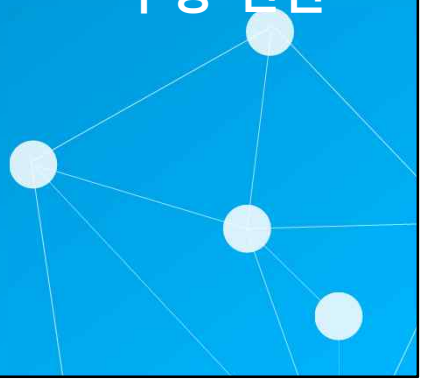
07 CHAPTER

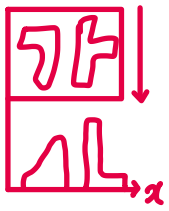
투상 변환



7장. 학습 내용

- 투상 변환의 종류
- 가시부피와 정규화 가시부피
- OpenGL의 평행 투상
- OpenGL의 원근 투상
- 뷰포트 변환 (Viewport Transformation)

7.1 투상 변환의 종류



- 모델좌표계에서 시작한 3차원 물체의 정점
- 전역좌표계
- 시점 좌표계
- 투상면이라고 불리는 2차원 평면으로 대응
- 투상(projection)
  - 이와 같이 3차원 공간상의 물체를 2차원인 화면으로 대응시키기 위한 작업이다.

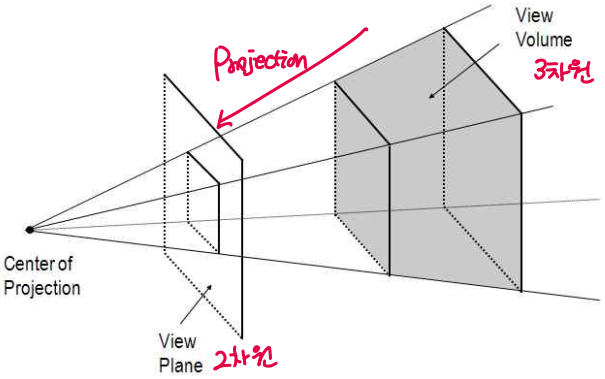
3

용어들

- 시점(Center of projection)
  - 투상 중심으로 카메라 또는 눈의 위치
- 시선(Line of sight)
  - 카메라가 바라보는 방향
- 투상선(projectors)
  - 시점으로부터 물체 곳곳을 향하는 선들
- 투상면(Projection Plane)
  - 물체의 상의 맺히는 면
  - 투상은 3차원 공간을 2차원으로 대응시키는 과정
  - 투상선이 투상면에 만나는 것이 투상
- 가시 부피(View Volume)
  - 전체 3차원 공간에서 최종적으로 화면에 출력하고자 하는 영역
  - 가시 부피의 형태는 투상 방법에 따라 다양

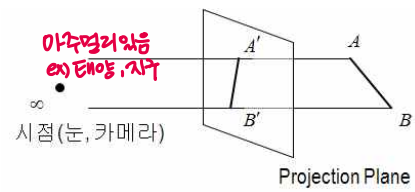
4

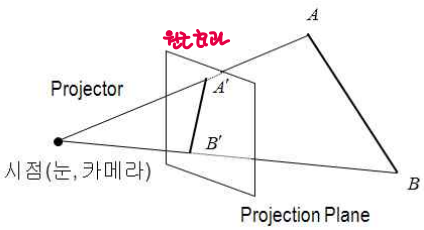
Projection의 개념



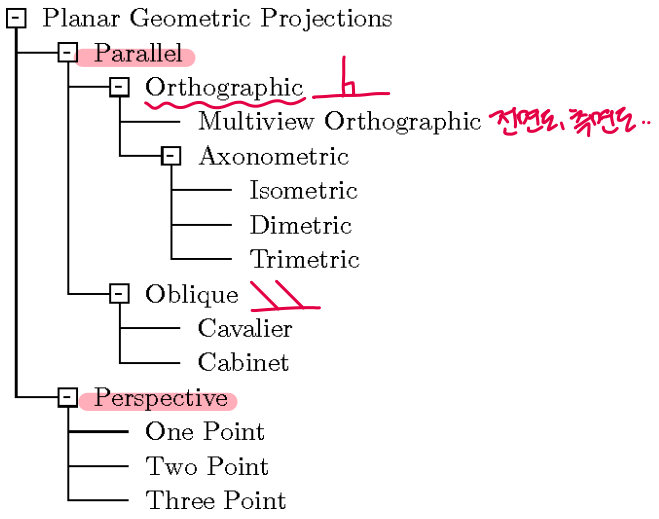
평행 투상과 원근 투상

- 평행 투상(parallel projection)
  - 시점이 물체에서 무한히 멀리 떨어진 경우
- 원근 투상(perspective projection)
  - 시점이 물체에 비교적 가깝게 있는 경우





Planar Geometric Projection의 종류



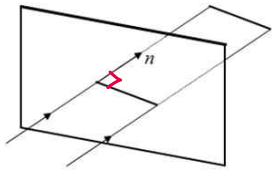
7

평행 투상

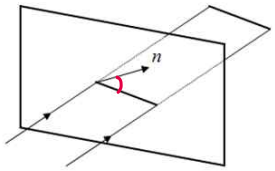
- 시점이 물체에서 무한히 멀리 떨어진 경우
- 실제로는 물체의 크기(깊이)에 비해 카메라의 위치가 매우 먼 경우
  - 원근감이 거의 없어지고 평행 투상과 같은 결과
- 정사투상(Orthographic Projection)
  - 투상선과 투상면이 직교함
  - 일반적인 경우
- 경사투상(Oblique Projection)
  - 투상선은 투상면과 직교하지 않게 처리되는 경우

8

정사 투상과 경사 투상



Orthographic Projection



Oblique Projection

9

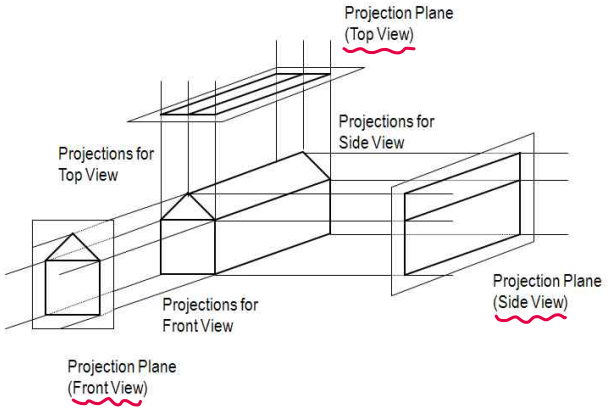
주 면에 의한 정사 투상

- 물체의 **주 면(Principal Plane)**을 설정
  - 건축물의 경우 주면을 이용해 건물을 전면도(입면도)와 평면도, 측면도 등으로 구분해 사용
  - 건물이나 인공물에서만 의미가 있음.
  - 산이나 바위 같은 자연물에서는 주면의 정의가 모호
- 주 면의 개념을 적용하면
  - 다중뷰 정사투상
  - 축측 투상

10

(1) 다중 뷰 정사 투상

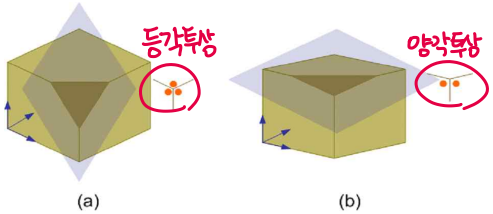
- 투상선이 주면에 직교하게 투상
  - x-z, x-y 및 z-x 평면과 평행한 투상면으로 직교투상
  - 전면도(Front View), 측면도(Side View), 평면도(Top View)



11

(2) 축측 투상(Axonometric Projection)

- 한꺼번에 여러 면의 정보를 보여주고자 함
  - 투상선이 투상면과 직교함: 평행 투상
  - 그러나 투상선이 주면들과는 더 이상 직교하지 않음
- 각 축에 대한 축소율에 따른 분류
  - 삼각 투상(Trimetric Projection) → 모든 각도가 자유롭게
  - 양각 투상(Dimetric Projection)
  - 등각 투상(Isometric Projection)



12

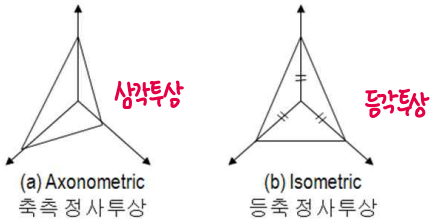


그림 7.6 축측 투상(Axonometric Projection)과 축소비율

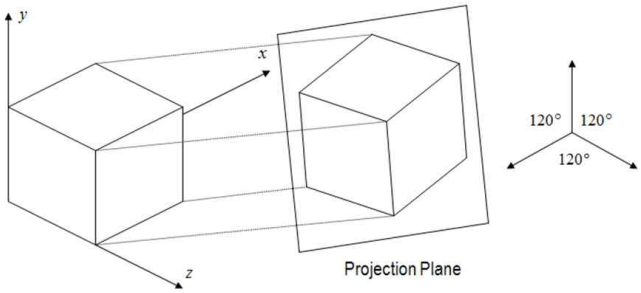
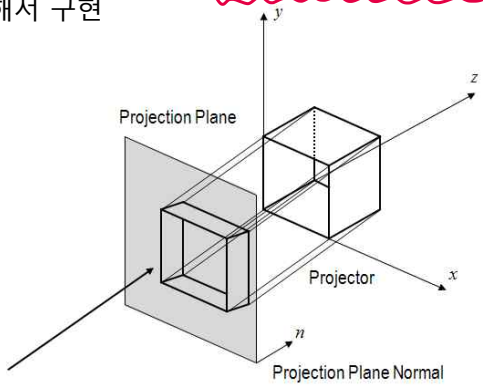


그림 7.7 등각 투상(Isometric Projection)

13

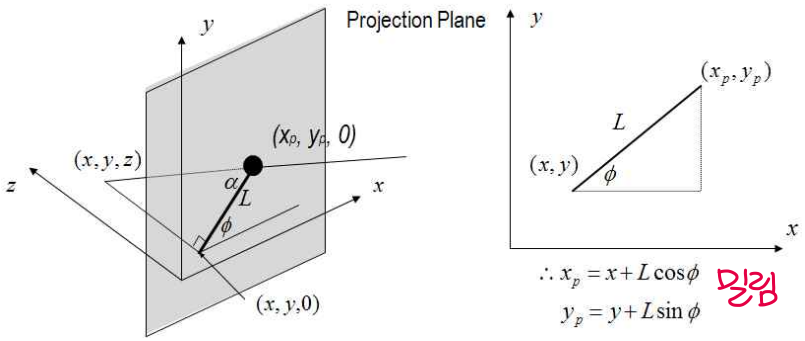
경사 투상

- 투상선이 방향이 투상면과 직교하지 않는 경우
  - 정점의 x나 y 좌표 값이 z값에 따라 밀리는 형태로, 밀림 연산을 이용해서 구현



14

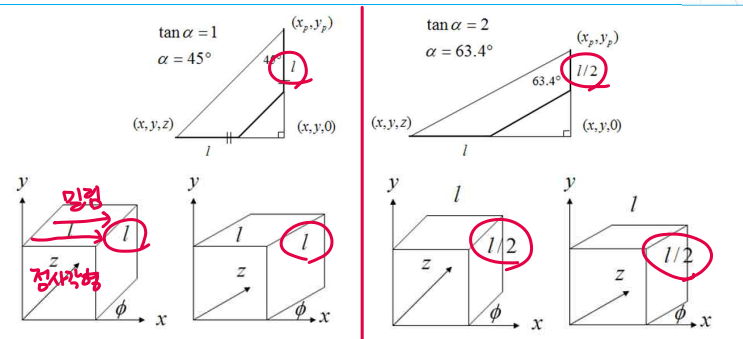
- 경사 투상(Oblique Projection)을 위한 밀림 연산

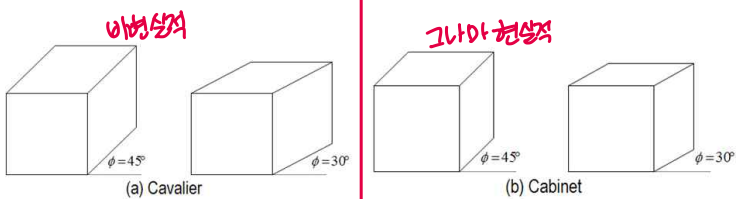


15

그냥 이런게 있다고만 생각하면됨

Cavalier Projection과 Cabinet Projection





16

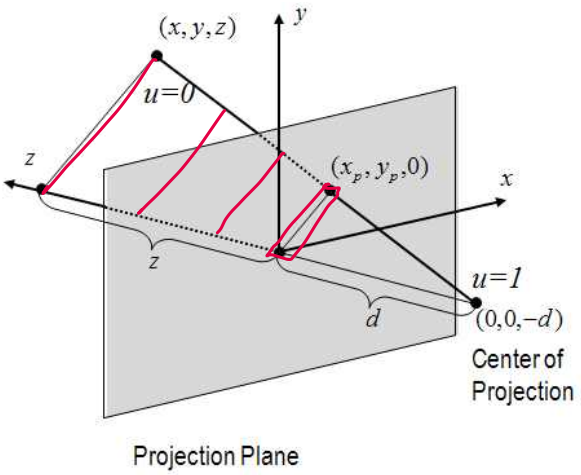
- [Lab 7-1] 추사 김정희 선생의 세한도(歲寒圖)를 찾아보고, 그림 안에 있는 집이 어떤 투상법으로 그렸는지 생각해 보라.

집을 정면으로 보았는데 옆면이 보이기 때문 ⇒ 경사투상

원근 투상(Perspective Projection)

- 물체의 크기에 비해 시점이 비교적 가까운 거리
- 투상선이 시점에서 출발하여 방사선 모양으로 퍼져나감 ⇒
- 원근 투상법을 사용하면 거리에 따라 물체의 크기가 달라 훨씬 실감나는 화면
- 그러나 최종적으로 투상면에 출력된 물체의 크기를 보고 실제 물체의 크기를 유추할 수 없다는 단점
  - 공학도면과 같이 그 내용을 바탕으로 직접 제작을 하는 용도로 사용할 수 없음.

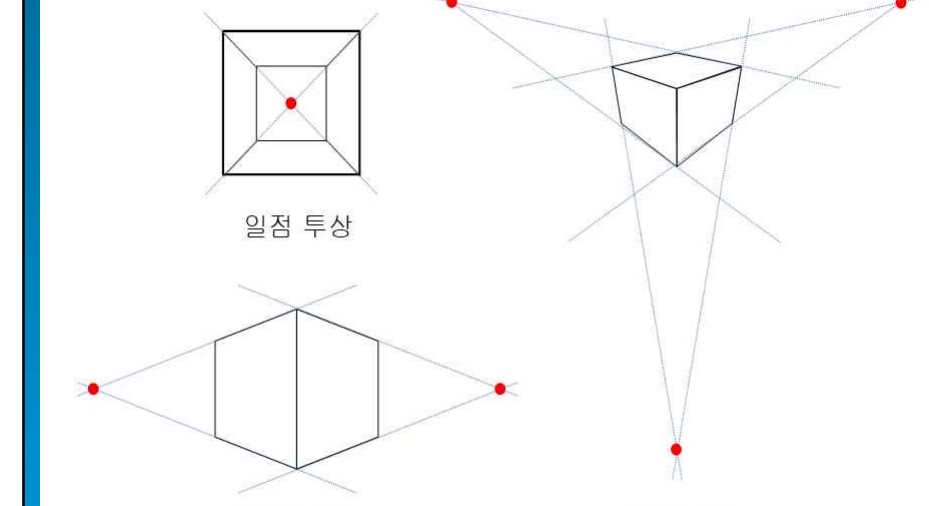
원근 투상의 개념



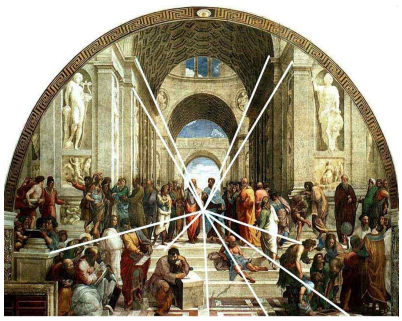
원근투상 소실점에 따른 분류

- 소실점(Vanishing point)
  - 원근 투상의 결과로 평행선이 만나는 점

- 소실점(Vanishing point)에 따른 분류
  - 일점 투상(One-point Projection)
  - 이점 투상(Two-point Projection)
  - 삼점 투상(Three-point Projection)



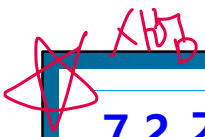
21



22

- [Lab 7-2] Tour Into the Picture이란 키워드를 인터넷에서 검색해보라. TIP 알고리즘과 소실점의 관계를 찾아보라.
- [Lab 7-3] 차량 전면에 장착된 카메라의 영상을 이용해 차선을 추출하고 앞차와의 거리를 계산하기 위해 소실점이 어떻게 사용되는지 알아보라.

23



7.2 가시부피와 정규화 가시부피

- 전후방 절단면과 가시부피
- 각 투상에서의 가시부피의 형태
- 정규화 가시부피의 개념

24

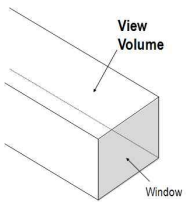
가시부피(View Volume)

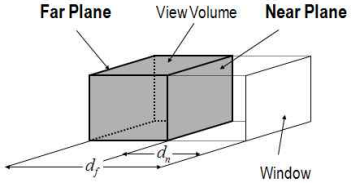
- 가상의 3차원 공간에 많은 로봇들이 흩어져 싸우고 있는 게임을 생각해 보자. 3차원 공간은 넓고 화면은 제한적이므로 화면에 모든 공간을 나타낼 수는 없을 것이고, 매 순간 화면에는 전체 공간의 일부만 보일 것이다.
- 전체 가상 공간에서 투상면에 투상하고자 하는 3차원 공간상의 일정 영역을 **가시부피(View Volume)**라 한다.
- 가시부피는 투상 방법에 따라 각기 다른 형태
- OpenGL의 가시부피 설정 함수
  - glOrtho()
  - glFrustum()
  - gluPerspective()

25

전방 절단면과 후방 절단면

- **전방 절단면(near plane)**
  - 시점에서 가까운 절단면
  - 이 면 앞쪽은 가시부피에서 제외됨
  - Near Clipping Plane, Near Plane, Front Plane, Hither 라고도 함.
- **후방 절단면(far plane):**
  - 시점에서 멀리 떨어진 절단면
  - 이 면의 뒷쪽은 가시부피에서 제외됨
  - Far Clipping Plane, Far Plane, Back Plane, Yon 이라고도 함.

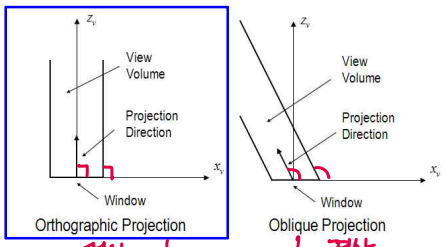




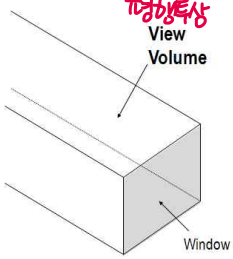
26

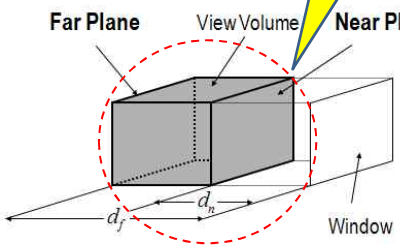
시험에 그림그리고 설명하기

정사투상에서의 가시부피



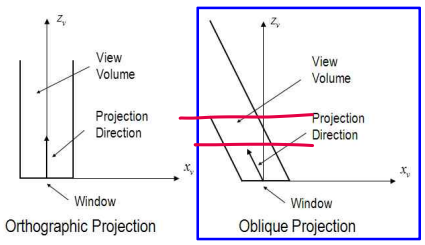
정사 평행투상 경사

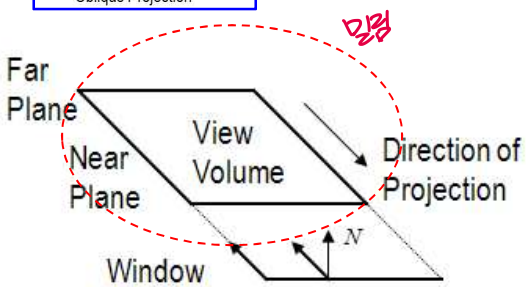




27

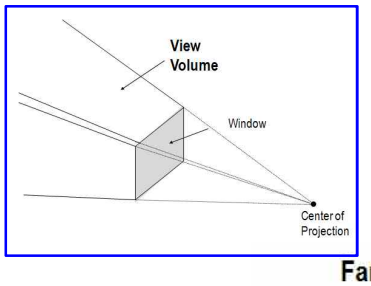
경사투상에서의 가시부피

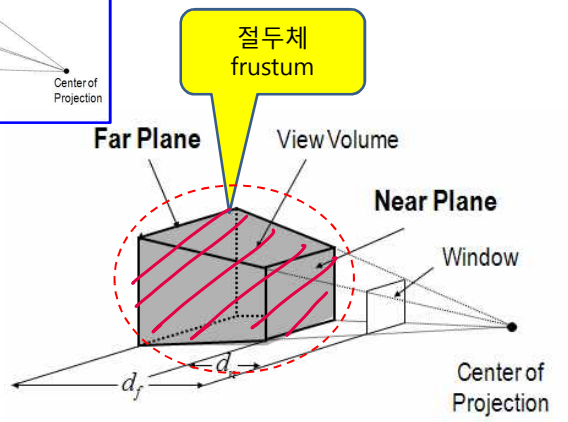




28

원근투상에서의 가시부피

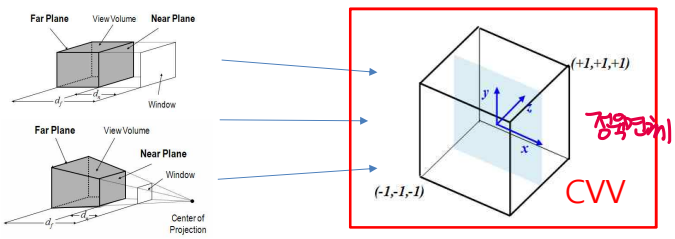




정규화(Canonical) 가시부피(CVV)

- 가로, 세로, 높이가 2인 정육면체의 가시 부피
- 정규화 변환(Normalization Transformation)
- 사용 이유
  - 평행투상, 원근투상을 동일한 모습의 정규화 가시부피로 변형
    - 동일 파이프라인 사용

장점
- 정육면체를 기준으로 하면 연산이 간단함.
- 다양한 해상도의 화면 좌표계로 변환하기가 간단함.



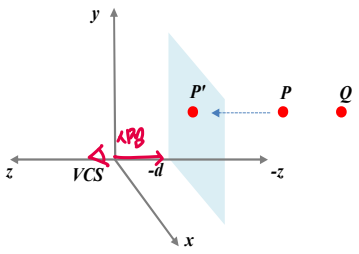
7.3 OpenGL의 평행투상

- 평행 투상의 기본 개념
- 평행 투상 가시부피 설정
- glOrtho()의 역할

31

평행투상 기본 개념

- 모델 좌표 → 전역 좌표 → 시점 좌표 순서로 변환된 상태
  - 현재 모든 점들은 시점 좌표계 VCS를 기준으로 표시
  - 카메라는 -z방향



- 공간상의 두 점 P와 Q의 x와 y좌표가 동일하다면
  - 평행 투상에서 P와 Q는 z값에 상관없이 투상면의 같은 지점 P'로 대응

32

평행투상을 위한 가시 부피 설정

- 행렬 표현

$$P' = M_{ortho} \cdot P$$

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

고대로임

특이변환(Singular Transformation)

- 역변환이 없는 변환 (Projection 후에는 원래 3차원 좌표로 돌아갈수 없다)
- (x, y, z, 1)에서 (x, y)만 읽어내면 그것이 투상된 2차원 좌표

평행투상 가시부피 설정 함수

- OpenGL 함수

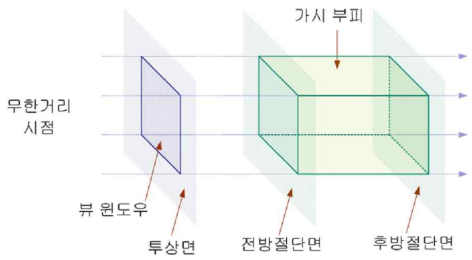
(→ View volume 지정 (네모이만큼만 볼거야!))

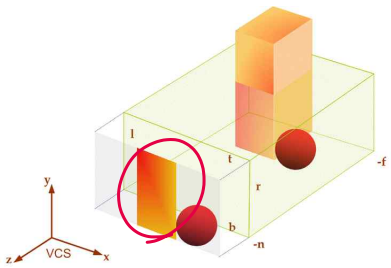
```
void glOrtho ( double left, double right, double bottom, double top,
               double near, double far);
```

- (left, right, bottom, top): 투상면의 좌, 우, 하, 상 지정
- (near, far): 전후방 절단면

- 사용 예

```
glMatrixMode( GL_PROJECTION );      // 현재 행렬을 프로젝션 행렬로
glLoadIdentity( );                  // 프로젝션 행렬을 항등행렬로
glOrtho ( l, r, b, t, n, f );       // 가시부피 지정
```

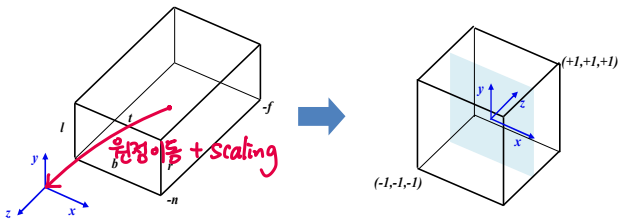




35

glOrtho()의 역할

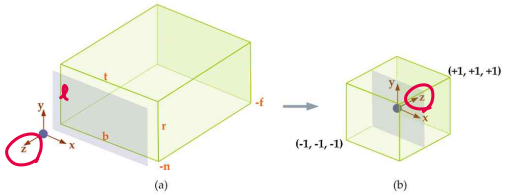
- 설정한 가시 부피를 정규화 가시부피 CVV로 변환하는 변환을 위한 행렬을 만들고
  - 이동, 크기조절 변환
  - 반사 변환: 정규화 가시부피는 왼손 좌표계



- 이것을 현재 Projection 행렬에 곱함
- 변환이 적용되면 → 절단 좌표계(CCS: Clip CS)

36

glOrtho() 행렬 계산



원점이동
$$T = \begin{pmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

reflection
반사
$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$P' = R\ S\ T\ P$$

- [Lab 7-4] OpenGL에서는 경사투상을 위한 함수를 지원하지 않는다. 위에서 설명한 방법을 이용하여 Cavalier 투상과 Cabinet 투상을 위한 변환 함수를 만들어보라.

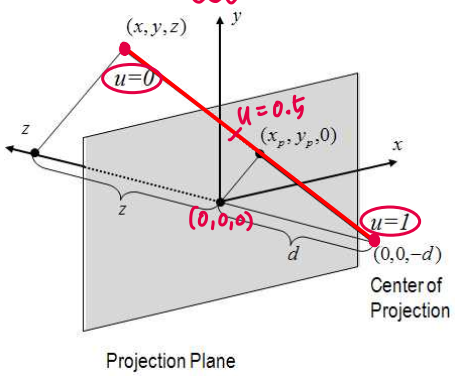
7.4 OpenGL의 원근투상

- 원근 투상의 기본 개념
- 원근 투상 가시부피 설정
- glFrustum()의 역할

원근투상 기본 개념

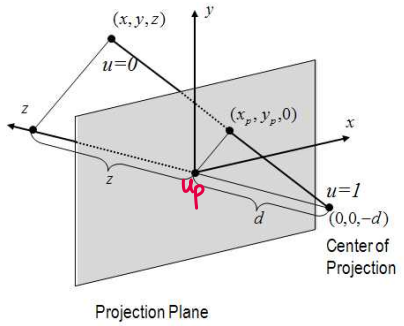
- 매개변수 방정식(parametric equation)으로 시선을 표현

$$x' = (1-u)x + u \cdot 0 = x - xu \qquad 0 \le u \le 1$$
$$y' = (1-u)y + u \cdot 0 = y - yu$$
$$z' = (1-u)z + u(-d) = z - (z+d)u$$



- 투상면에 맺히는 점

$$z_p = 0 \Rightarrow z-(z+d)u_p = 0$$
$$\therefore u_p = \frac{z}{z+d}$$



$$x_p = (1-u_p)x + u \cdot 0 = (1-\frac{z}{z+d})x = (\frac{d}{z+d})x = \boxed{(\frac{1}{z/d+1})x}$$
$$y_p = (1-u_p)y + u \cdot 0 = (1-\frac{z}{z+d})y = \boxed{(\frac{1}{z/d+1})y}$$
$$z_p = 0$$

41

원근투상 변환 행렬

원근투상변환행렬

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{1/d} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ \boxed{z/d+1} \end{bmatrix}$$

동차좌표

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z/d+1} \\ \frac{y}{z/d+1} \\ 0 \\ 1 \end{bmatrix}$$

3차원 공간상의 점

- **원근분할(Perspective Division)** 과정이 필요
  - 어파인 변환이 아님. 3차원 좌표 관점에서는 비선형 변환
  - OpenGL에서는 절단이 동차좌표에서 이루어지기 때문에 이 과정이 절단 이후로 미루어 짐

왜?? 마지막행렬 =[0 0 0 1]

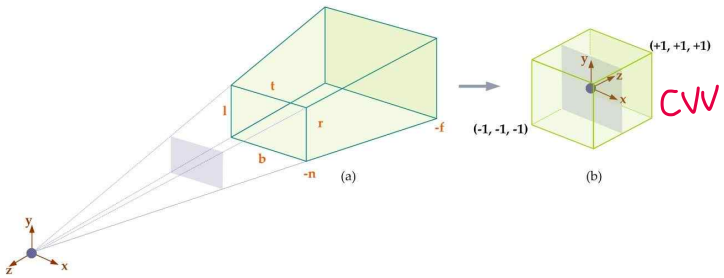
42

원근투상 가시부피 설정 함수

- OpenGL 함수

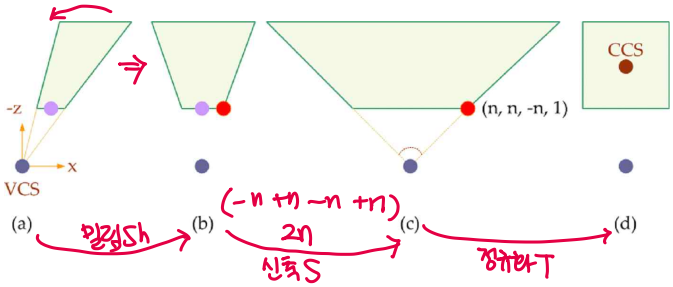
```
void glFrustum ( double left, double right, double bottom, double top,
                 double near, double far);
```

  - (left, right, bottom, top): 투상면의 좌, 우, 하, 상 지정
  - (near, far): 전후방 절단면

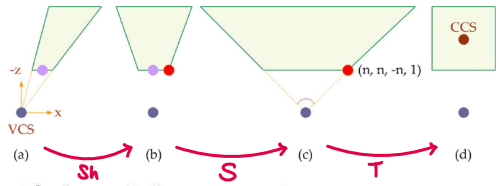


glFrustum() 역할

- 설정된 가시부피를 CVV로 변환하는 과정
  - 밀림 변환 Sh
    - 절두체의 중심이 z축과 일치하도록 변환
  - 신축 변환 S
    - 전방 절단면의 가로와 세로 크기가 모두 2n이 되도록 변환.
  - 정규화 변환 T
    - 가시부피를 최종적으로 정육면체로 변환



glFrustum() 행렬 변환

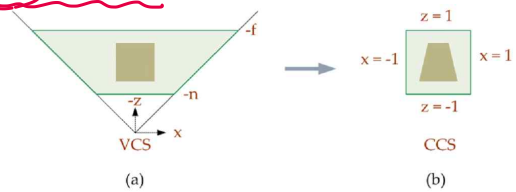


$$Sh = \begin{pmatrix} 1 & 0 & \frac{r+l}{2n} & 0 \\ 0 & 1 & \frac{t+b}{2n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, S = \begin{pmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$N = T \cdot S \cdot Sh = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

원근투상의 정규화 가시부피

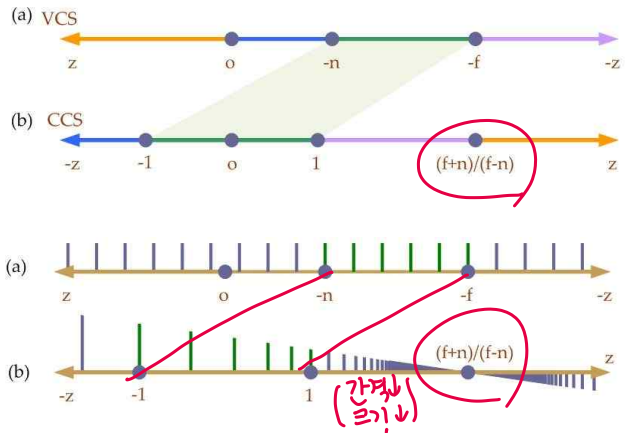
- 전방 절단면에 비해 후방 절단면이 줄어듬.
  - 멀리 있는 것이 작게 보여야 함.



- T 행렬
  - Z 값에 영향을 미침: 원래의 물체 정점의 깊이 z와 정규화 변환 후의 물체 정점의 깊이 z'의 관계

$$P' = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

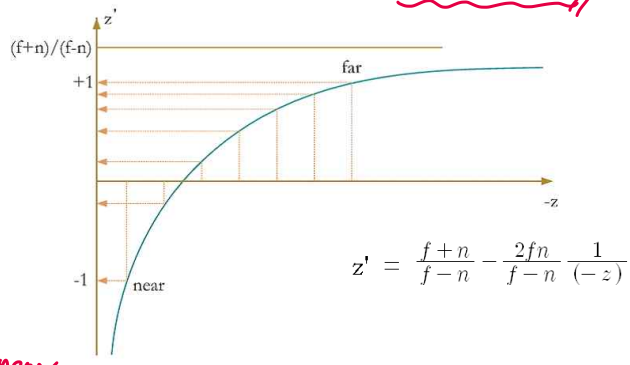
VCS → CCS



- 높이: 멀어질수록 전봇대 높이가 낮아짐(원근 변환)

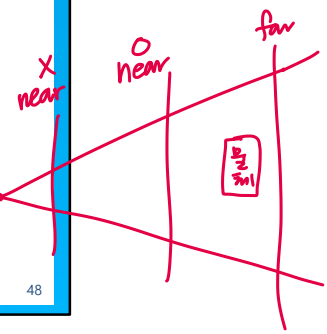
VCS → CCS

- 간격:더욱 촘촘해 짐에 유의(비선형 변환)



$$z' = \frac{f+n}{f-n} - \frac{2fn}{f-n}\frac{1}{(-z)}$$

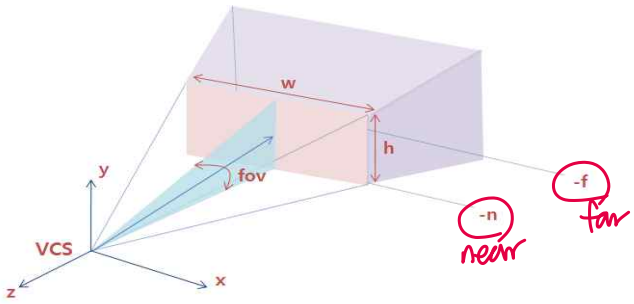
- 전방 절단면: 시점에서 멀리, 물체에 최대한 근접 설정
- 물체 간격이 상대적으로 보존, 지-버퍼 처리에 유리



대칭적 원근투상 함수

- GLU 함수

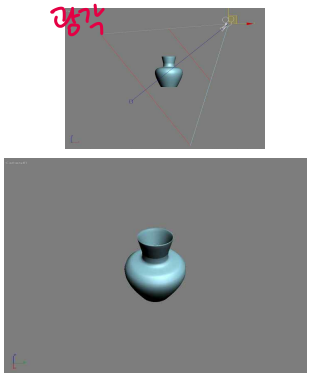
```
void gluPerspective ( double fov,
                      double aspect,
                      double near, double far );
```



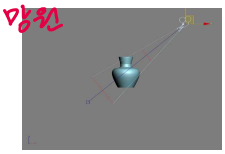
시야각과 카메라 렌즈

- 초점 거리 50mm 기준
  - 광각렌즈(Wide Angle Lens) : 50보다 작음
  - 망원렌즈(Telescope Lens) : 50보다 큼

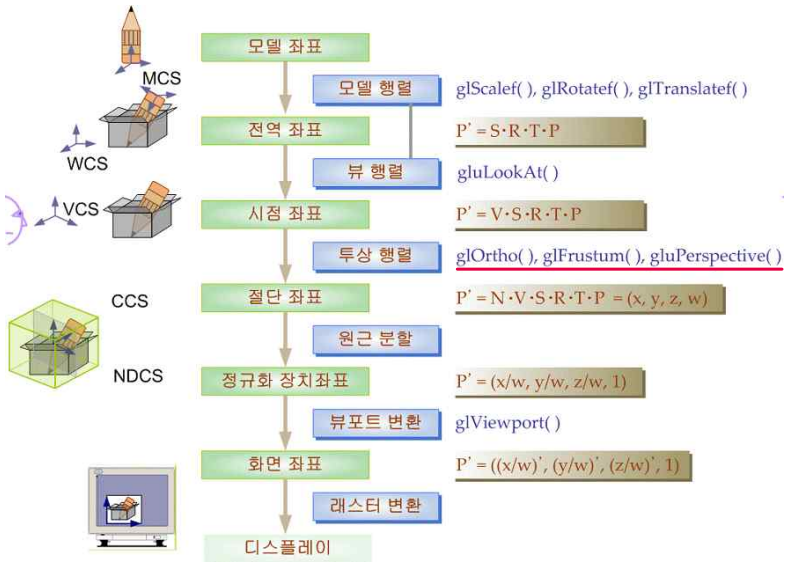
20mm = 85도 시야각



85mm = 24도 시야각



OpenGL의 파이프라인



51

전체 투상 과정

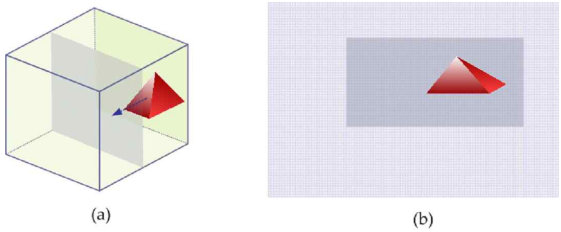
- 투상 변환에 의해 일단 정점들이 **절단 좌표계(CCS)**로 변환
  - 정점들은 동차좌표로 표시되고 절단(clipping) 알고리즘들은 동차좌표 공간에서 이루어 짐.
  - 절단 알고리즘은 다음 장에서.
- 절단작업이 끝나면 **원근분할**
  - 모든 정점들을 실제 3차원 좌표로 변환
  - 동차 좌표 (x, y, z, w) → 3차원 좌표 (x/w, y/w, z/w, 1)로 변환
  - 결과 → **정규화 장치 좌표계(NCDS)**
- 마지막으로 뷰포트로 투상
  - 정규화 가시부피의 z=0 평면 → 뷰포트
  - 3차원의 NCDS에서 먼저 깊이 정보를 활용하는 처리
  - 이후 z성분을 없애면 → 2차원의 좌표 (x/w, y/w)
  - (x/w, y/w)는 다시 뷰포트의 크기에 따라 **장치좌표계(DCS)**로 변경

52

7.5 뷰포트 (Viewport) 변환

뷰포트 변환

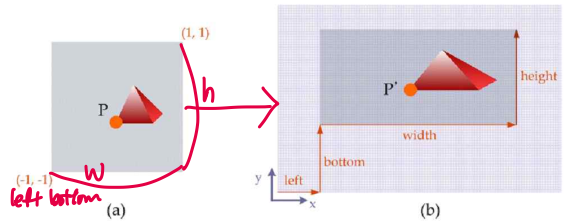
- 정규화 장치좌표계(NDCS)
  - 절단 이후 원근분할에 의해 물체 정점을 3차원 좌표로 변환한 것
  - (x', y', z', 1) = (x/w, y/w, z/w, 1)
- 뷰포트 변환(Viewport Transformation)
  - 정규화 장치좌표계에서 화면 좌표계로 가는 작업



(a) (b)

뷰포트 설정

```
void glViewport(GLint left, GLint bottom, GLsizei width, GLsizei height);
```



$$x' = \frac{x-(-1.0)}{(1.0)-(-1.0)} \times width + left$$

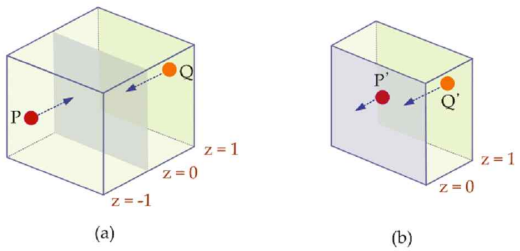
$$y' = \frac{y-(-1.0)}{(1.0)-(-1.0)} \times height + bottom$$

$$P' = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & left + \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & bottom + \frac{height}{2} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

55

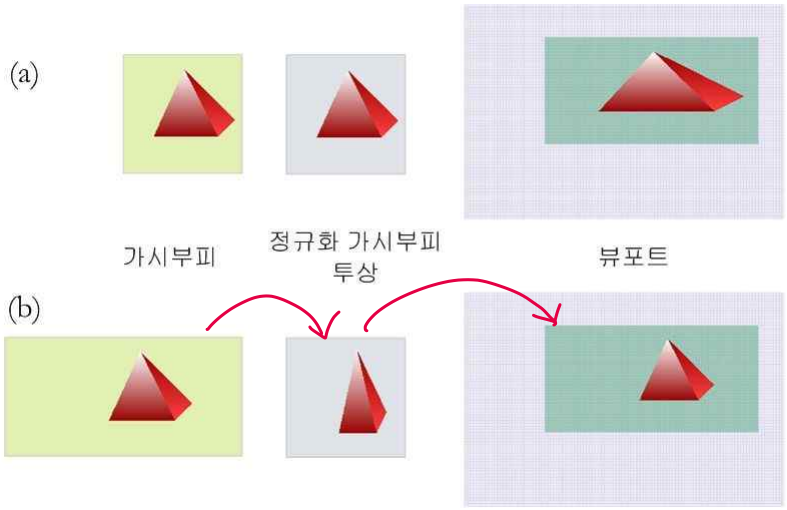
z 값의 재 정규화

- 정규화 가시부피에서의 z 값의 범위인 [-1, +1] 사이를 [0, 1] 사이로 사상
  - 정점 사이의 상대적인 깊이는 유지
  - 정규화 가시부피를 원점을 중심으로 해서 z축 방향으로 1/2 만큼 크기조절 변환을 가한 후, z 방향으로 1/2 만큼 이동



56

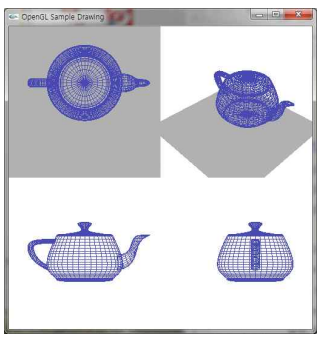
가시부피와 뷰포트



57

7장 연습문제

- [Lab 7-5] 3D Max와 유사하게 화면을 4개의 뷰포트로 만들고 6장에서 구현한 로봇에 대한 각각 정면도/측면도/평면도를 그리고, 나머지 하나에는 그림과 같이 glFrustum()을 이용한 원근투상 결과를 그려라.
  - 정사투상 뷰들에서는 마우스로 로봇을 회전.
  - 원근투상 뷰에서는 로봇 크기를 변경.



58

- [Lab 7-6] 앞의 프로그램에서 원근투상의 전방 절단면과 후방 절단면을 조절할 수 있도록 하는 기능을 구현하라. 마우스 이벤트를 사용해도 좋고 키보드 이벤트를 사용해도 좋다. 전후방 절단면 조절에 따른 원근투상 결과의 차이를 분석해 보라.

59

감사합니다!

60