

Part1: Rendering

2. 가시성 판단

눈에 보이는지 안보이는지.

Outline

- I. Background
- II. 후면 **인보이는** 양쪽.
- III. 절단 **눈에 보이지 않는** 부분 작르기.
- IV. 은면 제거 **보통** 여러개 접설때 **인보이는면** 제거.

1. Background

- Vector
- 평면표현
- 지엘의 법선벡터

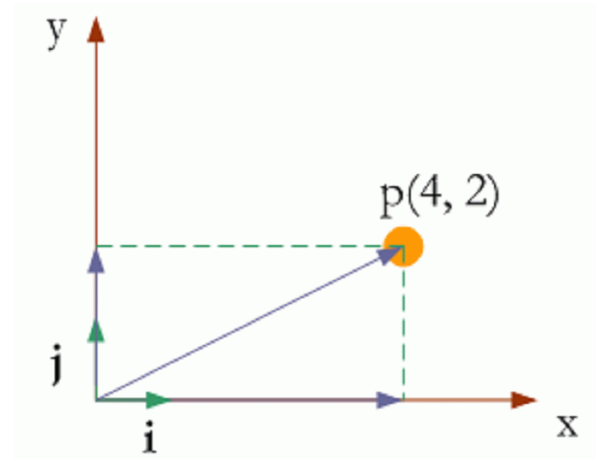
1.1 Vector

- 정규화 벡터(Normalized Vector)

크기를 1로 정규화. 방향은 그대로.
(=1)

$$= \frac{\text{벡터}}{\text{크기}}$$

벡터 크기
방향



$$| \mathbf{p} | = \sqrt{x^2 + y^2 + z^2}$$

$$\mathbf{p}' = \left(\frac{x}{| \mathbf{p} |}, \frac{y}{| \mathbf{p} |}, \frac{z}{| \mathbf{p} |} \right) \text{ 정규화 벡터 }$$

$$= \left(\frac{x}{\sqrt{x^2 + y^2 + z^2}}, \frac{y}{\sqrt{x^2 + y^2 + z^2}}, \frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$$

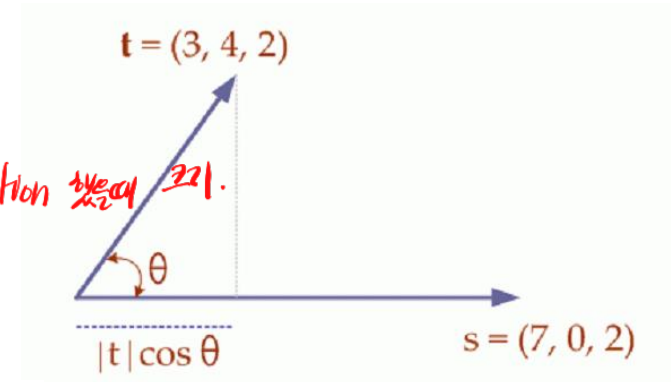
1.1 Vector

- 내적 (Inner Product, Dot Product)

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

→ 스칼라 (상수). 하나의 벡터를 다른 벡터로 projection 했을 때 크기.

$$3 \times 7 + 4 \times 0 + 2 \times 2 = 25.$$



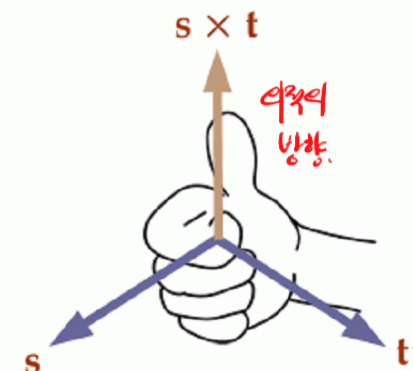
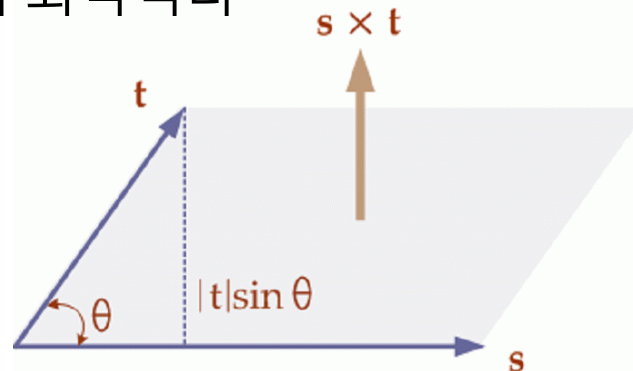
- 외적 (Outer Product, Cross Product)

$$s \times t = |s| |t| \sin \theta \text{ (n 벡터)}$$

$$= (s_y t_z - s_z t_y, -s_x t_z + s_z t_x, s_x t_y - s_y t_x) \text{ 3차원 벡터 이점은 3차원 벡터.}$$

$$s \times t = -t \times s \text{ 교환법칙 성립 X.}$$

- 정규화 법선벡터 = 정규화 외적벡터

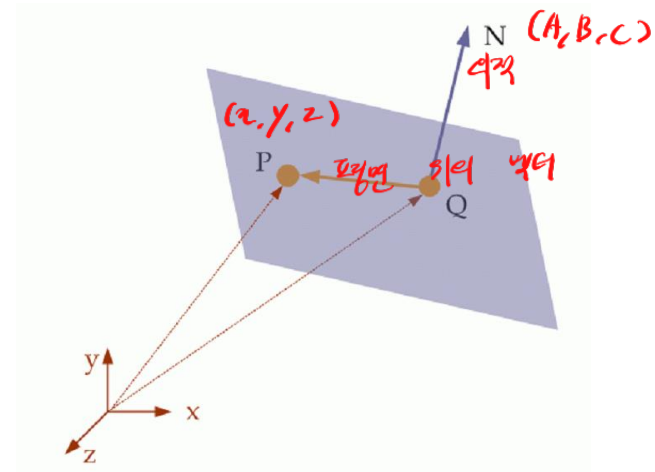


1.2 평면 표현

이로 수직인 벡터끼리의 내적 = 0.

$$(P - Q) \cdot \text{법선벡터 (이것?)} = 0$$

정리. $\hookrightarrow P \cdot N = Q \cdot N$



$(x, y, z) \cdot (A, B, C) = Q \cdot N$

(x, y, z)는 평면 위의 점, N는 법선벡터

$$Ax + By + Cz = Q \cdot N \rightarrow D \text{ (상수)}$$

$$Ax + By + Cz + D = 0$$

법선벡터.

여기에 x, y, z 도입했는데 0 아니다

$\rightarrow x, y, z$ 는 해당 평면 위의 점.

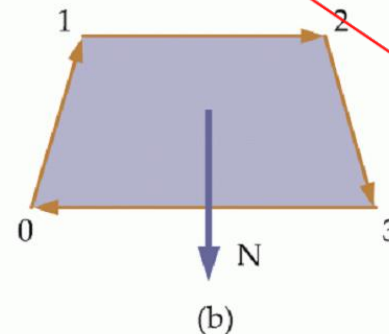
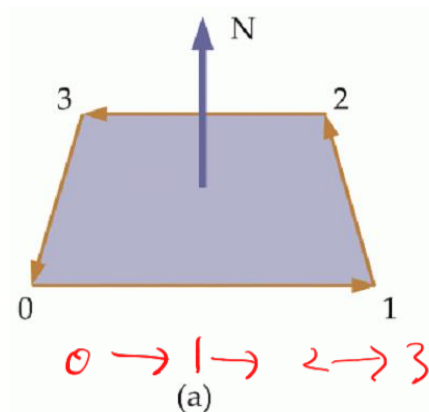
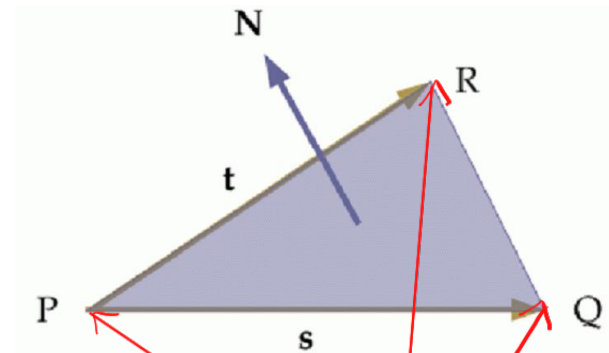
1.3 지엘의 법선벡터

- 법선 벡터 방향: 오른 손을 명시된 정점 순으로 감싸 쥐었을 때 엄지방향

$$\mathbf{s} = (Q_x - P_x, Q_y - P_y, Q_z - P_z)$$

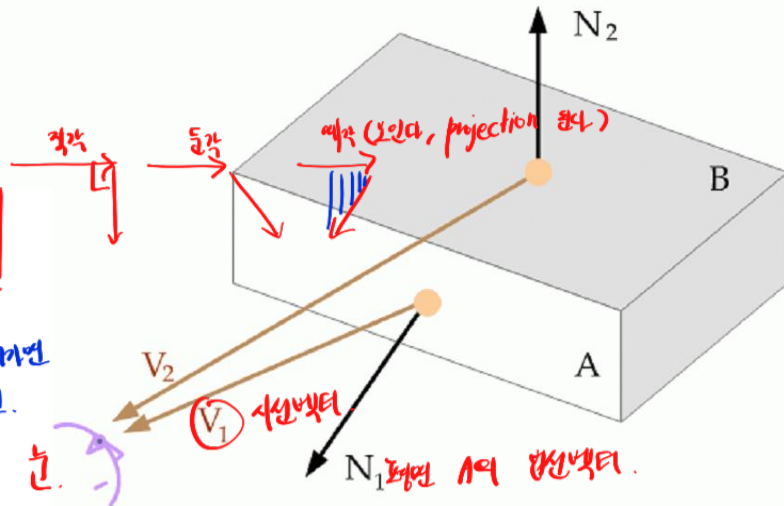
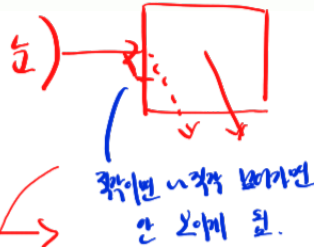
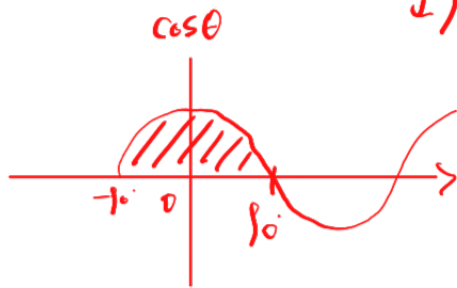
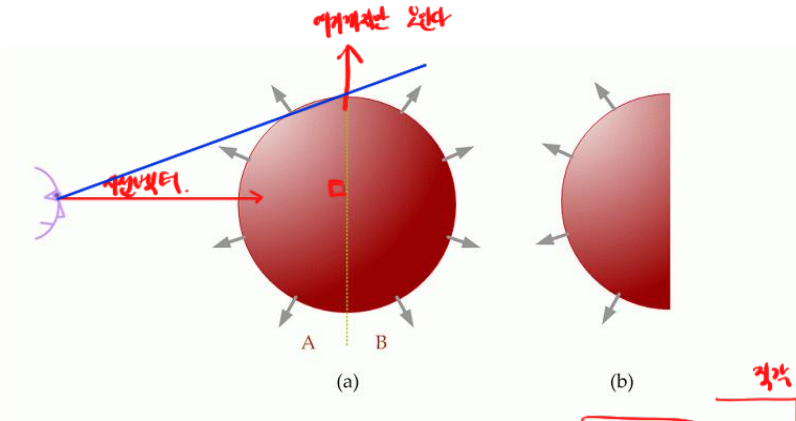
$$\mathbf{t} = (R_x - P_x, R_y - P_y, R_z - P_z)$$

$$\mathbf{N} = \mathbf{s} \times \mathbf{t}$$



2. 후면

- 전면(Front-facing polygon)과 후면(Back-facing polygon)
 - 눈에 보이는 면
 - 눈에 보이지 않는 면
- 후면 제거 (Backface Culling, Backface Removal)
 - 애플 보이게 하면
사실상 비. 머브라빙 비 → 제거해야.
 - 카메라 위치(시점)에 따라
시선벡터 정의 필요.



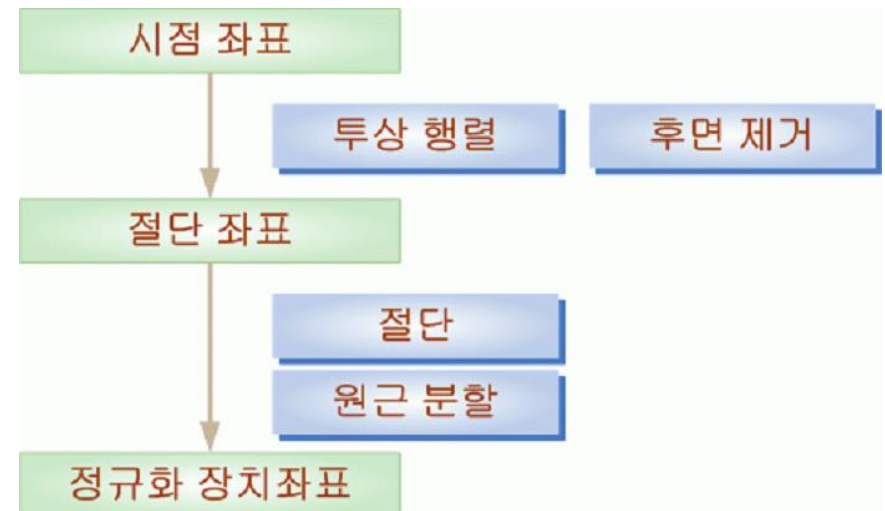
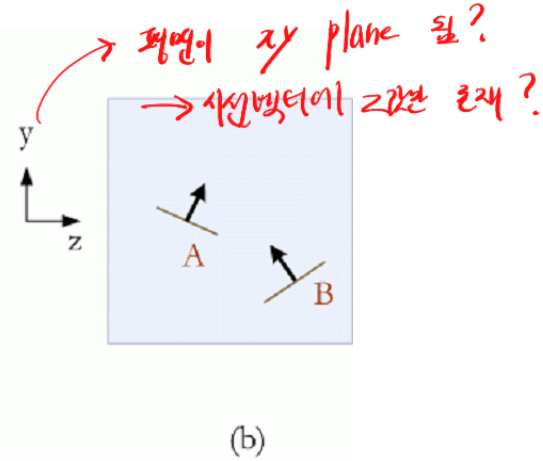
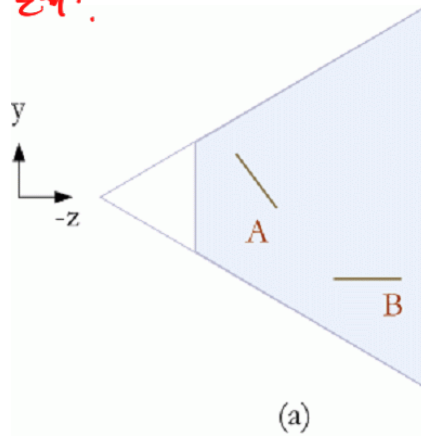
$$\text{Backface} = (N \cdot V < 0) = (|N| |V| \cos \theta < 0)$$

내적값을 때 0보다 작다.

2.1 지엘의 후면 제거

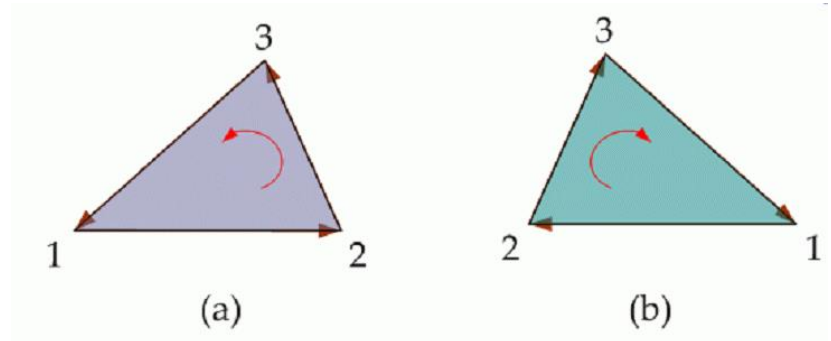
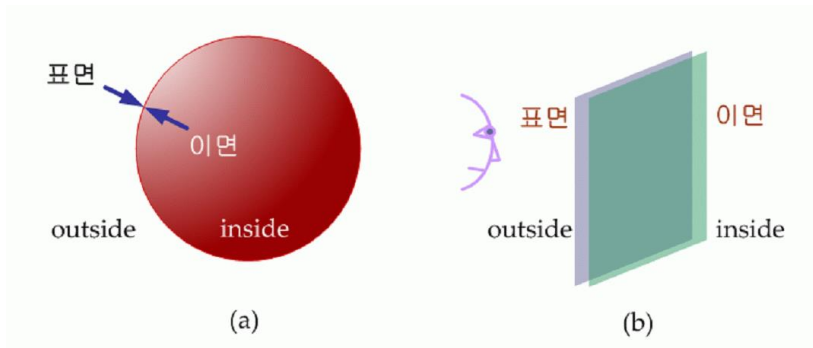
- 정규화 가시부피에서는 법선벡터의 z 값만으로 판단 가능

선벡터 값: z 값만 문제?
 x, y, z 값 2인..



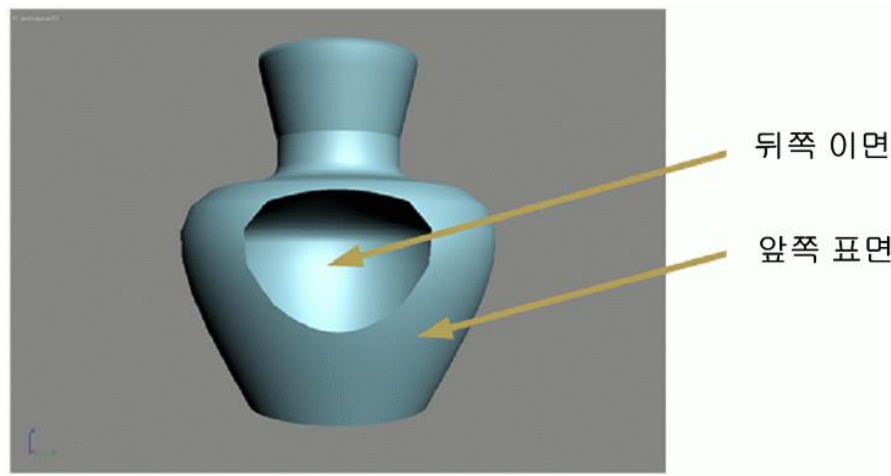
2.2 표면과 이면

- 하나의 면 = 표면 + 이면
- 표면: 반시계방향으로 정의 된 면 (꺾)
- 이면: 시계방향으로 정의 된 면 (ㄴ)



2.2.1 후면의 이면

- 시점이 결정되면 다각형의 표면과 이면 중 하나의 면만 보임
- 후면의 이면 = 표면



3. 절단 *렌더링에서 진행*

- 2차원 절단
: 윈도우(Window), 뷰포트(Viewport), 시저 박스(Scissor Box)
- 3차원 절단
: 가시부피(View Volume)
- 절단 다각형
: 절단 사각형(Clip Rectangle)

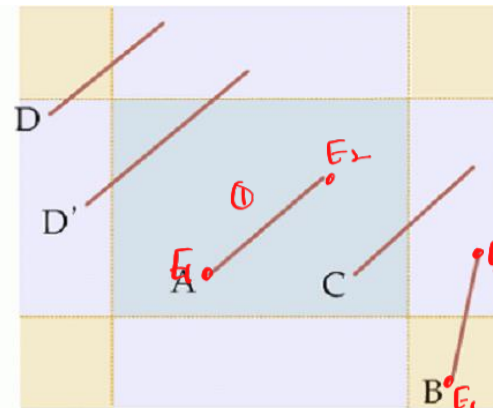


2.1 코헨 서더랜드 알고리즘 논리 활용

- 4비트 아웃코드(Outcode)
- ① • 테스트 1) $E_1 = E_2 = 0000$: 완전히 사각형 내부 선분이므로 보이는 선분으로 판정한다. (선분 A)
- 테스트 2) $E_1 \& E_2 \neq 0000$: 선분이 온전히 절단 사각형 밖에 있으므로 제거한다. (선분 B)
- 테스트 3) $E_1 \neq 0000, E_2 = 0000$ (또는 그 반대) : 교차점 계산에 의해 절단한다. (선분 C)
- 테스트 4) $E_1 \& E_2 = 0000$
 - 양끝점이 모두 절단 사각형 밖에 있지만 서로 다른 선분이다.
 - 교차점 계산에 의해 절단한다. (선분 D, D')

1010	1000	1001
0010	0000	0001
0110	0100	0101

(a)

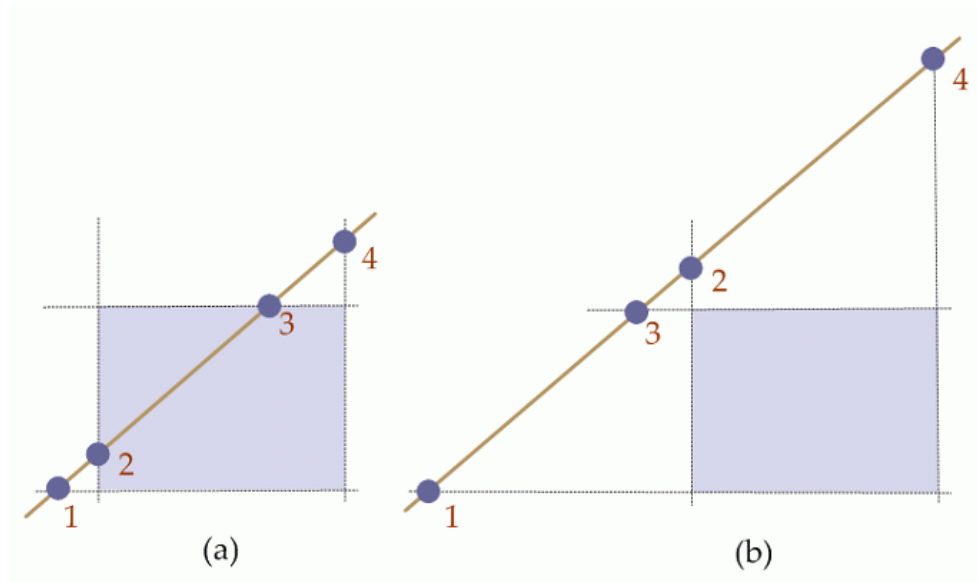


(b)

$E_1: 0000$
 $E_2: 0000$
 논리 결과: 0000
 B) $\begin{array}{r} 0101 \\ 0001 \\ \hline 0001 \end{array}$
 C) $\begin{array}{r} 0000 \\ 0001 \\ \hline 0000 \end{array}$
 D) $\begin{array}{r} 0010 \\ 1000 \\ \hline 0000 \end{array}$

2.2 리앙-바스키 알고리즘

- 교차점에서의 파라미터 값의 순서를 기준으로 여러가지 경우를 판단 (하변, 좌편, 상변 우변과 선분의 교차점)
- 리앙-바스키 알고리즘은 어디에서 어디까지 남겨지는 구간인지를 한번에 판단 가능 (코헨 서더랜드에 비해 강점)



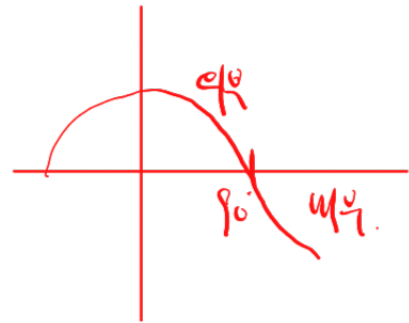
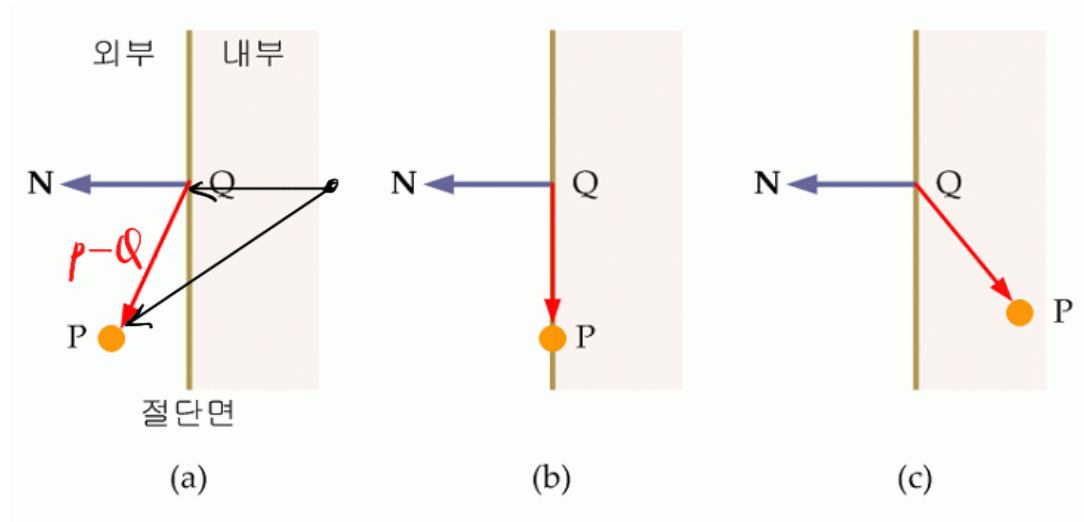
$$0 < t_1 < t_2 < t_3 < t_4$$

$$0 < t_1 < t_3 < t_2 < t_4$$

2.3 정점의 내외부 판정

- 절단 알고리즘의 대부분이 어떤 점이 주어진 절단선 또는 내부인지 외부인지 판단하는 데에서 출발.

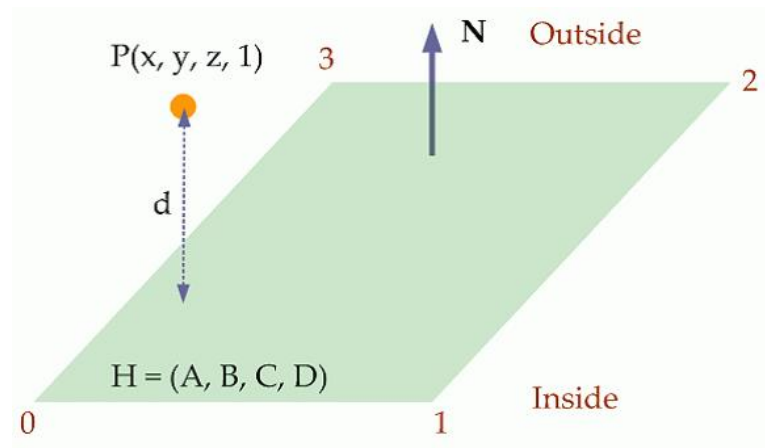
2.3.1 법선 벡터 이용



$(P - Q) \cdot N > 0$ iff P is Outside the Clip Plane
 $(P - Q) \cdot N = 0$ iff P is on the Clip Plane
 $(P - Q) \cdot N < 0$ iff P is Inside the Clip Plane

2.3.2 동차 좌표 이용

- 점과 평면간의 거리
 - 법선벡터 방향이 면의 외부로 정의됨



평면 *법선벡터(N)* *점*

$$\textcircled{d} = H \cdot P = (A, B, C, D) \cdot (x, y, z, 1) = Ax + By + Cz + D$$

$Ax + By + Cz + D > 0$ iff P is Outside the Clip Plane

$Ax + By + Cz + D = 0$ iff P is on the Clip Plane

$Ax + By + Cz + D < 0$ iff P is Inside the Clip Plane

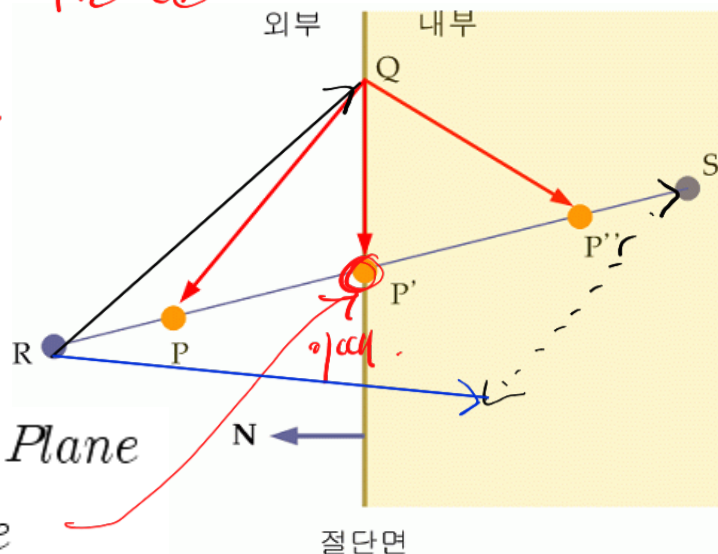
2.4 교차점 계산

$$p(t) = R + t(S - R) = (1-t)R + tS \quad R, S \text{를 지나는 선분}$$

$$x(t) = (1-t)R_x + tS_x \quad \text{선분 위의 점의 } x, y, z \text{ 값.}$$

$$y(t) = (1-t)R_y + tS_y$$

$$z(t) = (1-t)R_z + tS_z$$



$$(p(t) - Q) \cdot N > 0 \text{ iff } P \text{ is Outside the Clip Plane} \quad \text{노출벡터.}$$

$$(p(t) - Q) \cdot N = 0 \text{ iff } P \text{ is on the Clip Plane} \quad \text{직각.}$$

$$(p(t) - Q) \cdot N < 0 \text{ iff } P \text{ is Inside the Clip Plane} \quad \text{음수.}$$

$$p(t) \cdot N = Q \cdot N \quad \text{선분: 머리 안다.}$$

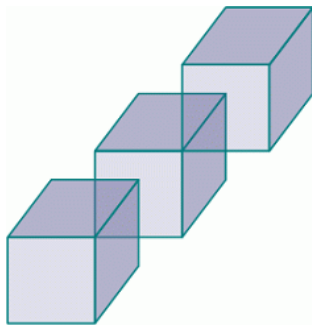
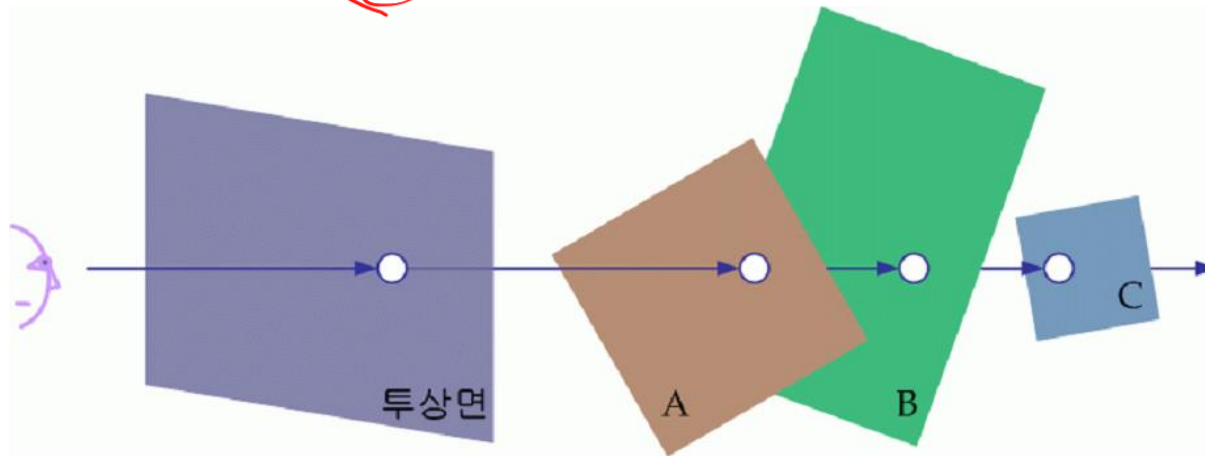
$$(R + t(S - R)) \cdot N = Q \cdot N \quad \text{노출벡터: 머리 안다.}$$

$$t = (Q - R) \cdot N / (S - R) \cdot N \quad \text{길이: 머리 안다.}$$

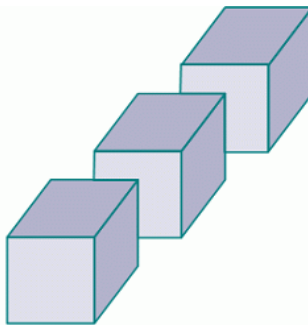
선분이 어느 직선에 교차했는지 알 수 있다

3. 은면 제거

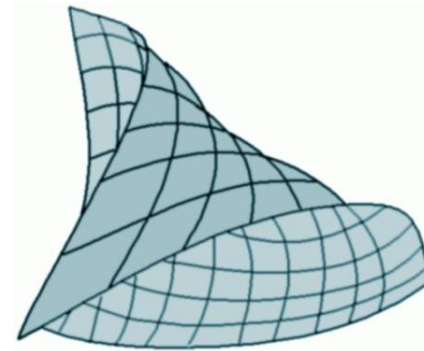
- Hidden Surface Removal
 - 앞 물체에 가려서 안 보이는 부분
 - 물체의 깊이정보(z 값)를 기준으로 판단



(a)

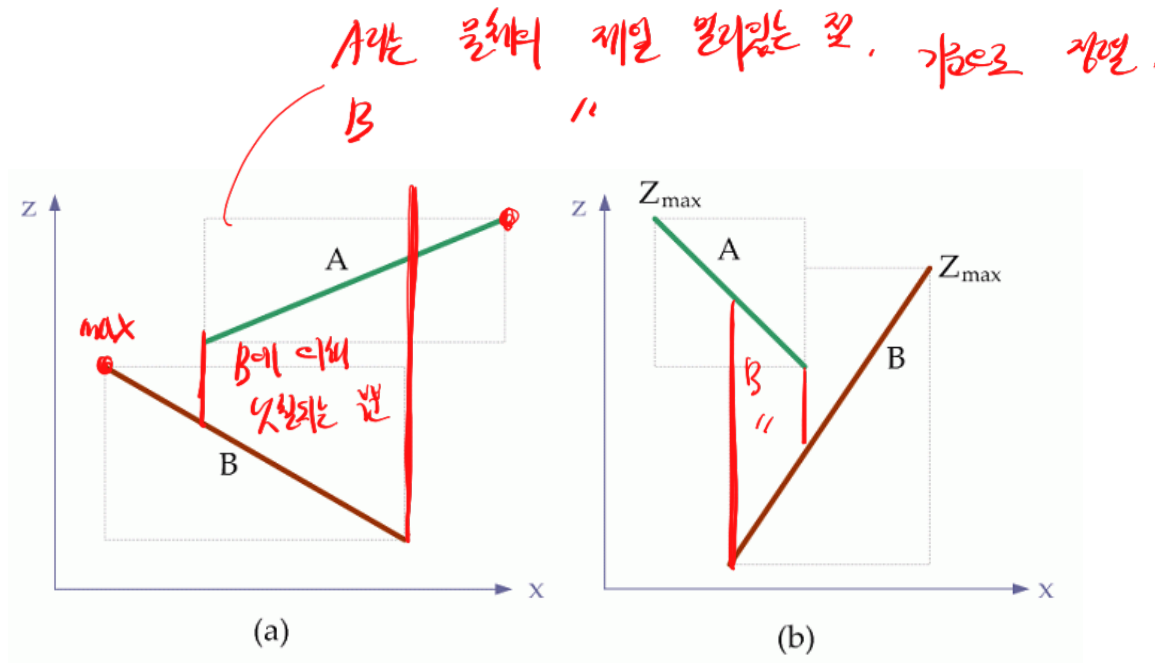


(b)



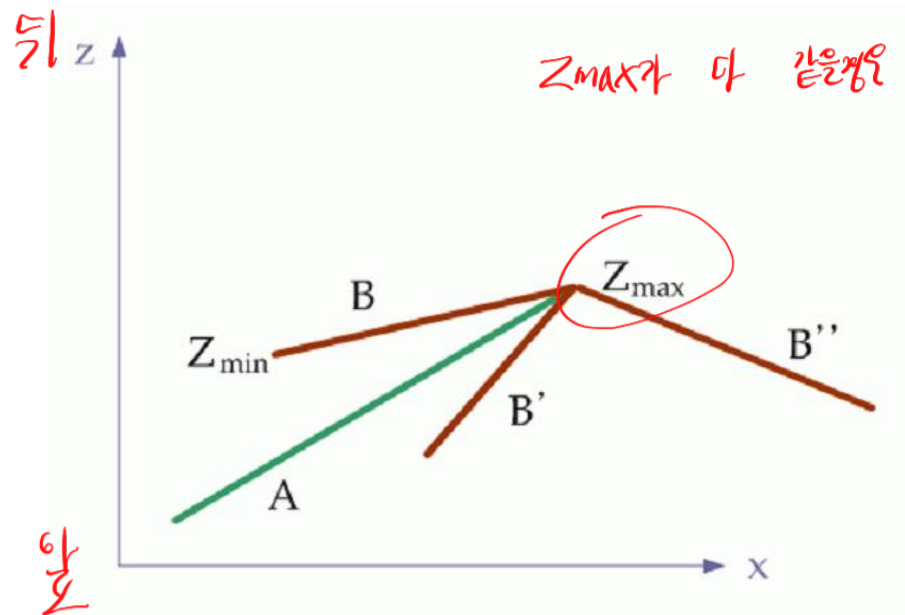
3.1 페인터 알고리즘

- 멀리 있는 배경위에 가까운 물체를 덧칠
 - 깊이 정렬(Depth Sort)이 필요
 - Z_{\max} 를 기준으로 물체를 정렬



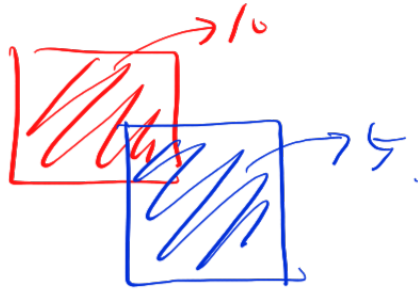
3.1 페인터 알고리즘

- B' , B''
 - Z_{min} 이 A의 Z_{min} 보다 앞에 있으면 그것을 나중에 그려야 함.
- B
 - Z_{min} 이 A의 Z_{min} 보다 뒤에 있으면 그것을 먼저 그려야 함.
- B''
 - x 또는 y 범위가 서로 중첩되지 않으므로 어느 것을 먼저 그리든지 무관함.

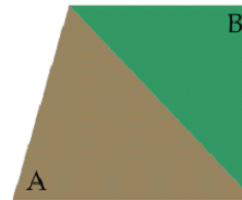


3.2 지-버퍼 알고리즘

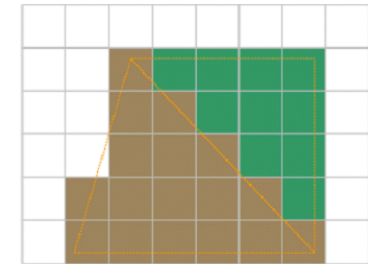
- 깊이버퍼라고 부르는 하드웨어 메모리를 사용 *color값 & depth값 저장.*
- 물체공간 vs. 화소공간
 - 결국 화소공간으로 사상
 - 화소공간 해상도로 은면을 판단하면 됨



depth값 보고 앞어있는 픽셀 선택



(a)



(b)

- 지-버퍼 알고리즘의 시선

