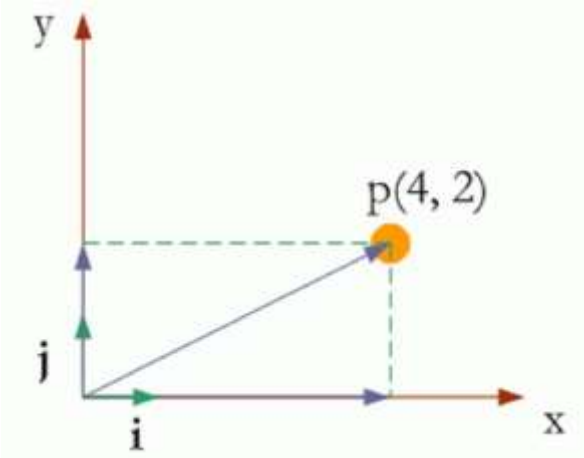
8장. 가시성 판단

학습목표

- 후면제거의 정의와 처리방법을 이해한다.
- 절단작업의 정의와 처리방법을 이해한다.
- 지엘의 절단 방법을 이해한다.
- 은면제거의 정의를 이해한다.
- 지-버퍼 알고리즘을 구체적으로 이해한다.

이것만 중요

- 정규화 벡터(Normalized Vector)
  - 벡터의 크기(절대값)



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

$$\boldsymbol{p}' = \left(\frac{x}{| \boldsymbol{p} |}, \frac{y}{| \boldsymbol{p} |}, \frac{z}{| \boldsymbol{p} |}\right)$$

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

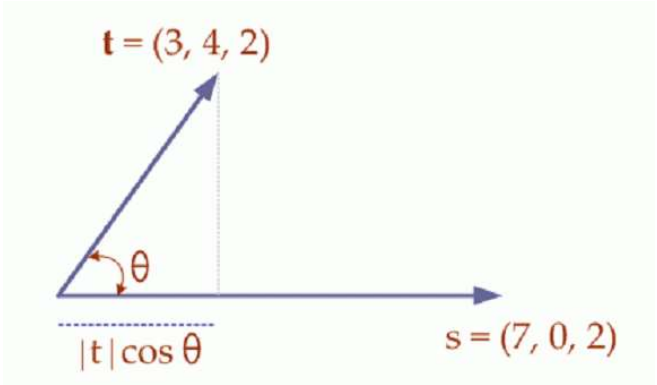
p의 방향은 유지하되 크기가 1이 되게 함

벡터 내적과 외적

- 내적(Inner Product, Dot Product)

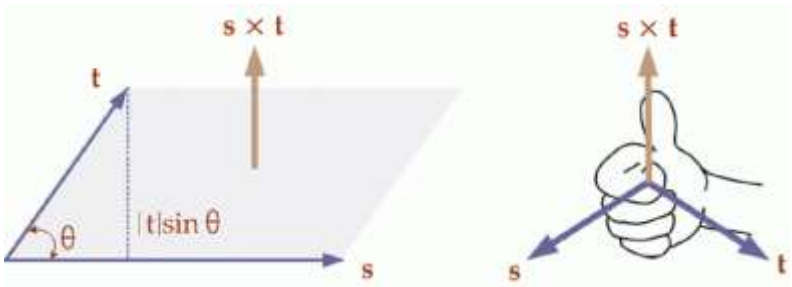
$$s \cdot t = |s||t|\cos\theta = s_x t_x + s_y t_y + s_z t_z$$

  - 두 벡터가 수직이면, 내적은 0
  - 예각이면 +, 둔각이면 -



- 외적(Outer Product, Cross Product)

$$s \times t = |s||t|\sin\theta\, n$$
$$= (s_y t_z - s_z t_y,\ -s_x t_z + s_z t_x,\ s_x t_y - s_y t_x)$$
$$s \times t = -t \times s$$



  - 크기는 삼각형 면적의 의 두 배, 방향은 삼각형 면의 수직방향

- 정규화 법선벡터 = 정규화 외적벡터
  - 법선벡터를 구하고 싶으면 두 벡터를 먼저 구한 후 외적하라.

평면 표현

- 주어진 (법선 벡터) N(A,B,C)에 수직인 평면을 구하고 싶다. 단 평면은 알고 있는 점 Q를 지난다.
  - 어떤 점 P(x,y,z)가 있어서, PQ 선분과 N이 수직하면 조건 만족.
  - P는 평면에 속한다.

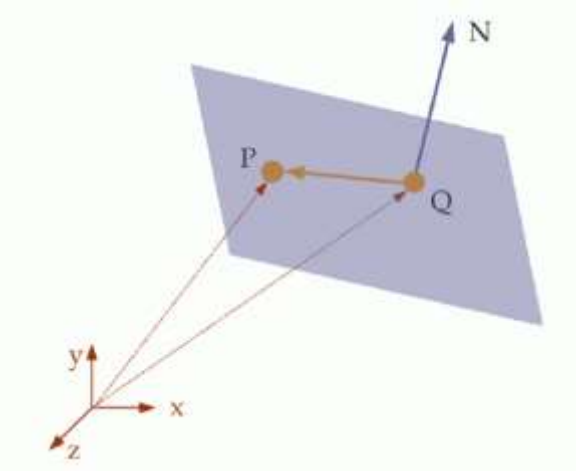
$$(P - Q) \cdot \boldsymbol{N} = 0$$

$$P \cdot \boldsymbol{N} = Q \cdot \boldsymbol{N}$$

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

$$Ax + By + Cz = Q \cdot \boldsymbol{N}$$

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



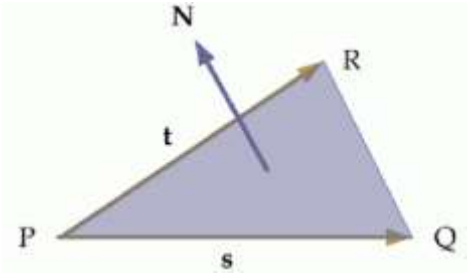
  - 여기서, D = -QN

- 즉, 평면 방정식의 계수 (A,B,C)는 법선벡터를 의미
- 법선 벡터를 구할 방법이 필요하다.

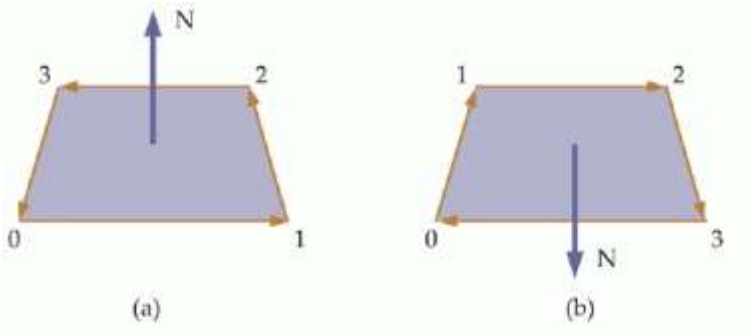
지엘의 법선벡터

- 삼각형을 구성하는 점 PQR의 점의 좌표를 모두 알고 있다고 가정하자. 삼각형의 법선 벡터를 구하라.
  - 삼각형을 구성하는 두 벡터를 찾아 외적

$$s = (Q_x - P_x, Q_y - P_y, Q_z - P_z)$$
$$t = (R_x - P_x, R_y - P_y, R_z - P_z)$$
$$N = s \times t$$

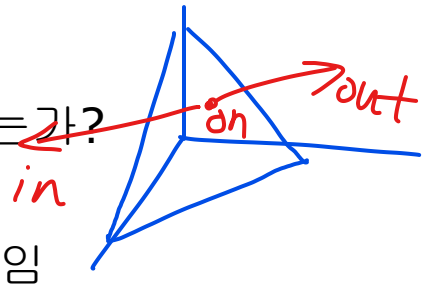


- 법선벡터 방향
  - 만약 N = t X s 로 계산하였다면, 방향이 반대가 된다.
  - 규칙: 오른 손을 명시된 정점 순으로 감싸 쥐었을 때 엄지방향으로 결정
  - s x t : PQR방향으로 감싸 위쪽방향 법선벡터
  - t x s : PRQ 방향으로 감싸 반대방향 법선벡터



내부/외부 판정에서 동차좌표 사용

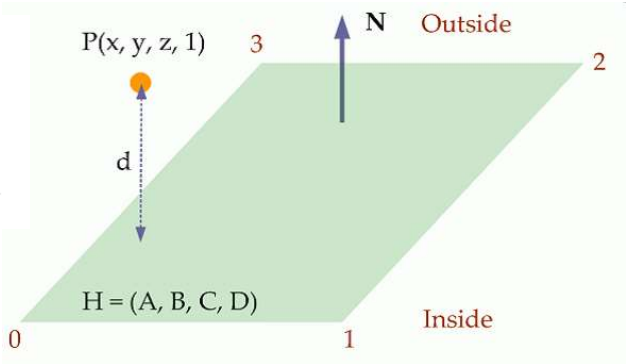
- 1x + 1y + 1z – 1 = 0 이라는 평면이 존재한다.
  - 점 (1,0,0), (0,0,0), (2,3,4)는 평면의 어디에 존재하는가?



- x,y,z를 식에 대입하였을 때, 등식이 만족하면 평면 위의 점임
  - 값이 0이 아닌 음수거나 양수라면 평면을 기준으로 다른 쪽에 위치

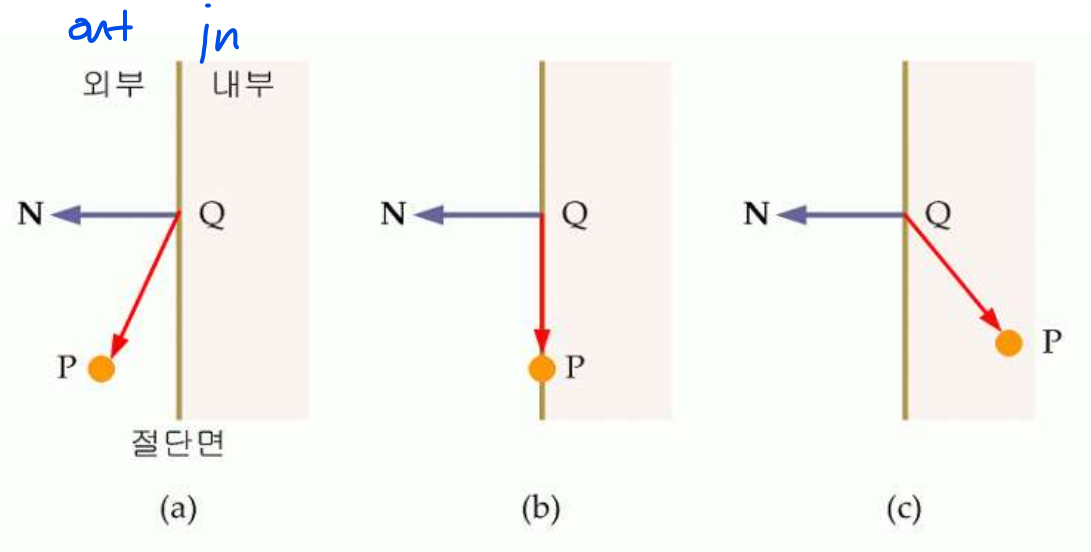
$$d = H \cdot P = (A, B, C, D) \cdot (x, y, z, 1) = Ax + By + Cz + D$$

$$Ax + By + Cz + D > 0 \; iff \; P \; is \; Outside \; the \; Clip \; Plane$$
$$Ax + By + Cz + D = 0 \; iff \; P \; is \; on \; the \; Clip \; Plane$$
$$Ax + By + Cz + D < 0 \; iff \; P \; is \; Inside \; the \; Clip \; Plane$$



- 참고: (wx, wy, wz, w)점에서 w>0 이라면 부호가 동일하므로 그대로 대입 가능
- (A,B,C)가 단위벡터라면, 점과 평면간의 거리 공식으로 사용 가능
  - 법선벡터 방향이 면의 외부로 정의됨

정점의 내외부 판정



$$(P-Q)\cdot N > 0 \; iff \; P is \; Outside \; the \; Clip \; Plane$$

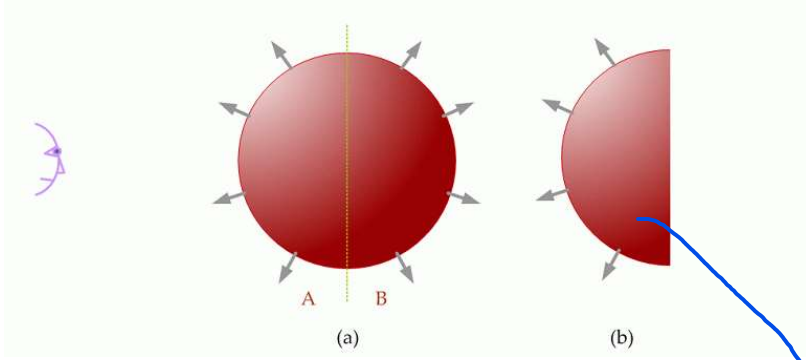
$$(P-Q)\cdot N = 0 \; iff \; P is \; on \; the \; Clip \; Plane$$

$$(P-Q)\cdot N < 0 \; iff \; P is \; Inside \; the \; Clip \; Plane$$

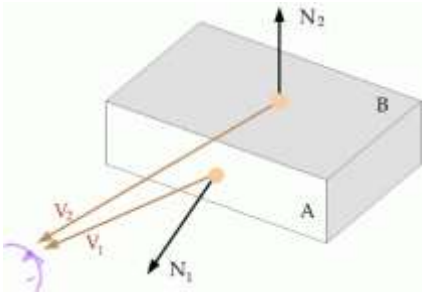
- (P-Q)·N = P · N - Q · N = (x,y,z) · (A,B,C) + D = Ax+By+Cz+D
- 앞 페이지와 동일한 내용

후면

- 법선 벡터가 물체 내부에서 외부 방향으로 나아가도록 설계(정점의 배치 순서를 고려하여 배치)되었다고 가정하자.



- 전면과 후면
  - 후면(Back-Facing Polygon)
  - 전면(Front-Facing Polygon)
- 후면제거(Backface Culling, Backface Removal)
  - 후면은 전면에 가려져서 안 보인다.
  - 후면을 그리지 않고 생략해도 최종 영상은 불변
  - 시점과 면의 관계로 빠르게 판단 가능
  - 보이지 않는 면의 거의 절반을 제거
  - 단, 볼록한 물체가 구멍이 없이 연결되었을 때(닫혀있을 때)만 성립



if (후면) continue

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

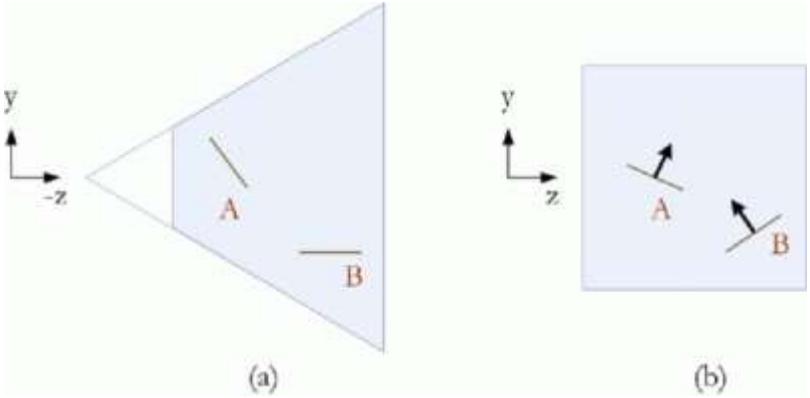
후면

back face에 대해 논할 줄 알기

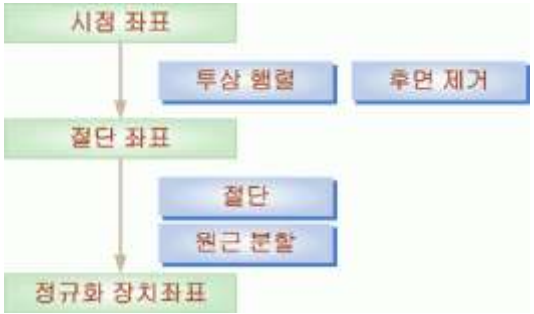
지엘의 후면제거

- 정규화 가시부피에서 판단한다면(b), 법선벡터의 z 값만으로 판단가능
  - 법선벡터의 z성분이 양수이면 제거, 음수이면 그리기
- OpenGL
  - glEnable(GL_CULL_FACE);
  - glCullFace(GL_FRONT);

GL_BACK

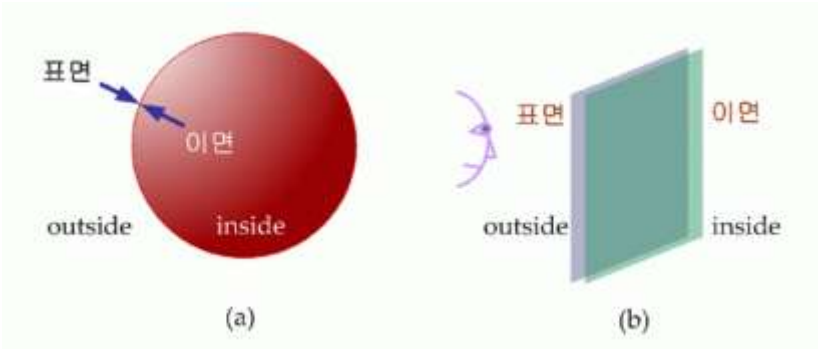


- 참고: OpenGL에서는 후면제거를 원근분할 전에 적용
  - 불필요한 데이터를 가능한 빠르게 제거하기 위해
  - 영상에 맺히는 도형의 넓이가 음수임을 판별하는 방식 사용



표면과 이면

- 하나의 면 = 표면 + 이면

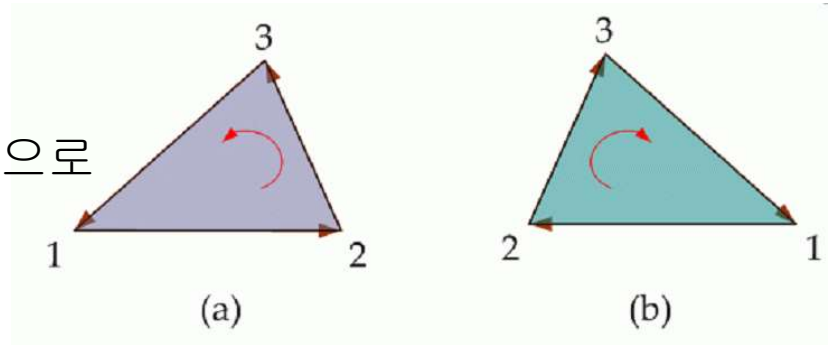


- 표면
  - 반시계방향으로 정의된 면을 표면으로
    - glFrontFace(GL_CCW)

반시계

  - 시계방향으로 정의된 면을 표면으로
    - glFrontFace(GL_CW)

시계방향

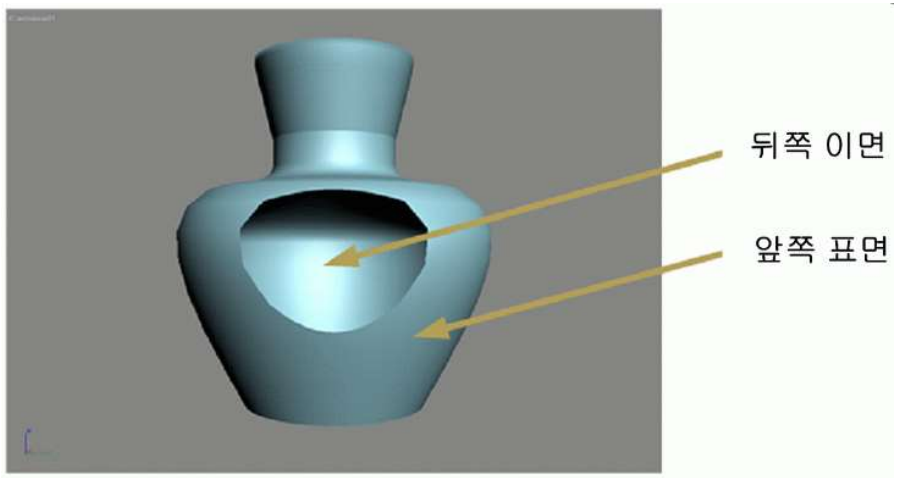


표면과 이면

- 후면의 이면
  - 시점이 결정되면 다각형의 표면과 이면 중 하나의 면만 보임.
  - 지엘은 표면과 이면 중 하나만을 선택하여 그 면으로 해당 다각형을 대신함

- 후면이면 = 표면

원래는 안보여야 뒷면을 지우지만, 이렇게도 가능

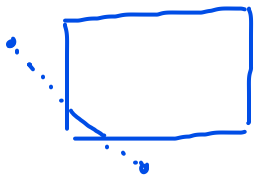


절단(Clipping)

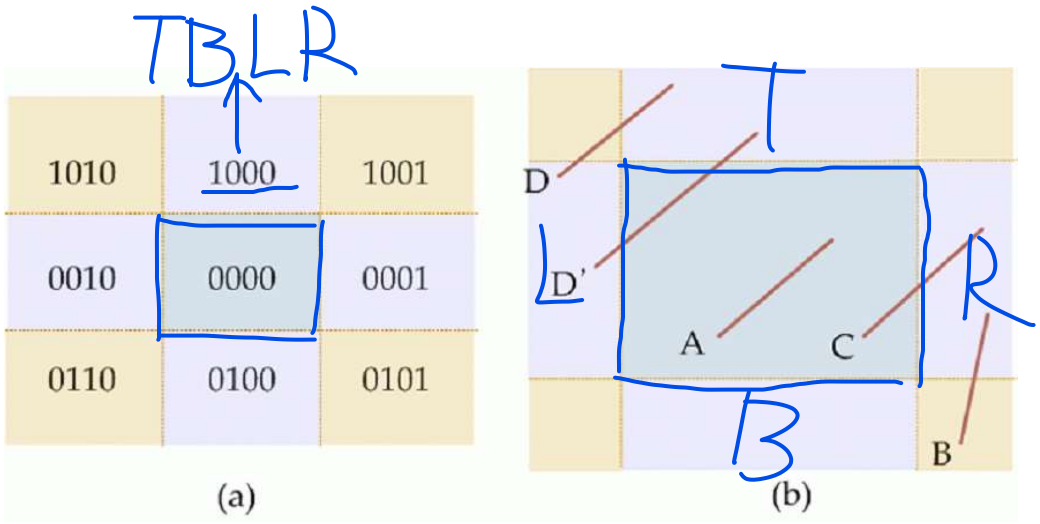
- 2차원 절단
  - 윈도우(Window), 뷰포트(Viewport), 시저 박스(Scissor Box)
- 3차원 절단
  - 가시부피(View Volume)
- 절단 다각형
  - 절단 사각형(Clip Rectangle)

코헨-서더런드 알고리즘

- 4비트 아웃코드(Outcode)



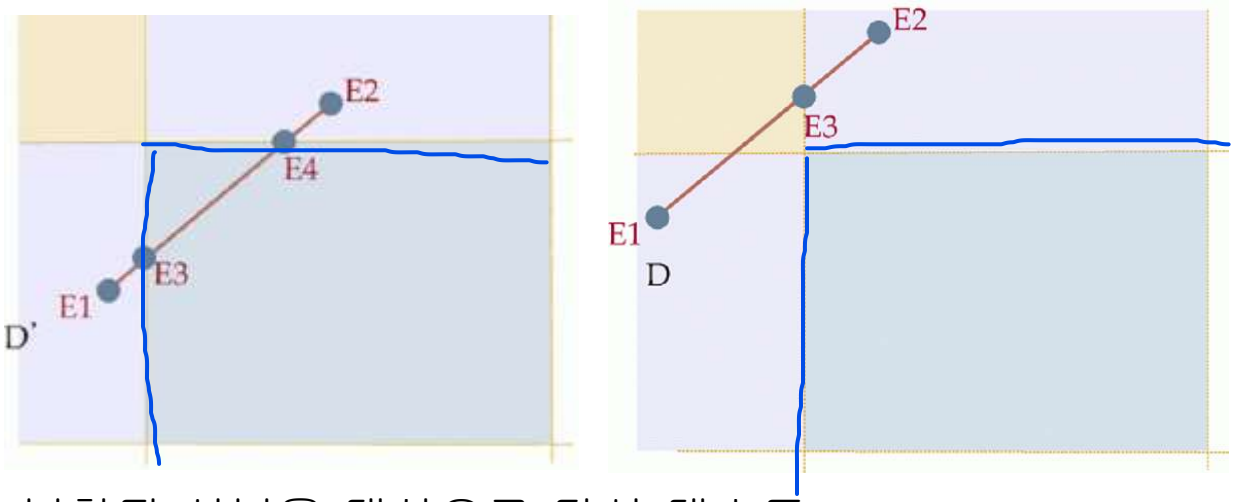
보이는 부분만 그리는 것



- 테스트 1) E1 = E2 = 0000
  - 완전히 사각형 내부 선분이므로 보이는 선분으로 판정한다. (선분 A)
- 테스트 2) E1 & E2 != 0000
  - 선분이 온전히 절단 사각형 밖에 있으므로 제거한다. (선분 B)
- 테스트 3) E1 != 0000, E2 = 0000 (또는 그 반대)
  - 교차점 계산에 의해 절단한다. (선분 C)
- 테스트 4) E1 & E2 = 0000
  - 양끝점이 모두 절단 사각형 밖에 있지만 서로 다른 선분이다.
  - 교차점 계산에 의해 절단한다. (선분 D, D')

코헨-서더런드 알고리즘

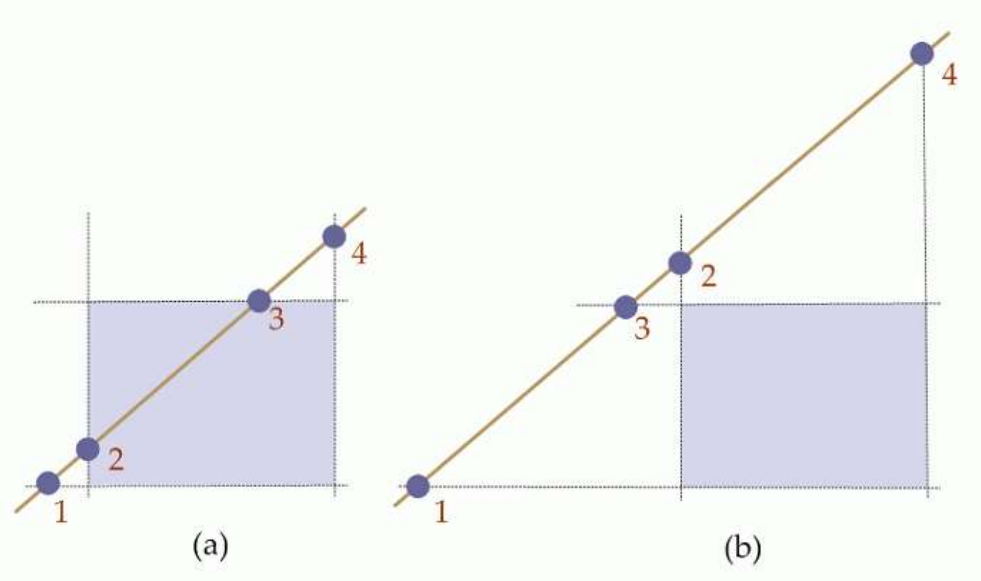
- 선분분할



- 분할된 선분을 대상으로 다시 테스트
- 예)D = E1(0010), E2(1000)
  - 왼쪽선 (또는 위쪽선) 기준으로 E1, E2의 경계점인 E3(1000)의 좌표를 계산한다.
  - 기준선이 왼쪽선이므로, 기준선 밖에 있는 E1을 E3로 옮긴다.
  - 이제 E3,E2 선분에 대해 다시 검사한다. Test 4에 의해 제거된다.
  - Test4: E3 & E2 == 1000 != 0000 이므로 제거

리앙-바스키 알고리즘

- 교차점에서의 파라미터 값의 순서를 기준으로 여러 가지 경우를 판단

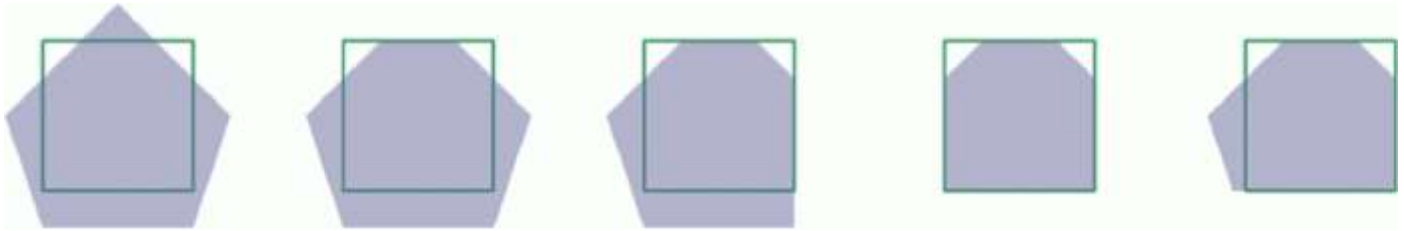


$0 < t_1 < t_2 < t_3 < t_4$

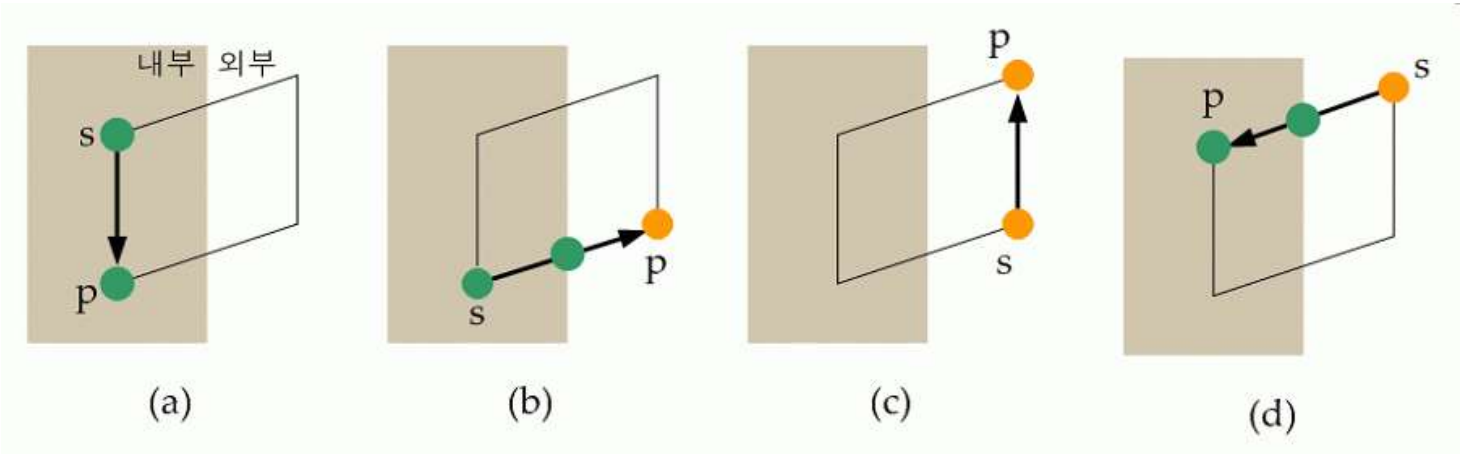
$0 < t_1 < t_3 < t_2 < t_4$

서더런드 핫지먼 알고리즘

- 절단 다각형을 기준으로 순서대로 절단

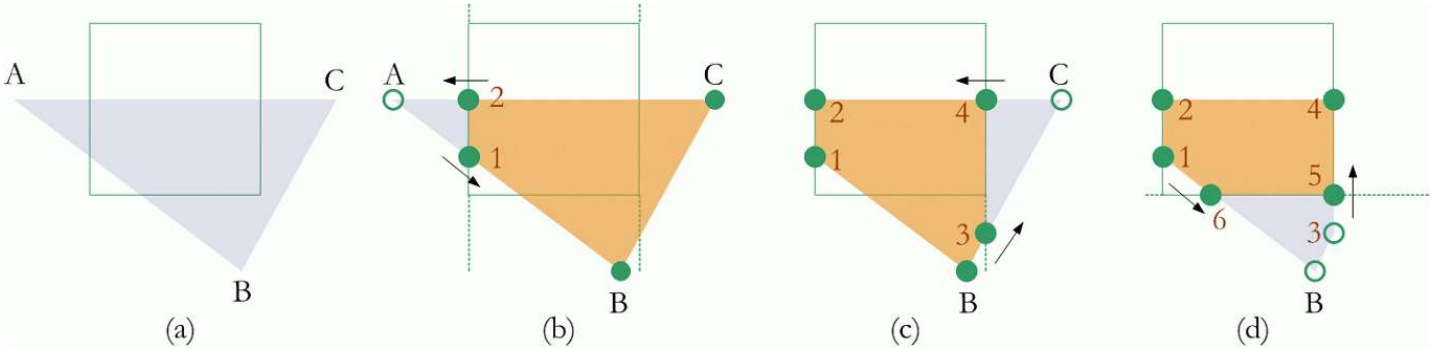


- 절단 규칙

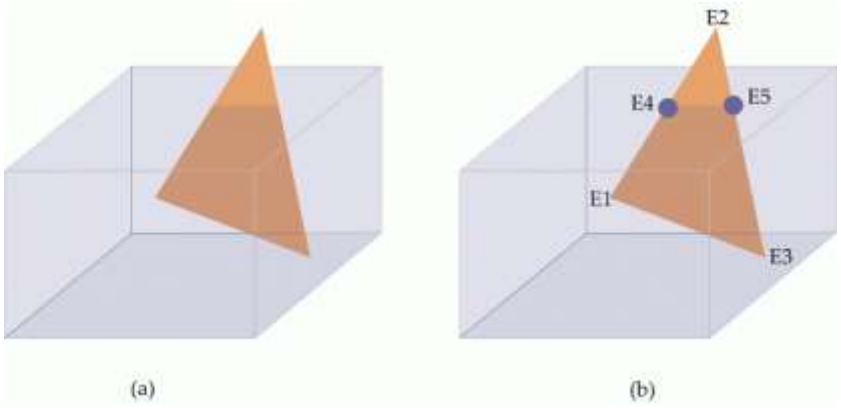


서더런드 핫지먼 알고리즘

- 선분을 연장한 직선을 기준으로 절단
- Ex. 좌변기준의 절단



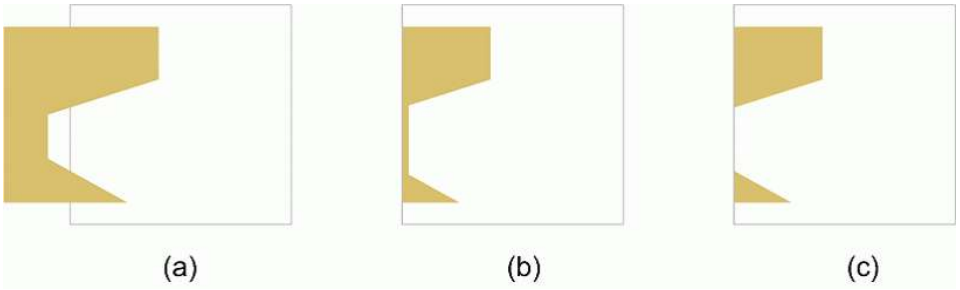
- 3차원 절단
  - 상, 하, 좌, 우, 전, 후의 6개의 면을 기준으로 절단
  - 면을 기준으로 내외부 판정



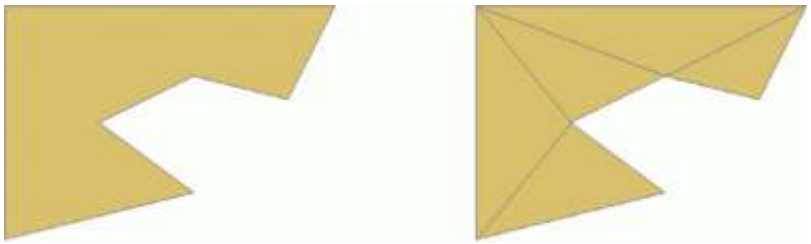
- Silicon Graphics의 Geometry Engine에 사용

볼록, 오목

- 서더런드-핫지먼
  - 볼록 다각형에만 적용
  - 하나의 다각형으로 취급
  - 오목 다각형 처리결과: 오류



- 해법 1: 다각형 분할(Tessellation)
  - 오목 -> 볼록

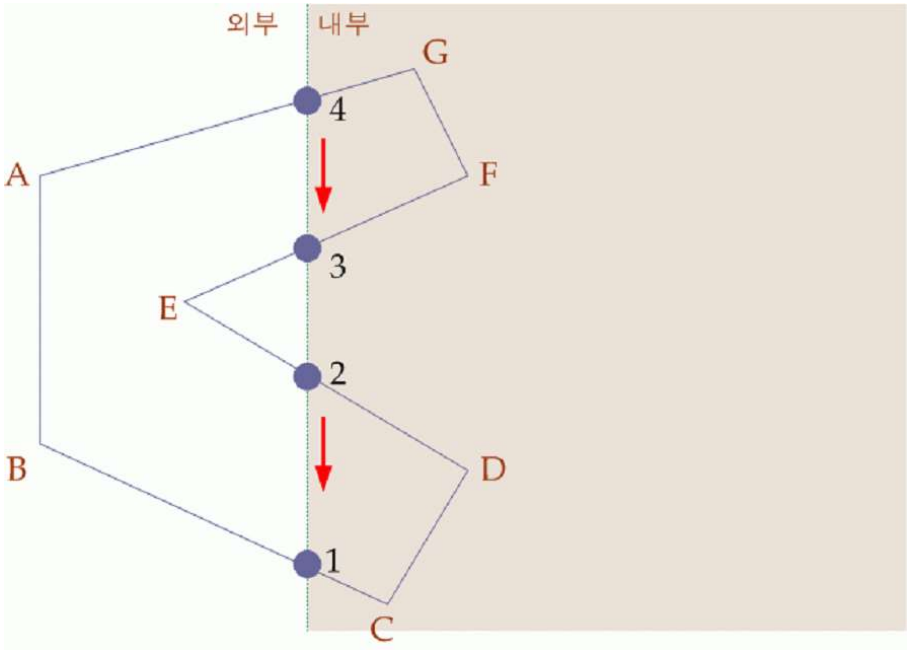


- 해법 2: 웨일러-애서톤 알고리즘

웨일러-애서톤 알고리즘

- 내부에서 외부로 가는 교차점이 추가되면 즉시 그 교차점으로부터 절단 사각형을 따라서 반 시계 방향으로 간다. 즉, 가장 최근에 외부에서 내부로 들어온 교차점을 만날 때까지 간다.

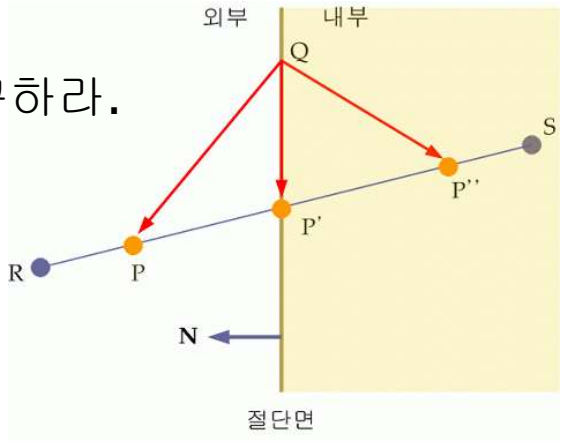
- 1-C-D-2로 구성되는 하나의 다각형이 완성



- 분리된 여러 개의 다각형을 생성함

교차점 계산

- 평면 위의 한 점 Q와 법선벡터 N을 알고있다.
- 두 점 S,R을 지나는 직선이 평면과 만나는 점을 구하라.



$$p(t) = R + t(S - R) = (1-t)R + tS$$

$$x(t) = (1-t)R_x + tS_x$$

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

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

- 선분 QP'과 벡터 N은 수직이어야 하므로,

$$(p(t) - Q) \cdot N > 0 \; iff \; P is \; Outside \; the \; Clip \; Plane$$

$$(p(t) - Q) \cdot N = 0 \; iff \; P is \; on \; the \; Clip \; Plane$$

$$(p(t) - Q) \cdot N < 0 \; iff \; P is \; Inside \; the \; Clip \; Plane$$

- 방정식을 풀어, t값을 얻으면 좌표를 구할 수 있다.

$$p(t) \cdot N = Q \cdot N$$

$$(R + t(S - R)) \cdot N = Q \cdot N$$

$$t = (Q - R) \cdot N / (S - R) \cdot N$$

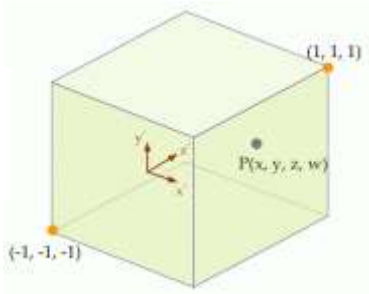
지엘의 절단

- 3차원좌표(x′, y′, z′)

$$-1 \leqq x' \leqq 1, -1 \leqq y' \leqq 1, -1 \leqq z' \leqq 1$$

- 원근변환 행렬에서 z∈[-n, -f]에서 w'=-z > 0 이므로
- w'가 음수인 경우는 원근변환에서 눈 뒤에 있어 보이지 않는다.
- 정규화 장치좌표계

$$-1 \leqq x/w \leqq 1, -1 \leqq y/w \leqq 1, -1 \leqq z/w \leqq 1$$



- 절단 좌표계 (동차 좌표계에서 절단)
- w'가 음수인 경우도 허용한다고 가정하면

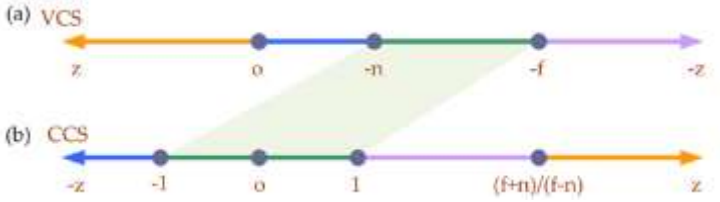
$$Case\ w > 0 : -w \leqq x \leqq w, -w \leqq y \leqq w, -w \leqq z \leqq w$$

$$Case\ w < 0 : -w \geqq x \geqq w, -w \geqq y \geqq w, -w \geqq z \geqq w$$

고급 주제 - 지엘의 절단

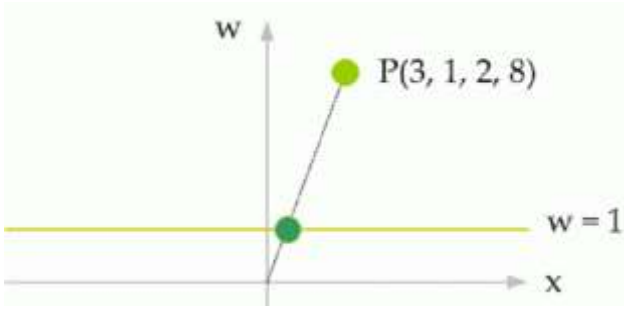
- 관찰 가능 영역 [-n, -f]의 범위 내 점 P가, 변환 후는 w' = –z 이므로, 양수이다. z'/w'의 범위는 [-1, 1]이다.
- 눈보다 뒤에 있는 점 Q(z>0)가, 변환 후 w'=-z<0 (참고로 z' << 0), 그리고 그림에서 z'/w' > (f+n)/(f-n) > 0 이다.
- 예) n=1,f=2에서 Q' = M*Q(0,0,n,1)=(0,0,-3,-1)=(0,0,3,1), clip

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

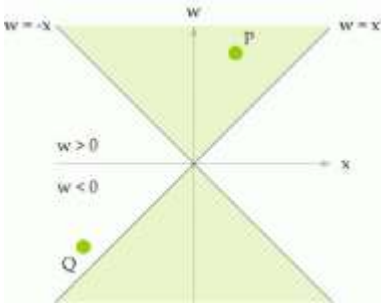


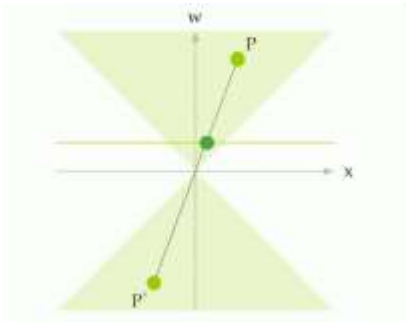
- 만약 (원근 분할 후) 정규화 장치 좌표계에서 절단하면, z'/w' 값을 이용하여 절단하므로, P[-1,1]와 Q(>>0)사이의 절단은 +1에서 교점 발생
- 따라서 원근 분할 전에 w의 부호를 보고 세심하게 clip해야 한다.

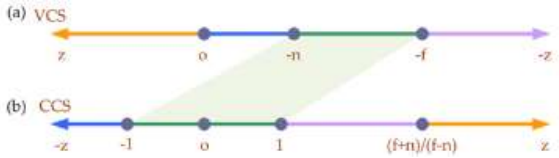
고급 주제 - 지엘의 절단



- 서더런드 핫지만 알고리즘과 유사
  - 지엘은 4차원 절단
  - 4차원 교차점 계산이 필요
  - P(3,1,2,8) = P'(3/8, 1/8, 2/8, 1)
  - 만약, x'와 w'값만 관심 있는 경우, 선분 OP와 직선 w=1의 교점이 실제 공간 점의 좌표이다.
- 내부점, 외부점, 동일점
  - 모래시계 영역 내부가 의미 있는(보이는) 영역
  - 원점에 대해 대칭이면 동차 좌표계 정의상 같은 점이다.
  - Q' = M*Q(0,0,n,1)=(0,0,-3,-1)에서 관찰한 바와 같이, 원근 투영으로는 w'<0 이면서 모래시계 영역에 속하는 경우는 없다.

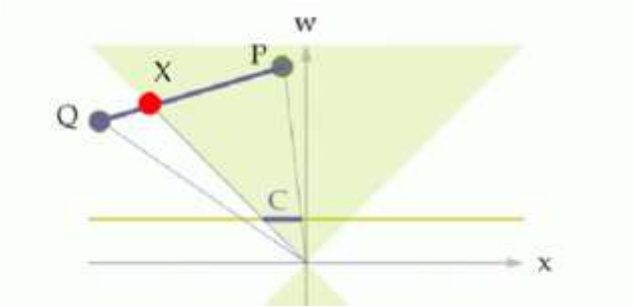






고급주제 - 선분의 절단

- View volume의 내부와 외부 사이의 절단은 간단한 교점 찾기 문제.
- 선분 PQ의 절단된 결과를 얻고 싶다고 가정한다.
- 점P(R_x, R_y, R_z ,R_w)와 점Q(S_x, S_y, S_z, S_w)를 알고 있다고 가정한다.



$$x(t) = R_x + t(S_x - R_x)$$
$$y(t) = R_y + t(S_y - R_y)$$
$$z(t) = R_z + t(S_z - R_z)$$
$$w(t) = R_w + t(S_w - R_w)$$

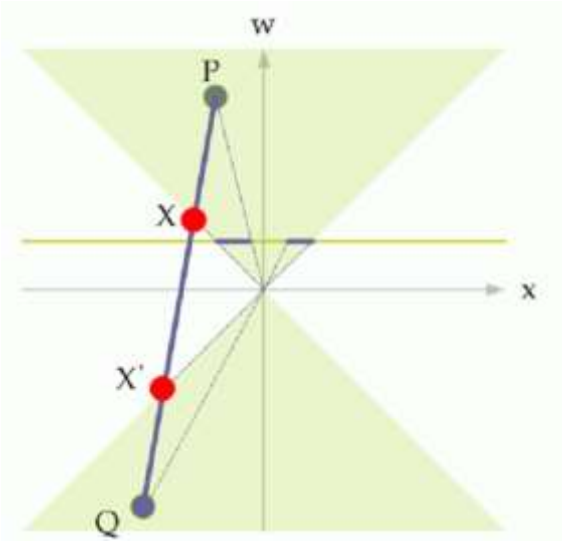
- 만약, 절단면 x=w 에 대해 교점을 찾고 싶다면, x-w = 0 이므로,

$$x(t) - w(t) = R_x - R_w + t((S_w - R_w) - (S_x - R_x)) = 0$$
$$t = (R_x - R_w)/((S_w - R_w) - (S_x - R_x))$$

- 구한 t값을, 위의 x(t), y(t), z(t), w(t)에 대입하여 교점 좌표 얻음

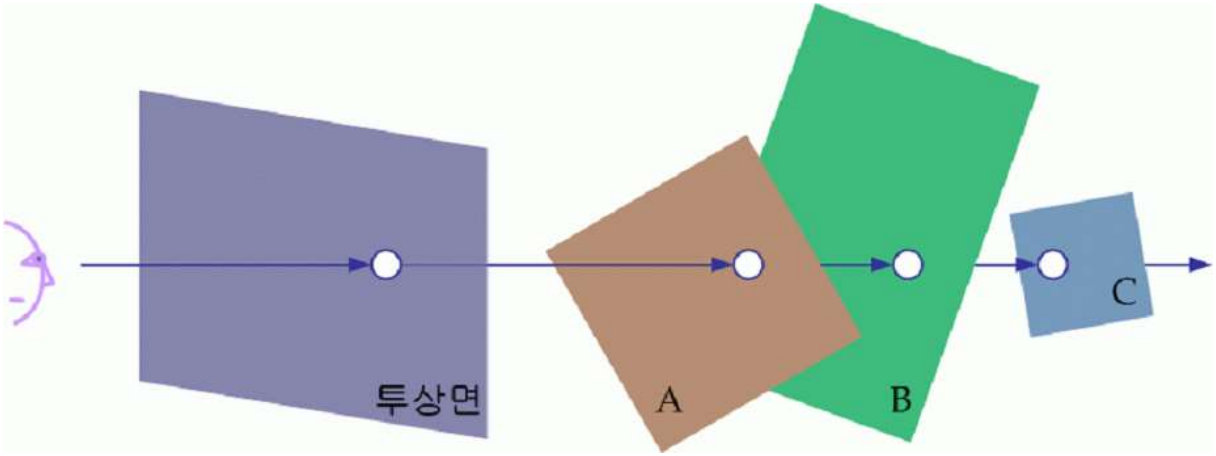
고급주제 - 선분의 절단

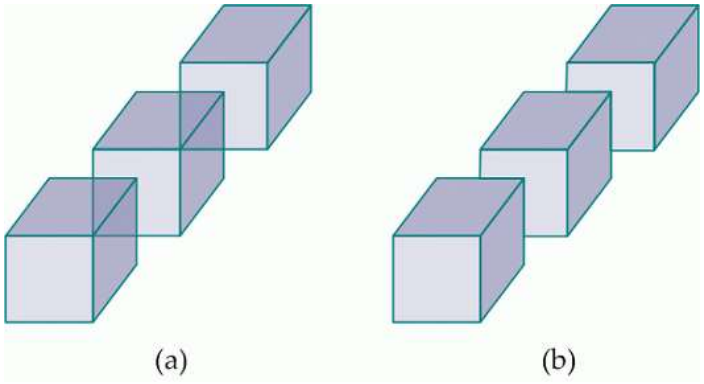
- 단, 두 점이 모두 모래시계 영역내부이며 각각 아래(w<0),위(w>0)에 존재하는 경우는 원근 투영이 아닌 매우 특수한 경우임
  - 고차 곡선 사용 등
  - 이 경우, 선분을 따라 점이 움직이며, w=1에 투영된 점의 자취를 찾는 것을 목적으로 한다.

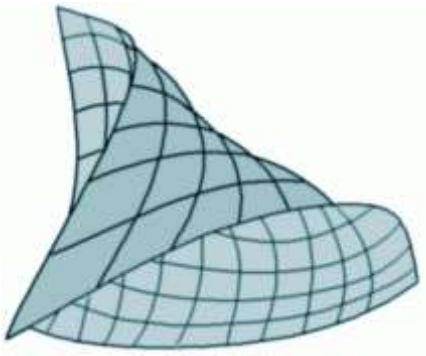


은면제거

- Hidden Surface Removal
  - 앞 물체에 가려서 안 보이는 부분을 제거하는 것
  - 물체의 깊이정보(z값)를 기준으로 판단



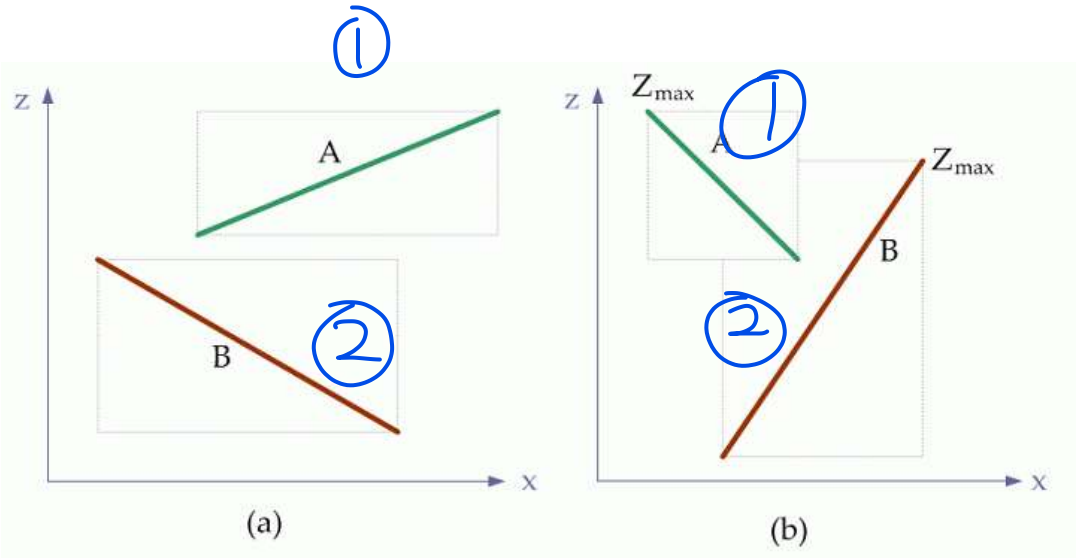




페인터 알고리즘

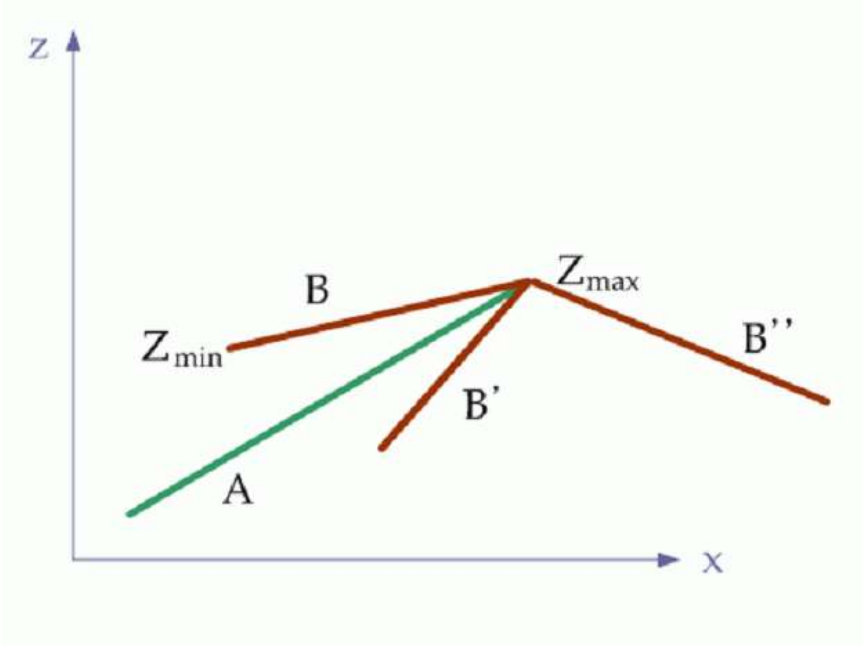
- 멀리 있는 배경위에 가까운 물체를 덧칠
  - 깊이 정렬(Depth Sort)이 필요
  - Zmax를 기준으로 물체를 정렬

지버퍼 알고리즘과
페인터 알고리즘을 구분할줄 알기



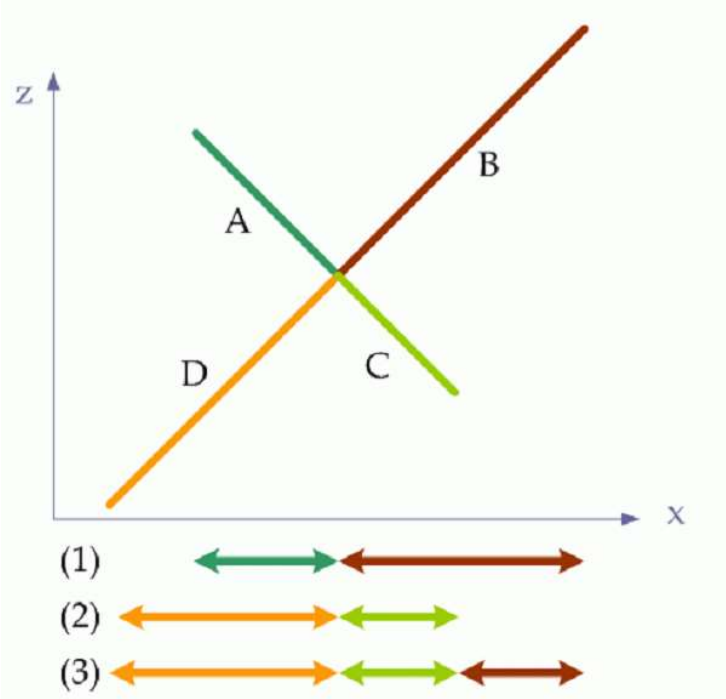
페인터 알고리즘

- 그림에서 z가 작을수록(원점에 가까울 수록) 눈과 가까움
- B′, B″
  - Zmin이 A의 Zmin보다 앞에 있으면 그것을 나중에 그려야 함.
- B
  - Zmin이 A의 Zmin보다 뒤에 있으면 그것을 먼저 그려야 함.
- B″
  - x 또는 y 범위가 서로 중첩되지 않으므로 어느 것을 먼저 그리던지 무관함.



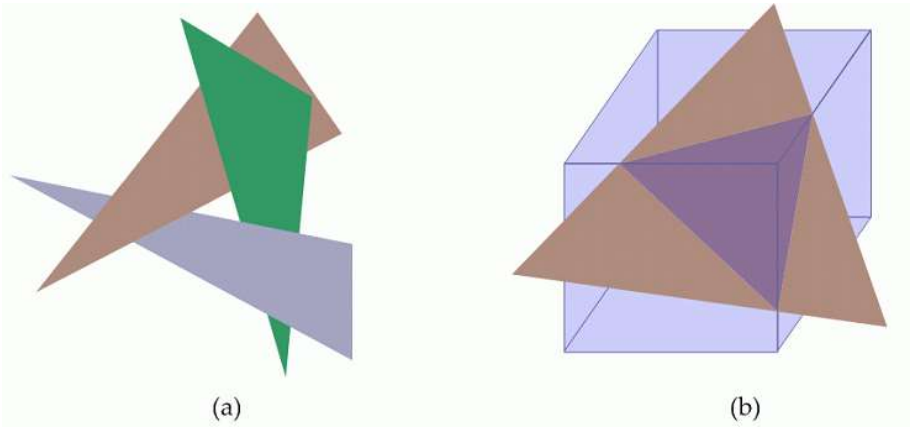
페인터 알고리즘

- 면의 분할
- (1)
  - 먼저 A, B를 Zmax 기준으로 그려냄.
- (2)
  - C와 D는 Zmax는 같지만 x(또는 y)의 범위가 중첩되지 않으니 어느 것을 먼저 그려도 무방함
- (3)
  - 최종결과



페인터 알고리즘

- 가시성 사이클(Visibility Cycle)과 침투(Penetration)



- 페인터 알고리즘
  - 물체공간 알고리즘(Object Space Algorithm)
    - 정밀도는 높지만 실행속도가 느림
  - 경우에 따른 처리가 매우 복잡함.

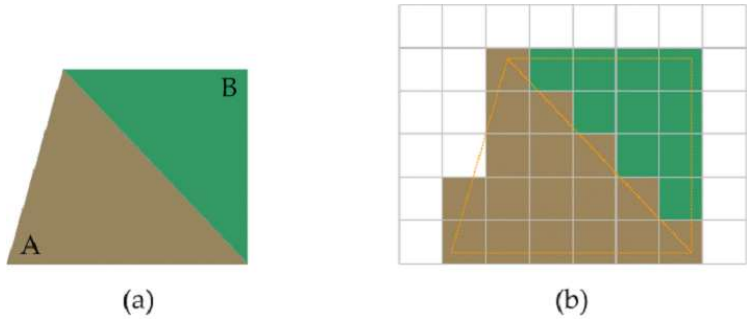
정렬 비용이 많이 든다 ($N \log N$)

지-버퍼 알고리즘

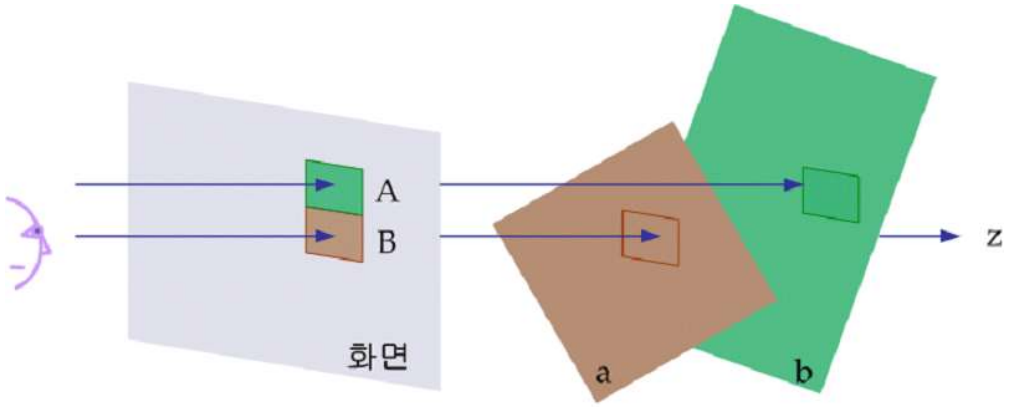
- 물체공간 vs. 화소공간
  - 결국 화소공간으로 사상
  - 화소공간 해상도로 은면을 판단하면 됨

각 픽셀에 대해서, 현재까지 가장 눈과 가까운 픽셀이 무엇인지 거리를 기억

하드웨어 가속?



- 지-버퍼 알고리즘의 시선



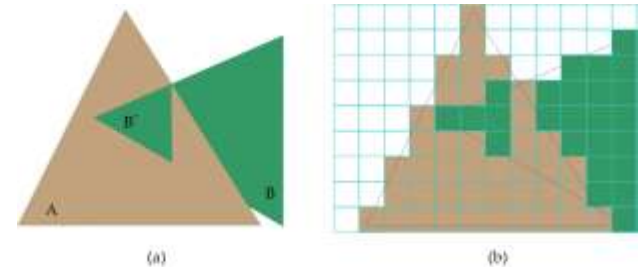
지-버퍼 알고리즘

- 지-버퍼(Z-Buffer) 또는 깊이버퍼(Depth Buffer)

- 지-버퍼 알고리즘

```
Initialize Frame Buffer with Background Color;
Initialize Z Buffer with Infinite Distance;
for Each Polygon {
    for Each Pixel {
        Calculate z of Intersection
        if (Calculated z < Current z of Z-Buffer) {
            Update Z-Buffer with Calculate z;
            Update Frame Buffer with the Color of Current Polygon;
        }
    }
}
```

지-버퍼 알고리즘



1이 젤 먼 것

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

(a)

W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W	W	W	W	W

(b)

z-buffer 에는 현재까지 그려진것 중 가장 가까운 깊이값이 저장됨

1.0	1.0	1.0	1.0	1.0	.5	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.5	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.5	.5	.5	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.5	.5	.5	.5	1.0	1.0	1.0	1.0
1.0	1.0	1.0	.5	.5	.5	.5	.5	1.0	1.0	1.0	1.0
1.0	1.0	1.0	.5	.5	.5	.5	.5	.5	1.0	1.0	1.0
1.0	1.0	.5	.5	.5	.5	.5	.5	.5	1.0	1.0	1.0
1.0	1.0	.5	.5	.5	.5	.5	.5	.5	.5	1.0	1.0
1.0	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	1.0

(a)

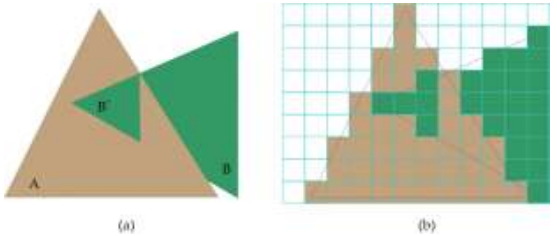
z - buffer

W	W	W	W	W	B	W	W	W	W	W	W
W	W	W	W	W	B	W	W	W	W	W	W
W	W	W	W	B	B	B	W	W	W	W	W
W	W	W	W	B	B	B	B	W	W	W	W
W	W	W	B	B	B	B	B	W	W	W	W
W	W	W	B	B	B	B	B	B	W	W	W
W	W	B	B	B	B	B	B	B	B	W	W
W	W	B	B	B	B	B	B	B	B	W	W
W	B	B	B	B	B	B	B	B	B	B	W

(b)

color buffer

지-버퍼 알고리즘



1.0	1.0	1.0	1.0	1.0	.5	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.5	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.5	.5	.5	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	.5	.5	.5	.5	1.0	1.0	1.0	1.0
1.0	1.0	1.0	.5	.5	.5	.5	.5	1.0	1.0	1.0	1.0
1.0	1.0	1.0	.5	.5	.5	.5	.5	.5	1.0	1.0	1.0
1.0	1.0	.5	.5	.5	.5	.5	.5	.5	1.0	1.0	1.0
1.0	1.0	.5	.5	.5	.5	.5	.5	.5	.5	1.0	1.0
1.0	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	1.0

(a)

W	W	W	W	W	B	W	W	W	W	W	W
W	W	W	W	W	B	W	W	W	W	W	W
W	W	W	W	B	B	B	W	W	W	W	W
W	W	W	W	B	B	B	B	W	W	W	W
W	W	W	B	B	B	B	B	W	W	W	W
W	W	W	B	B	B	B	B	B	W	W	W
W	W	B	B	B	B	B	B	B	B	W	W
W	W	B	B	B	B	B	B	B	B	W	W
W	B	B	B	B	B	B	B	B	B	B	W

(b)

1.0	1.0	1.0	1.0	1.0	.5	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	.5	1.0	1.0	1.0	1.0	1.0	.95
1.0	1.0	1.0	1.0	.5	.5	.5	1.0	.65	.75	.85	.95
1.0	1.0	1.0	1.0	.5	.5	.45	.5	.65	.75	.85	.95
1.0	1.0	1.0	.5	.25	.35	.45	.5	.65	.75	.85	.95
1.0	1.0	1.0	.5	.5	.5	.45	.5	.5	.75	.85	.95
1.0	1.0	.5	.5	.5	.5	.5	.5	.5	.5	.85	.95
1.0	1.0	.5	.5	.5	.5	.5	.5	.5	.5	.85	.95
1.0	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.95

(a)

W	W	W	W	W	B	W	W	W	W	W	W
W	W	W	W	W	B	W	W	W	W	W	G
W	W	W	W	B	B	B	W	G	G	G	G
W	W	W	W	B	B	G	B	G	G	G	G
W	W	W	B	G	G	G	B	G	G	G	G
W	W	W	B	B	B	G	B	B	G	G	G
W	W	B	B	B	B	B	B	B	B	G	G
W	W	B	B	B	B	B	B	B	B	G	G
W	B	B	B	B	B	B	B	B	B	B	G

(b)

선형보간

- 래스터 변환 단계에서 실행
  - 정점으로부터 내부점의 깊이(Depth)및 컬러(Color)를 보간

선형보간을 통해
깊이를 계산