선이 평행
  - 원래 물체의 평행선은 투영 후에도 평행
  - 시점과의 거리에 무관하게 같은 길이의 물체는 같은 길이로 투영
- **정사투영, 축측투영, 경사투영 등으로 분류**

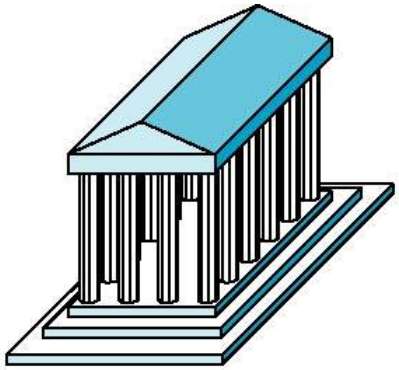
고전적인 뷰잉

- 물체와 시점 사이의 특정 관계를 가짐

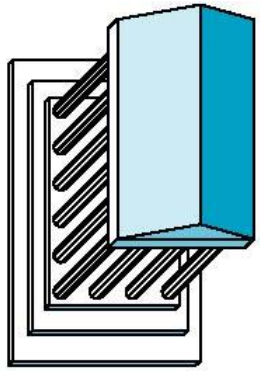
pp.216참조



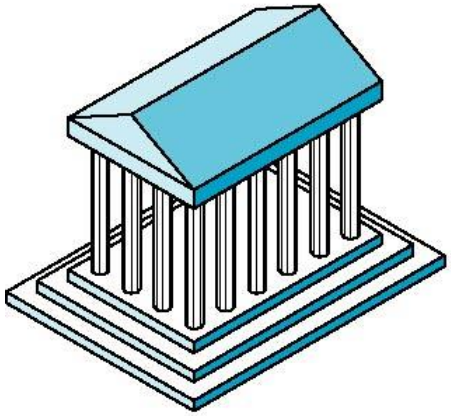
Front elevation
정면



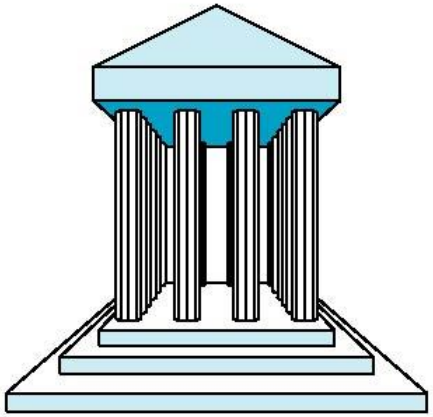
Elevation oblique
입면경사



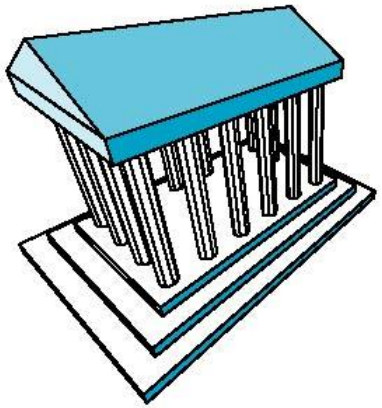
Plan oblique
평면경사



Isometric
등축



One-point perspective
1점 투시



Three-point perspective
3점 투시

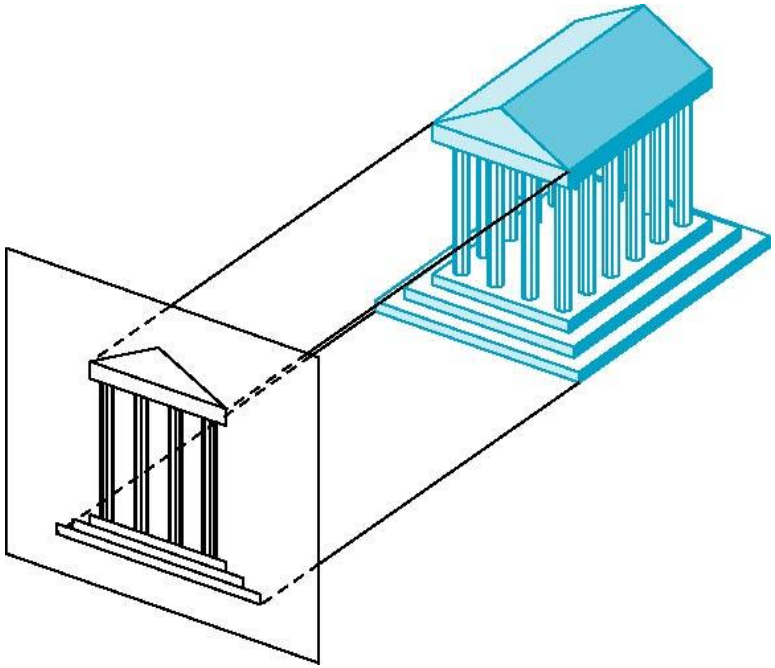
용어 정리

- **등축 (Isometric)**
  - 3차원 물체를 평면 상에 표현하기 위한 방법의 일종으로 *x*, *y*, *z* 세 좌표축이 서로 이루는 각도가 모두 같거나 120도를 이루는 특성

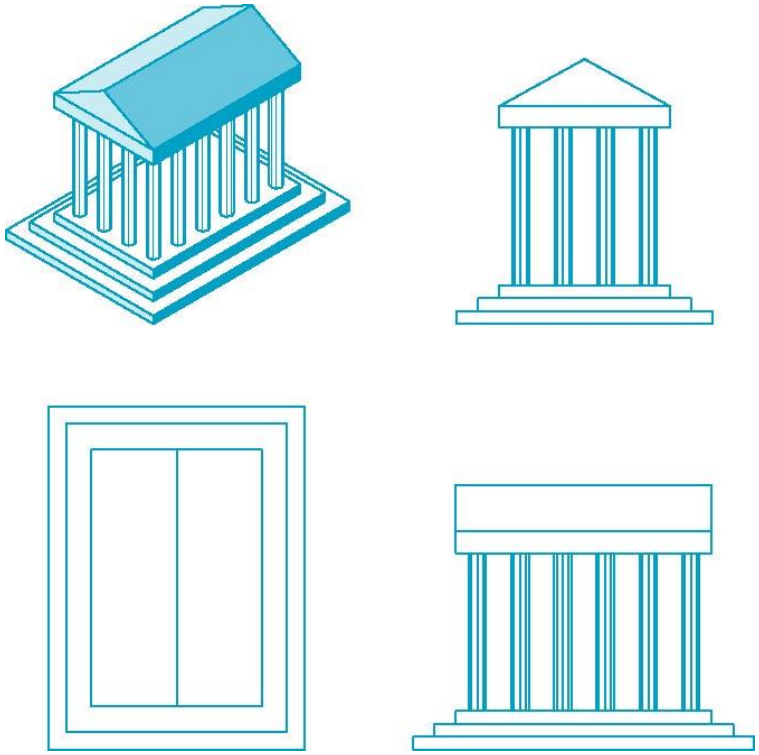
직교 투영 (Orthographic Proj.)

- 투영 방향과 투영면이 직각을 이루는 경우
  - 길이와 각도가 유지됨

pp.216참조



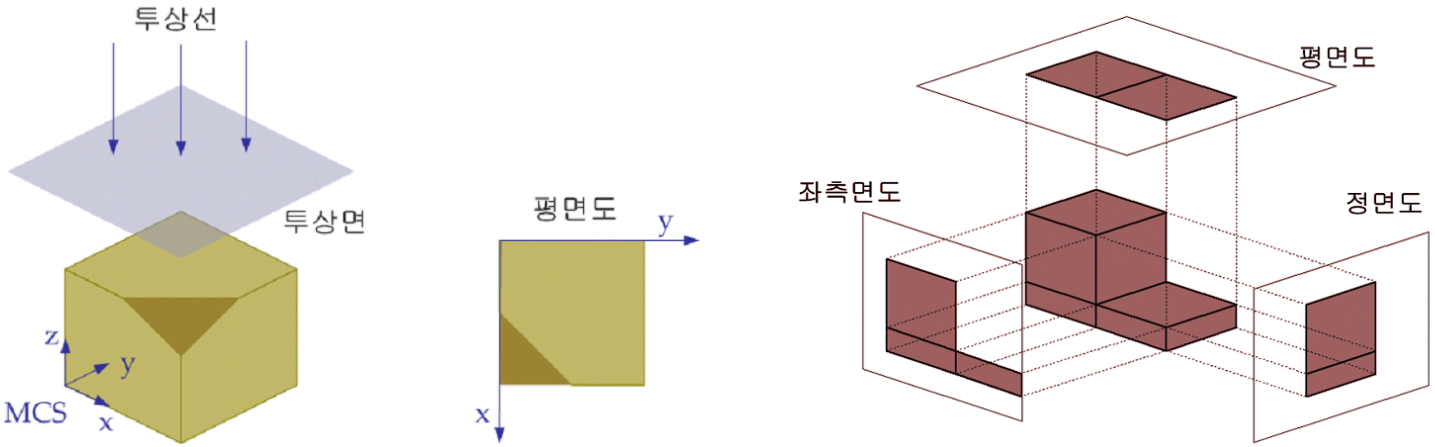
직교 투영



'사원'과 직교 투영을 이용한 삼면도

직교 투영 (Orthographic Proj.)

- **평면도, 입면도, 측면도 등**
  - 주면(Principal Plane): MCS 주축인 x, y, z에 의해 형성되는 x-y, y-z, z-x를 주면이라고 하며,
  - 투영면은 주면 중 하나와 평행
- **투영선(투상선)은 투영면(투상면)과 직교**
  - 원래 물체의 길이를 정확히 보존 → 공학도면에 사용
  - 투영선이 반드시 투영면과 직교 → 시점위치가 제한됨

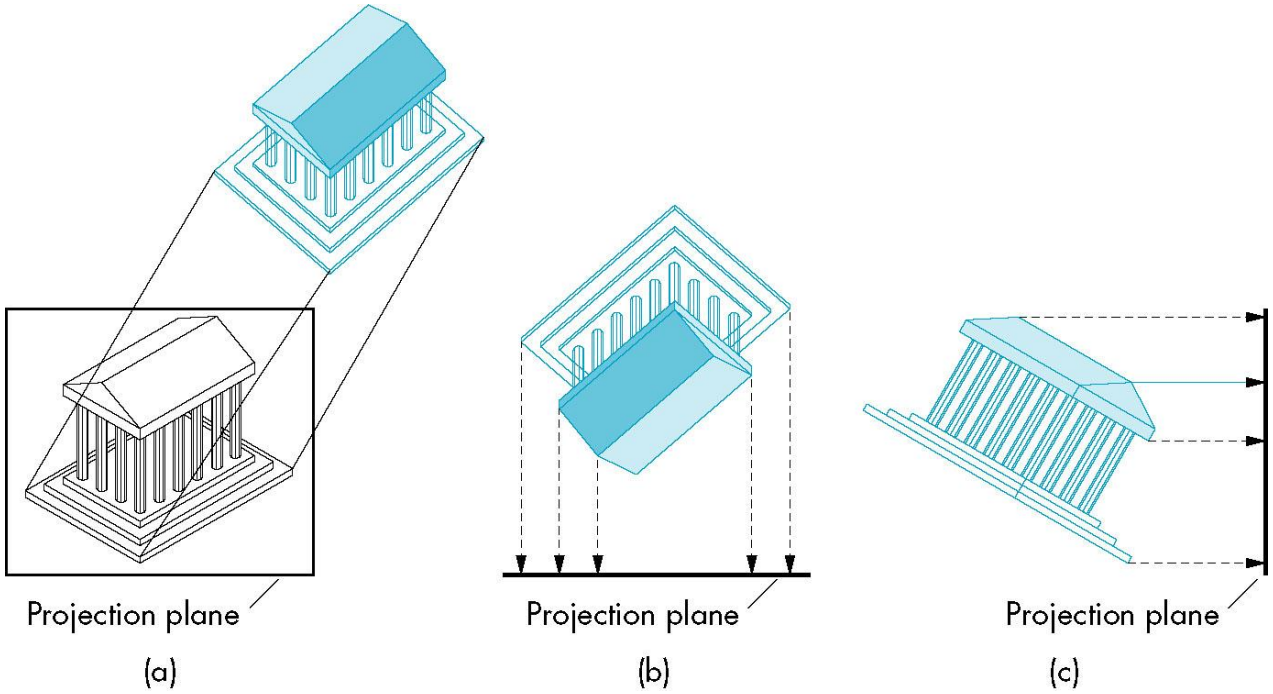


직교 투영

축측 투영 (Axonometric Pro.)

pp.217참조

- 투영면이 임의의 위치에 놓임
  - 투영 방향과 투영면은 직각을 이룸



Axonometric 투영

위

옆

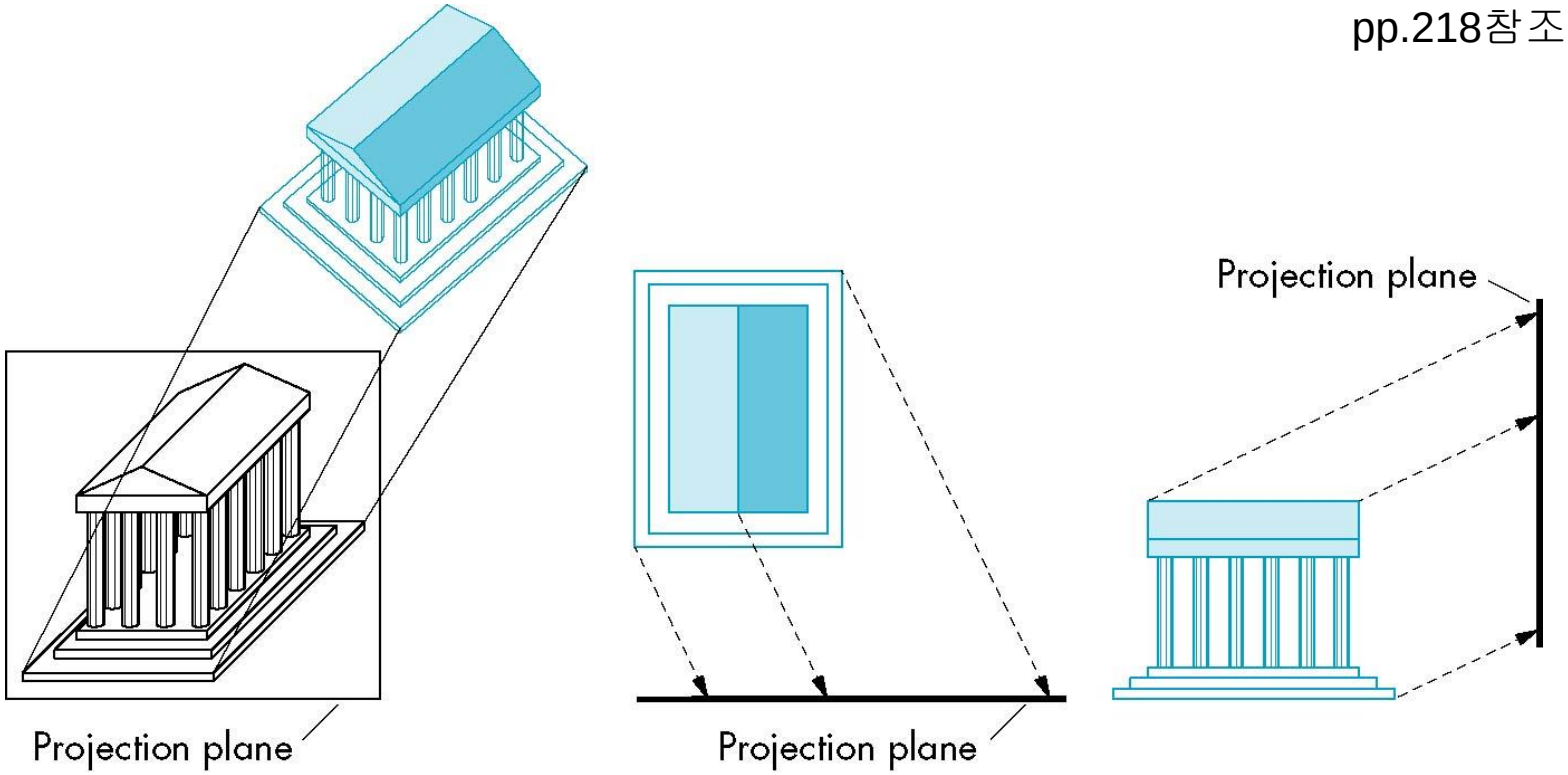
용어 정리

- 축측 세 각이 모두 같으면, 등각 투영(Isometric), 두 각이 같으면 이각 투상(Demetric), 모두 다르면 삼각 투상(Trimetric)

경사 투영 (Oblique Projection)

- 투영 방향과 투영면이 수직이 아님
  - 투영면에 평행한 면들에 대해서만 각이 유지됨

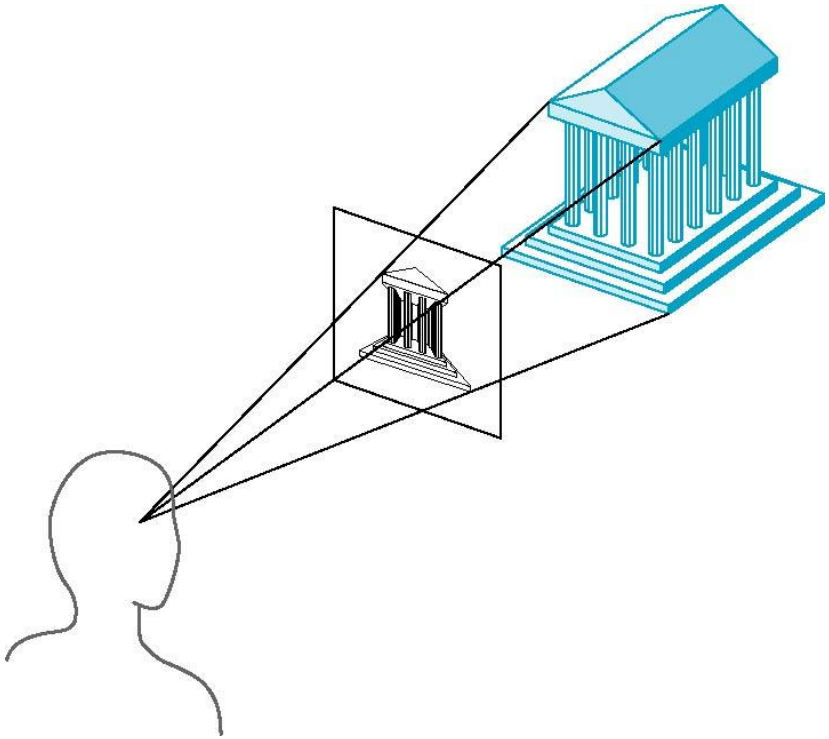
pp.218참조



경사 투영 위 옆

투시 투영 (Perspective Proj.)

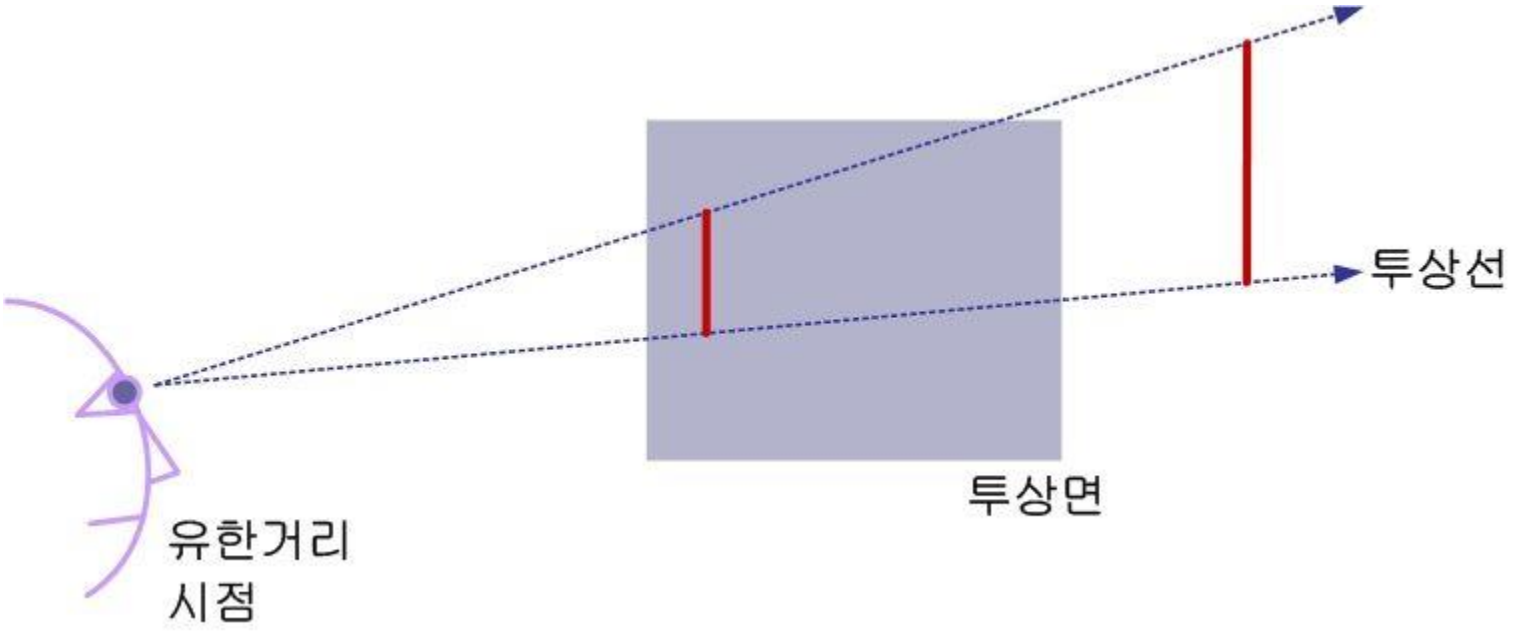
- 현실감 있음
  - 투영면에서 멀리 떨어져 있는 객체는 작게, 가까운 거리에 있는 객체는 상대적으로 크게 투영



pp.219참조

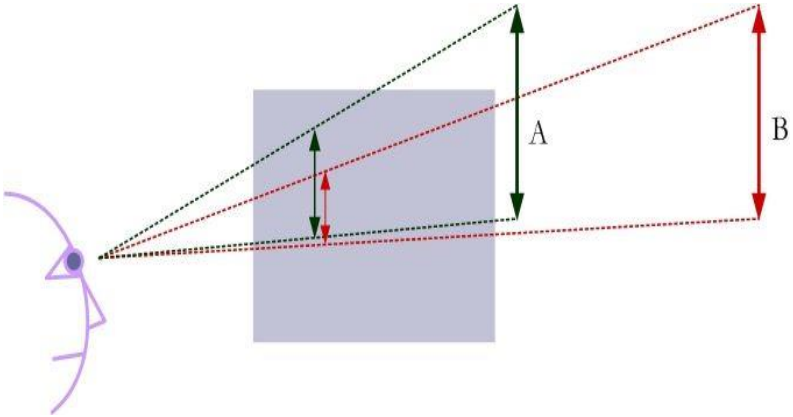
투시 투영 (Perspective Proj.)

- 시점이 물체로부터 유한한 거리에 있다고 간주
- 투영선이 시점에서 출발하여 방사선 모양으로 퍼져감
- 카메라나 사람의 눈이 물체를 포착하는 방법

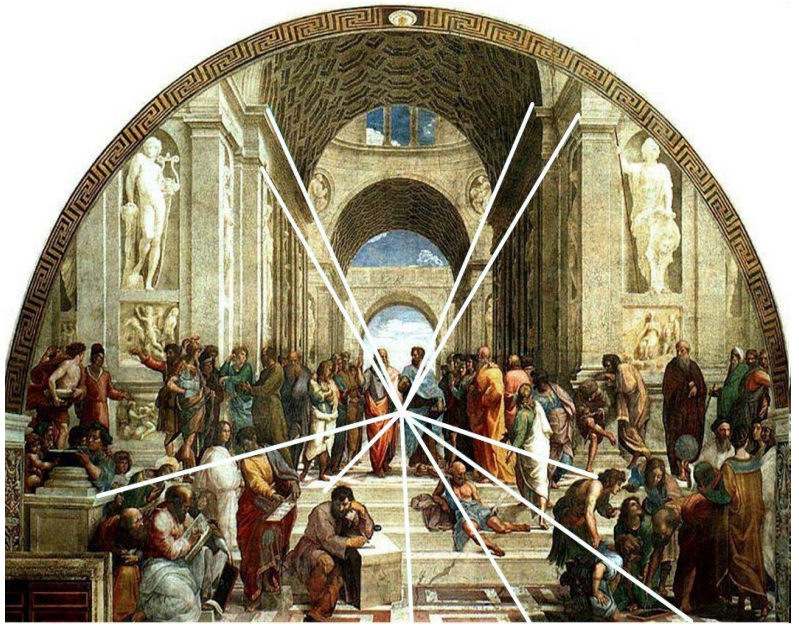


투시 투영(Perspective Projection)

- **원근감(Depth Feeling)**
  - 동일한 크기의 물체라도 시점으로부터 멀리 있는 것은 작게 보이고 가까운 것은 크게 보임



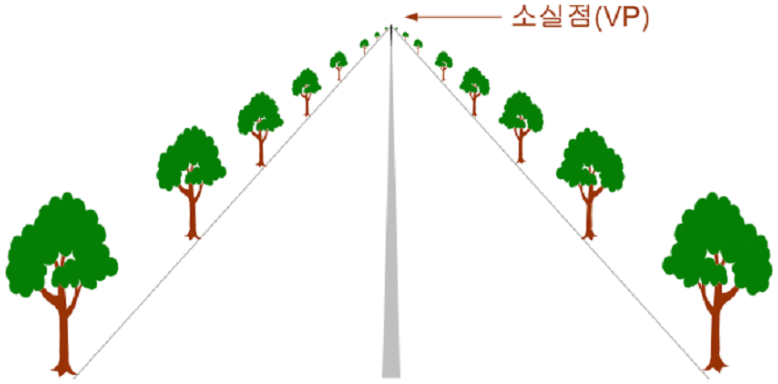
축소율 차이



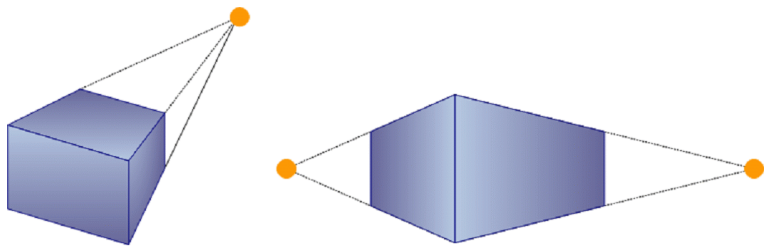
라파엘,“ 아테네학당”

투시 투영(Perspective Projection)

- **소실점(VP: Vanishing Point)**
  - 투시투영 결과 평행선이 만나는 점
  - 소실점의 수
    - 일점투영(One-point Projection), 이점투영(Two-point Projection), 삼점투영(Three-point Projection)
- **투시투영변환(Perspective Transformation)**
  - 직선→직선, 평면 → 평면
  - 물체 정점간의 거리에 대한 축소율이 달라짐



소실점



일점투영과 이점투영

투시 투영의 종류

- **One-, two-, and three-point perspectives**
  - 소실점(vanishing point)의 개수
  - 3개 주요 방향(x, y, z축) 중 몇 개가 투영면과 평행인가?

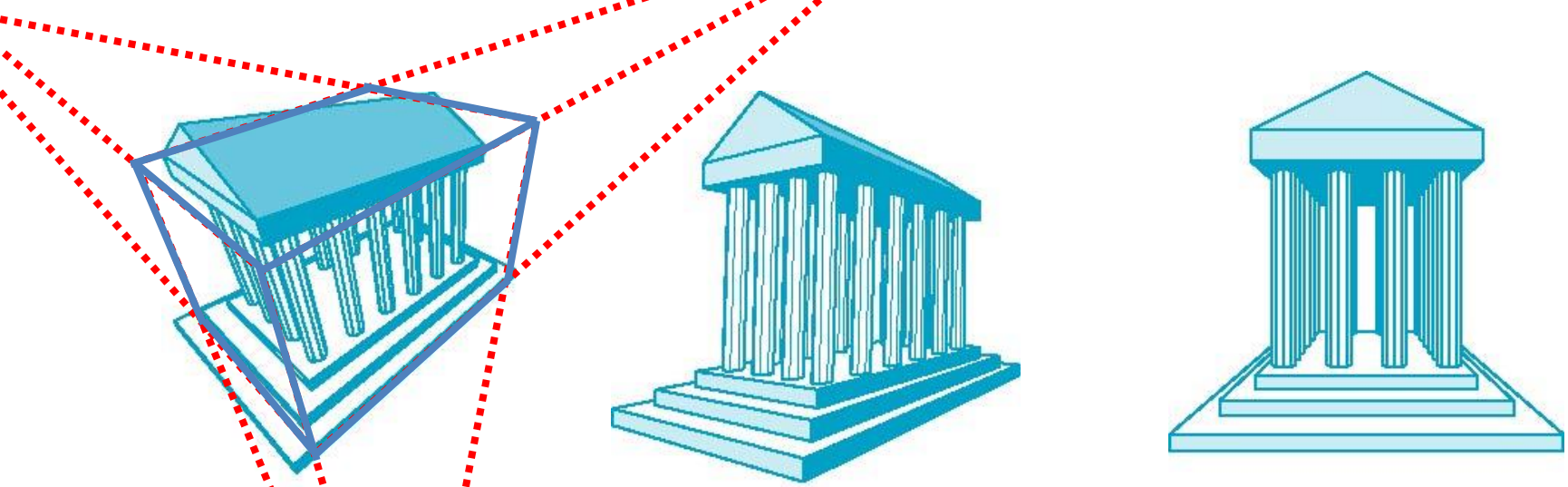
three-point perspective

two-point perspective

one-point perspective

투시 투영의 종류

- **One-, two-, and three-point perspectives**
  - 소실점(vanishing point)의 개수
  - 3개 주요 방향(x, y, z축) 중 몇 개가 투영면과 평행인가?



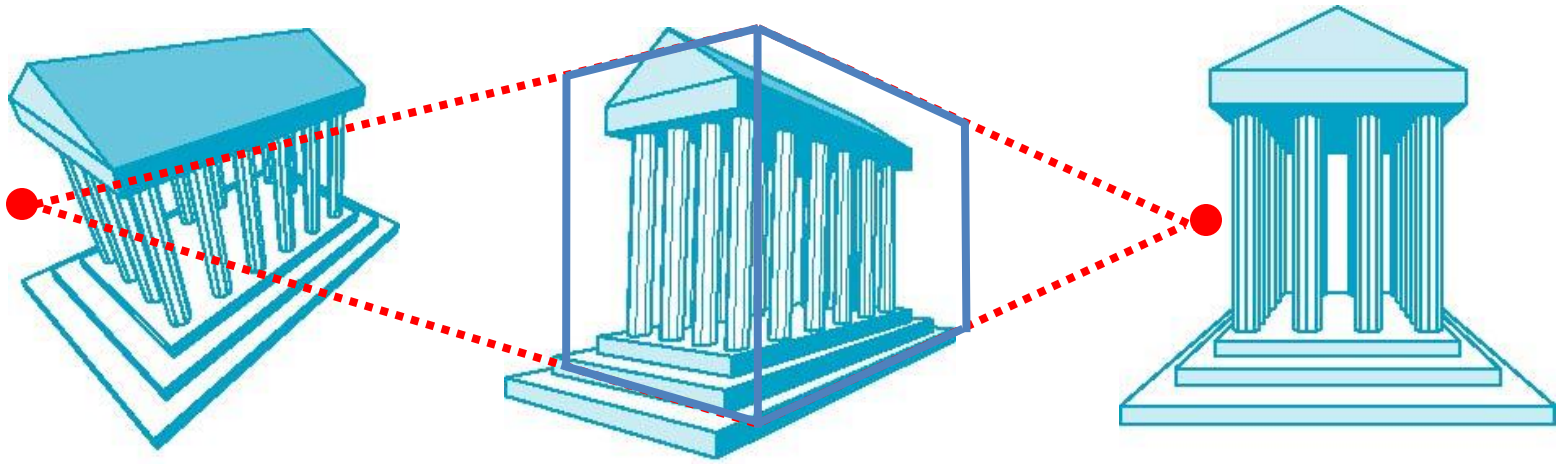
three-point perspective

two-point perspective

one-point perspective

투시 투영의 종류

- **One-, two-, and three-point perspectives**
  - 소실점(vanishing point)의 개수
  - 3개 주요 방향(x, y, z축) 중 몇 개가 투영면과 평행인가?



three-point perspective

two-point perspective

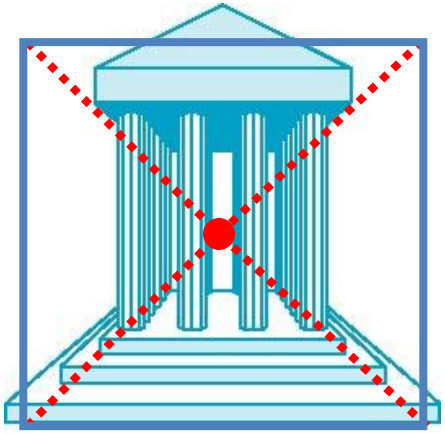
one-point perspective

투시 투영의 종류

- **One-, two-, and three-point perspectives**
  - 소실점(vanishing point)의 개수
  - 3개 주요 방향(x, y, z축) 중 몇 개가 투영면과 평행인가?

three-point perspective

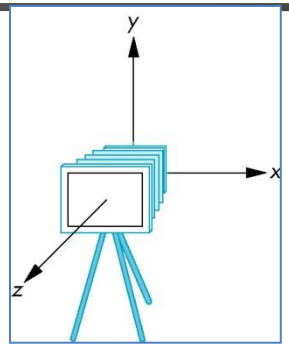
two-point perspective



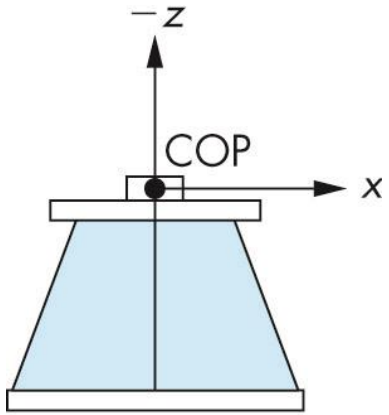
one-point perspective

지난 시간 이어서...

투시 투영의 원리 (1)

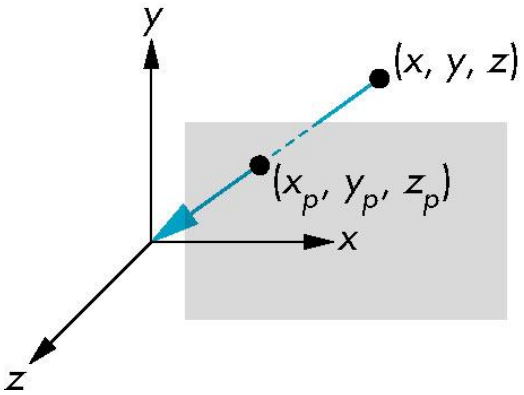


pp.247참조

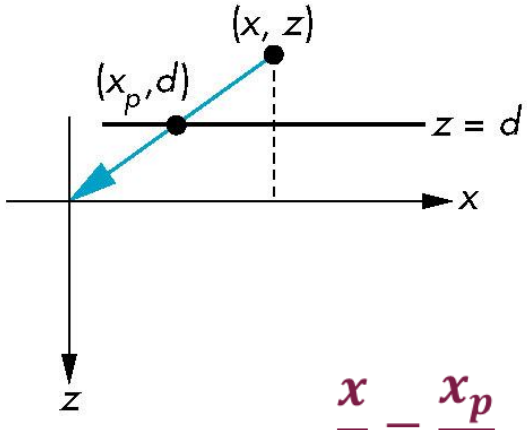


- 가정) 간단한 카메라
  - 투영면은 z축에 수직
  - COP 앞에 투영면 위치

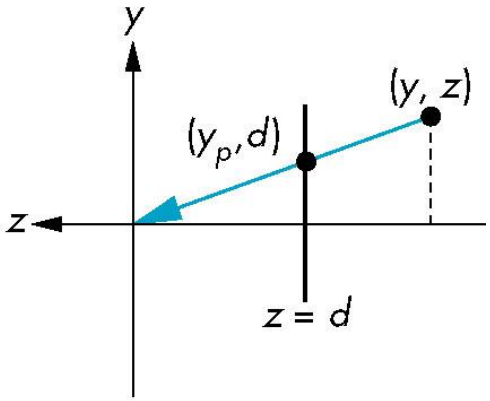
$$x_p = \frac{x}{z/d}, y_p = \frac{y}{z/d}, z_p = d$$



투시 투영



위 $\frac{x}{z} = \frac{x_p}{d}$



옆 $\frac{y}{z} = \frac{y_p}{d}$

투시 투영의 원리 (2)

pp.248참조

- 동차 좌표계 (Homogeneous coordinates)

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ \frac{z}{z/d} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

- 투시 투영 행렬

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

투시투영 파이프라인

직교 투영의 원리

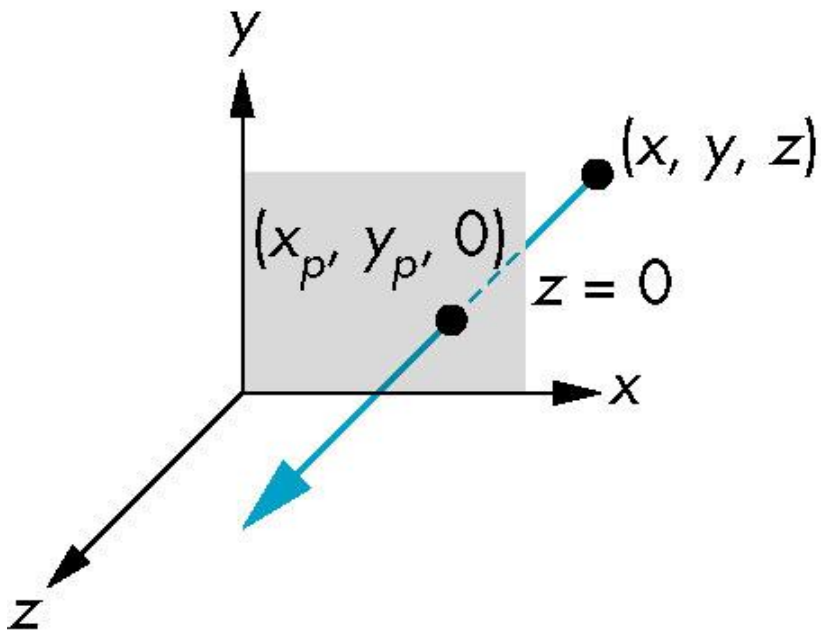
pp.235참조

- 투영 방향은 투영면에 수직

$$x_p = x$$
$$y_p = y$$
$$z_p = 0$$

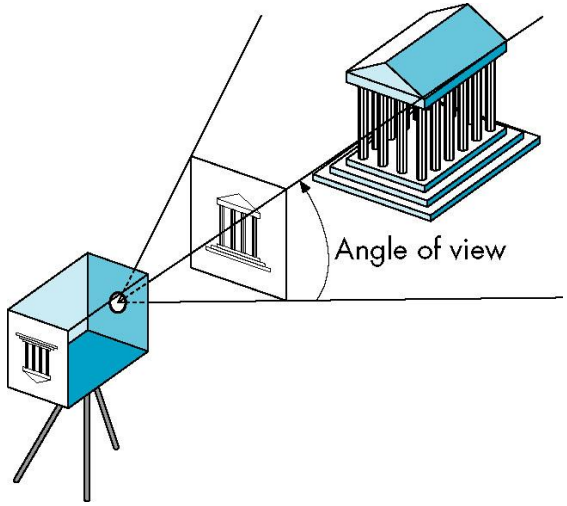
- 직교 투영 행렬

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

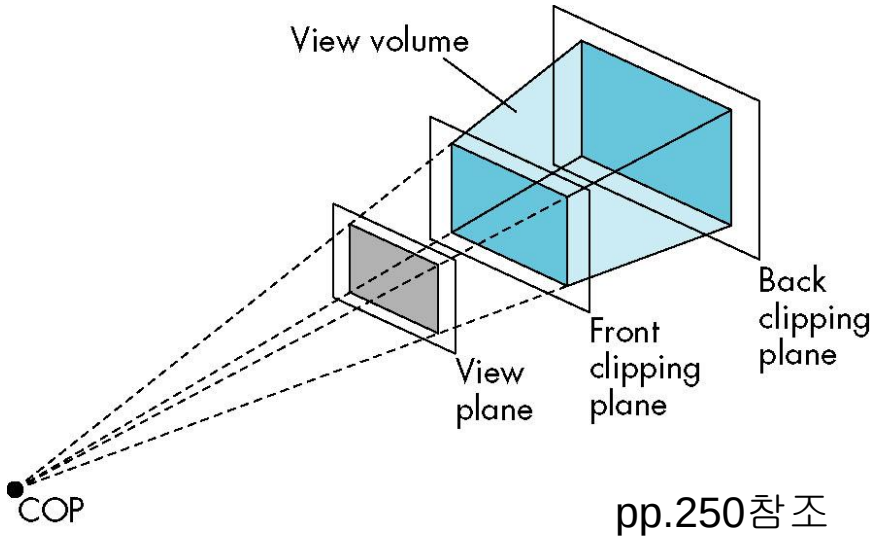


OpenGL의 투영

- 시야각 (Angle of view)
  - 카메라의 시야각 안에 들어오는 물체만 이미지로 나타남



- 뷰 볼륨 (View volume)
  - 장면으로 잘라내는 공간
  - frustum - 잘려진 피라미드



pp.250참조

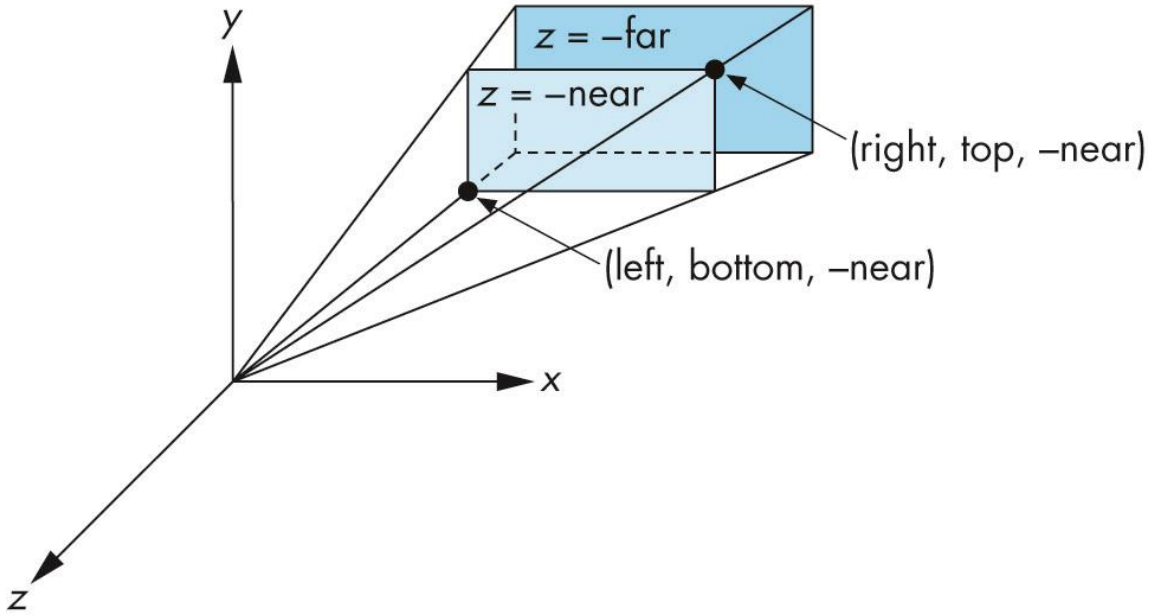
OpenGL의 원근 투영 (1)

pp.251참조

- **Frustum**의 크기를 지정하는 방법

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity( );
glFrustum(left, right, bottom, top, near, far);
```

  - near, far: 반드시 양수 !!
    - ➔ z_{max} = - far
    - ➔ z_{min} = - near

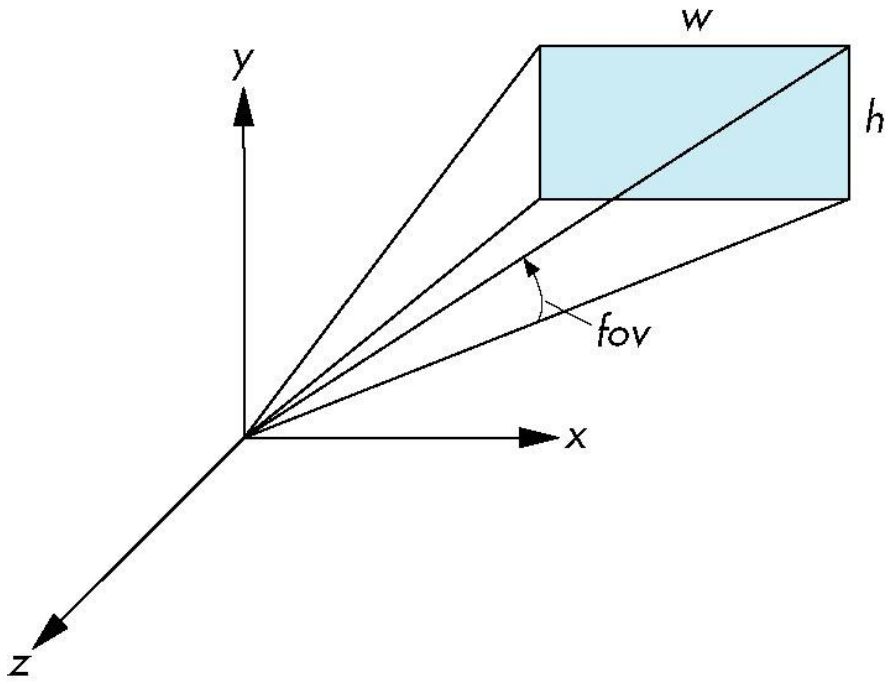


OpenGL의 원근 투영 (2)

- 시야 범위를 지정하는 방법

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity( );
gluPerspective(fovy, aspect, near, far);
```

  - fovy: y축 방향으로 시야각
    (top과 bottom 사이각)
  - 종횡비(aspect ratio):
    aspect = width / height

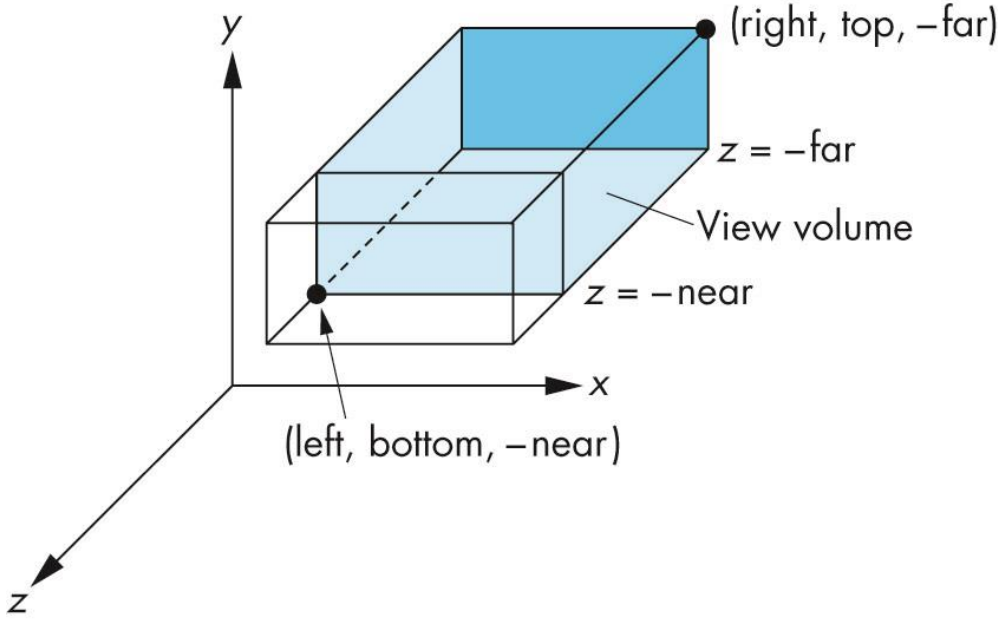


OpenGL의 평행 투영

- 직교 투영 함수

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity( );
glOrtho(left, right, bottom, top, near, far);
```

  - OpenGL은 한 종류의 평행 투영 함수만 제공
  - near < far !!
    - ➔ 양수, 음수 모두 가능
    - ➔ z_{max} = - far
    - ➔ z_{min} = - near

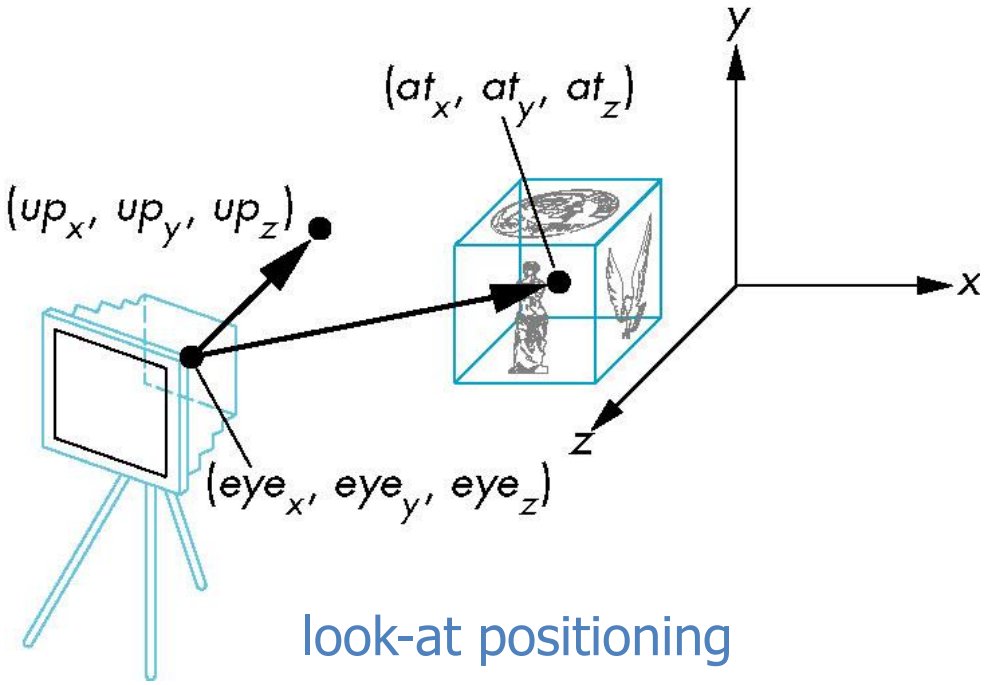


카메라 : Look-At 함수

- OpenGL utility 함수

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

  - eye-position, target-position, and up-vector



look-at positioning

pp.232참조

3차원 공간에서의 카메라

- 시점은 어디를 향하고 있나?

영화 그래비티

3차원 공간에서의 카메라

영화 그래비티

3차원 공간에서의 카메라

- 사진의 우주인은 앞쪽을 보고 있을까요?
- 사진의 우주인은 뒤집혀 있을까요?

영화 그래비티

투영과 그림자 (1)

- 3D 게임에서의 그림자

두 영상의 차이점은 무엇인가?

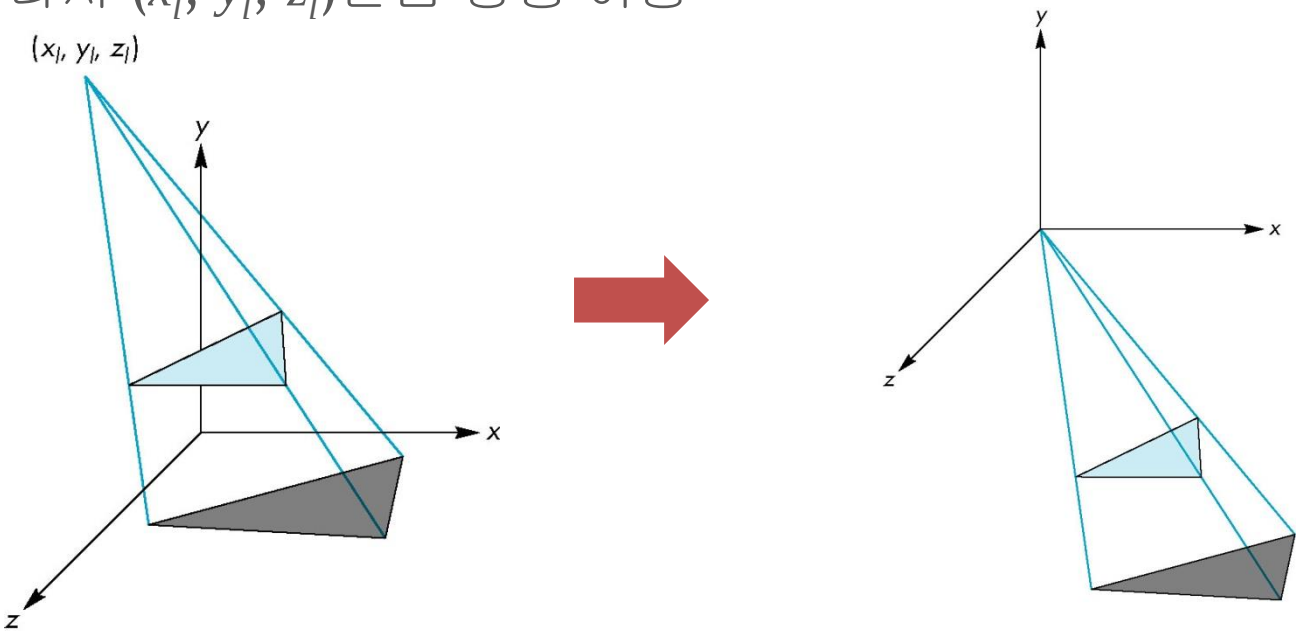
투영과 그림자(2)

Unreal Engine 5

투영과 그림자(3)

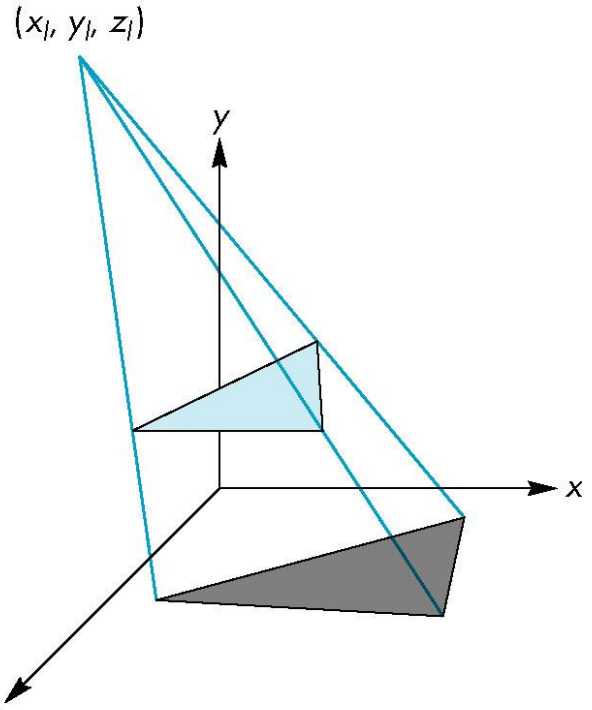
- 그림자 폴리곤
- 과정
  - 광원을 (x_l, y_l, z_l)에 위치
  - $(-x_l, -y_l, -z_l)$만큼 평행 이동
  - 원점 중심으로 투시 투영
  - 다시 (x_l, y_l, z_l)만큼 평행 이동

pp.271참조



투영과 그림자(4)

- 그림자 폴리곤
- 과정
  - 광원을 (x_l, y_l, z_l)에 위치
  - $(-x_l, -y_l, -z_l)$만큼 평행 이동
  - 원점 중심으로 투시 투영
  - 다시 (x_l, y_l, z_l)만큼 평행 이동



$$\mathbf{M} = \mathbf{T}^{-1}\mathbf{P}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & x_l \\ 0 & 1 & 0 & y_l \\ 0 & 0 & 1 & z_l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1/y_l & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_l \\ 0 & 1 & 0 & -y_l \\ 0 & 0 & 1 & -z_l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

투영과 그림자(5)

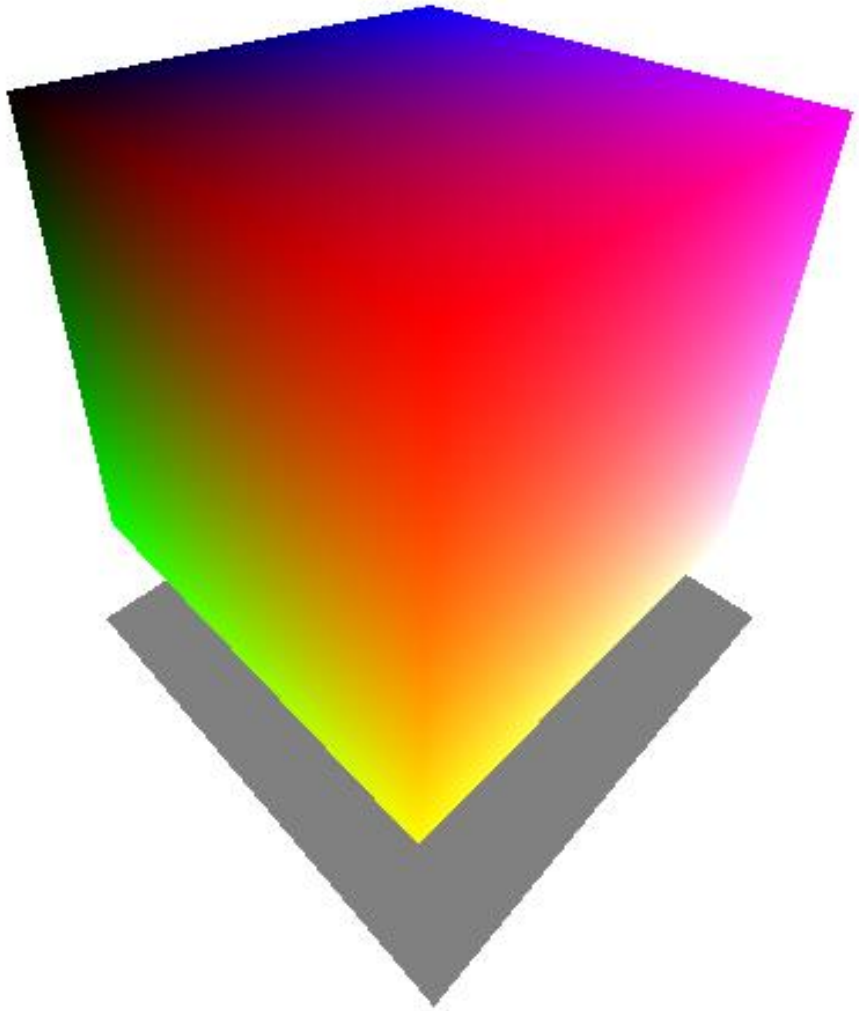
```
GLfloat m[16];                    /* shadow projection matrix */
for(i=0; i<16; i++) m[i] = 0.0;
m[0] = m[5] = m[10] = 1.0;
m[7] = -1.0/yl;

glColor3fv(polygon_color);
glBegin(GL_POLYGON);
.
.                                 /* draw the polygon normally */
.
glEnd( );

glPushMatrix( );                  /* save state */
glTranslatef(xl, yl, zl);         /* translate back */
glMultMatrixf(m);                 /* project */
glTranslatef(-xl, -yl, -zl);      /* move light to origin */

glColorfv(shadow_color);
glBegin(GL_POLYGON);
.
.                                 /* draw the polygon again */
.
glEnd( );
glPopMatrix( );                   /* restore state */
```

Shadows from a Cube onto Ground
큐브에서 지면으로 그림자 드리우기



중간고사 일정

- **일정: 4월 20일 (목요일) 오후 1시 10분**
- **장소: D411 (공학 4호관)**

- **시험 범위**
  - 하영드리미 게시판 확인 할 것