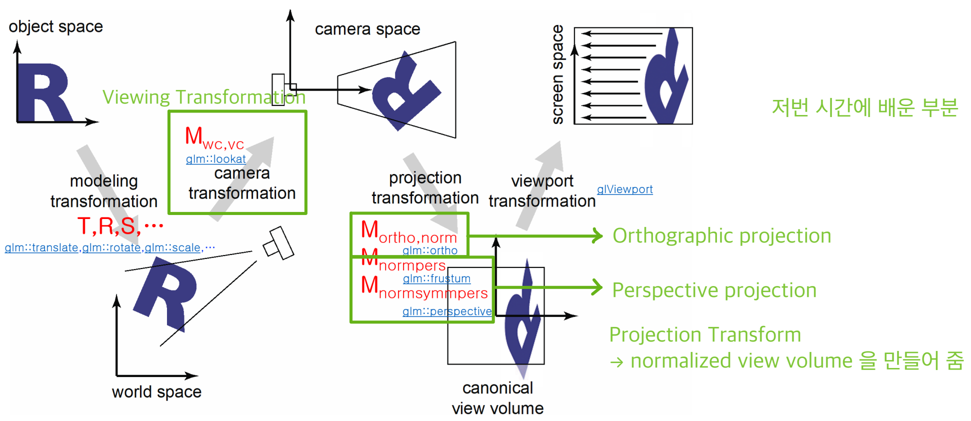
7. Clipping

> **Polygon clipping algorithm**
> - **Sutherland-Hodgman polygon clipping**
> - **Specialized S-H for a triangle**
> - **Acceleration technique**

Graphics Pipeline



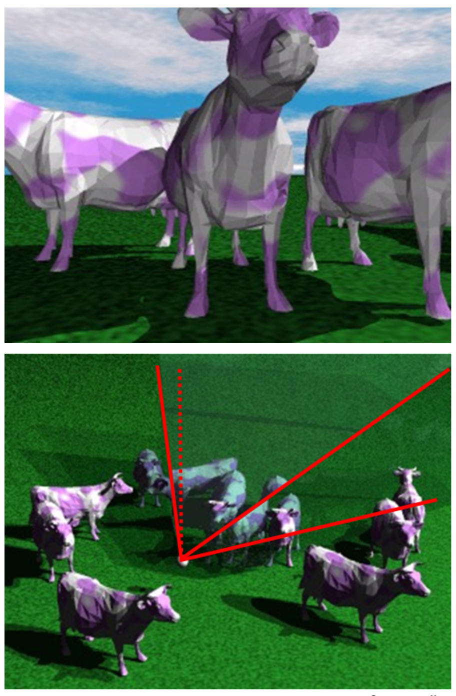
Clipping

- **Clipping: viewing volume 밖으로 벗어난** geometric primitive (점, 선, 면)을 제거

> 왜 Clipping 을 하는가?
> - Pixel 의 color 를 계산하기 위한 과정인 lighting, texturing 은 매우 많은 계산이 필요
> - Clipping 을 한 이후 color 값이 필요한 곳들만 계산해 낭비될 수 있는 계산을 절약
>
>   = Optimization

- 3가지 경우
  1. **Trivial acceptance**: viewing volume 안에 완벽하게 들어오는 경우
     ⇒ Clipping 이 필요 없음
  2. **Trivial rejection**: viewing volume 밖으로 완전히 벗어난 경우
     ⇒ 더 이상 신경 쓸 필요가 없어짐 → 남은 projection, color 계산 x
  3. **Crossing clip plane**: 일부는 vv 안에, 일부는 vv 밖에 있는 경우
     - vv 안에 들어온 부분은 keep, vv 밖 부분은 날려야 함
     - 가장 어렵고 우리가 중점적으로 신경 쓸 부분

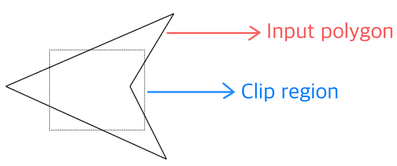
- Primitives 의 관점
  1. 점 (Points)
     - 3번째 경우 불가능
     - Trivial accept/reject 경우만 존재
  2. 선 (Lines)
     - Clip plane 과 만나는 부분에서 잘라내 vv 안쪽 부분만 keep
  3. 면 (Polygon): Vertex 로 정의
     - Clipping 을 하더라도 input 의 vertex 순서를 유지해야함
       = **Connectivity 유지**


courtesy of L. McMillan

Sutherland-Hodgman Clip

(2차원의 경우)

- input polygon: clip 할 polygon
- clip region: clip 되는 영역
  - 3차원에서는 clip volume: normalized view volume
- clip region 밖에 있는 input polygon 부분을 잘라내고 싶음
- 그 과정에서 vertex ordering 은 유지되어야 함



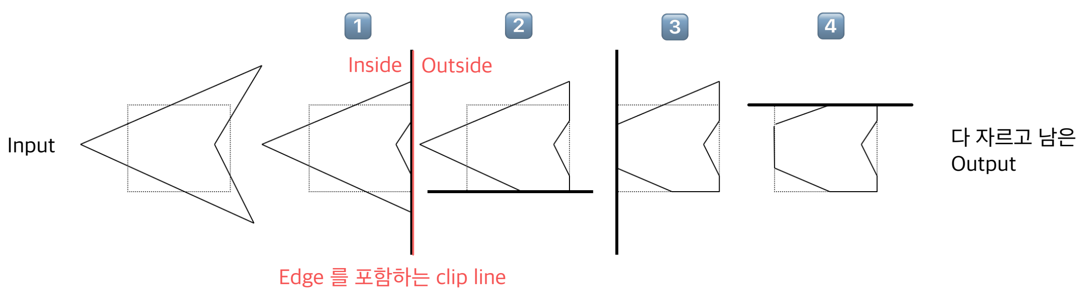
(과정)

1️⃣

1. clip region 의 하나의 edge 선택
2. edge 를 포함하는 **clip line** 을 생각하자
   - clip region 을 포함하는 쪽이 inside, 포함하지 않는 쪽이 outside 가 됨
   - inside 쪽은 keep, outside 쪽은 잘라버림

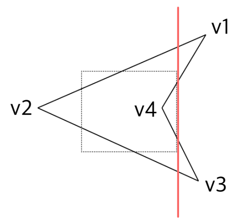
2️⃣ ~ 4️⃣

- 다음 edge 선택 후, 1️⃣ 반복 (돌려서 다 자를 때까지)

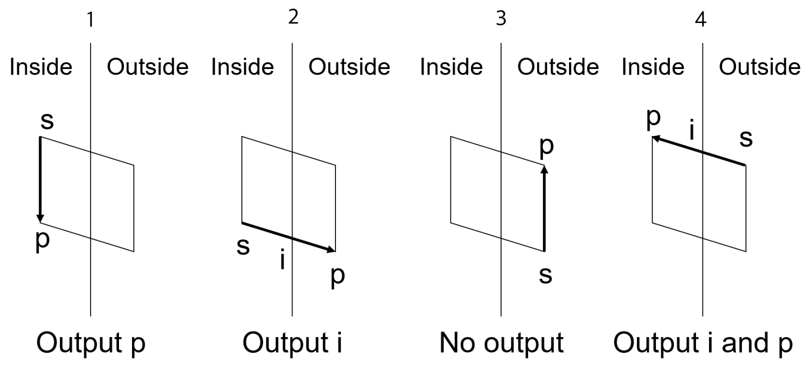


(구현)

- 1️⃣ ~ 4️⃣ 는 모두 유사한 **프로세스 1개의 반복**
- 1️⃣ 에서 vertex 1~4를 순회하면서, 어디가 inside/outside 인지에 따라 자를 것인지 여부 결정
- 순회할 때 4가지 케이스에 따라 적절한 행동을 취함

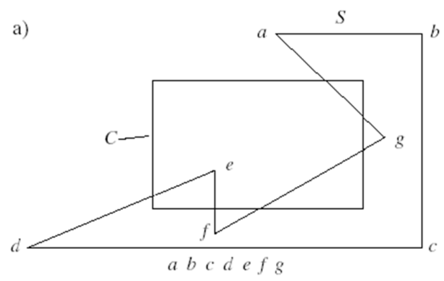


1. 시작점: inside → 도착점: inside
   - **도착점 p 만** 출력
2. inside → outside
   - **clip line 과 polygon edge** 이 만나는 점 **(intersection) i** 를 구해서 출력
3. outside → outside
   - 잘려나가므로 아무것도 출력하지 않음
4. outside → inside
   - case 2 의 반대 버전
   - **i 를 먼저** 출력하고, **도착점 p** 도 출력



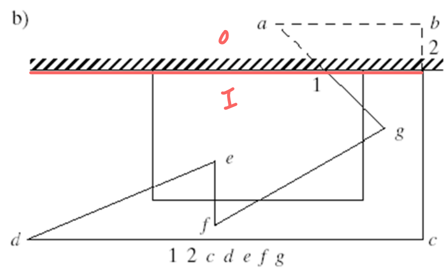
(예시)

- input: vertex 가 7개인 polygon
- clip region: 직사각형



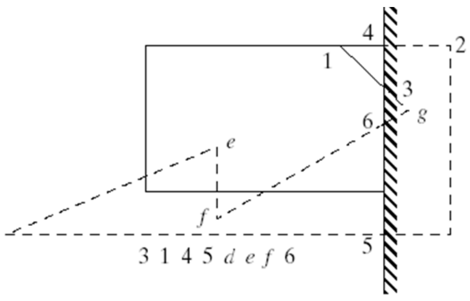
1️⃣

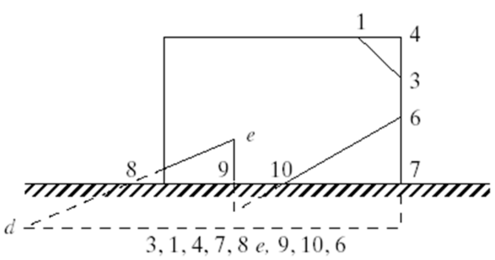
1. clip region 의 edge 선택
2. input polygon 을 돌며 4가지 case 중 무엇이냐에 따라서 적절하게 출력
   - polygon 의 점 g 에서 시계방향으로 순회
   - g → a: inside → outside ⇒ intersection **1**
   - a → b: outside → outside ⇒ skip
   - b → c: outside → inside ⇒ intersection **2**, 도착 vertex **c**
   - c → d → e → f → g: inside → inside ⇒ 각 구간의 도착 vertex **d, e, f, g**
- 여기까지가 첫 번째 iteration ⇒ **1 2 c d e f g** (vertex 순서)

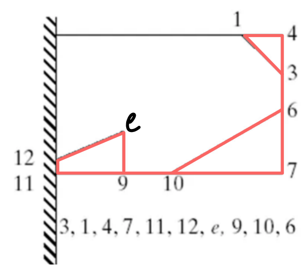


2️⃣ ~ 4️⃣

- 다른 clip region edge 를 선택하고, 1️⃣ 과 비슷한 방식으로 진행





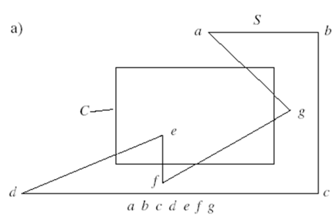


- 최종 결과물: **3, 1, 4, 7, 12, e, 9, 10, 6**
  = 삼각형 2개 + 사각형 1개
- (3, 6), (9, 10) 선은 실제 OpenGL 등에서는 날려버리나, 수업시간에는 포함한 것을 결과물로 생각

(3차원의 경우)

- Viewing pipeline 에서는 viewing volume 이 사각형이 아니라, **정육면체** 형태
  - Clip Line 대신, **Clip plane 을 품는 평면을 가지고 자름**
  - Clip volume 포함하는 쪽의 평면이 inside, 반대쪽이 outside
- 알고리즘
  - 바깥쪽 loop: clip region 의 각 edge 를 도는 것
  - 안쪽 loop: polygon 의 각 vertex 를 도는 것
    - 각 vertex 를 돌며, 현재 clip region 의 edge 와 비교하면서 4가지 case (inside, outside) 에 따라 출력

For each edge/plane (c_i, c_j) in clip region/volume
    For each successive edge (s, p) in input polygon
        1. If both inside (c_i, c_j): Output **p**
        2. If both outside: Output nothing
        3. If **s** inside, **p** outside: Output intersection of (**s**, **p**) with (c_i, c_j)
        4. If **s** outside, **p** inside: Output intersection of (**s**, **p**) with (c_i, c_j), then **p**



💡 알고리즘 실제로 구현하려면 2가지 해결 필요

1. 시작, 도착 vertex 가 **어떤 case 에 해당하는지 어떻게 알 것인가?**
2. inside → outside, outside → inside 경우, **i 를 어떻게 구할 것인가?**

① Inside-Outside Testing

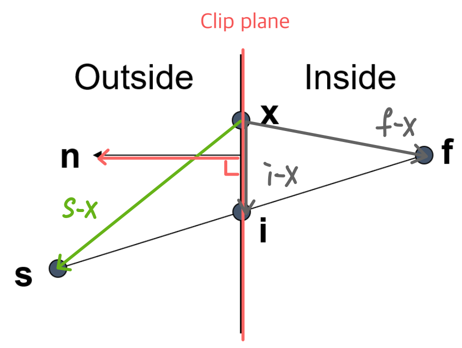
- 시작, 도착 vertex 가 어떤 case 에 해당하는지 알 수 있는 방법

  → vertex 가 clip plane 의 inside 에 있는지 outside 에 있는지 알고 싶음

(가정) **outward pointing normal** = plane 의 수직 방향 벡터 n 은, **clip region 의 outside** 을 가리킴

(방법)

1. clip plane 상의 임의의 점 x 를 잡음
2. inside/outside 여부를 확인하고 싶은 점 s 로 **s - x 벡터**를 구함
3. n • (s - x) 계산
   - **n • (s - x) > 0**: s 가 **outside** 에 있음
     - s 가 outside 에 있다면, s 와 n 이 같은 방향에 놓여있고, n 벡터와 (s - x) 의 끼인 각 < 90 이기 때문
   - **n • (s - x) < 0**: s 가 **inside** 에 있음
     - s 가 inside 에 있다면, s 와 n 이 반대 방향에 놓여있고, n 벡터와 (s - x) 의 끼인 각 > 90 이기 때문
   - **n • (s - x) = 0**: s 가 **clip plane 위**에 있음
     - s 가 clip plane 위에 있다면, n 벡터와 (s - x) 의 끼인 각 = 90 이기 때문



$$\mathbf{n} \bullet (\mathbf{s} - \mathbf{x}) > 0$$
$$\mathbf{n} \bullet (\mathbf{i} - \mathbf{x}) = 0$$
$$\mathbf{n} \bullet (\mathbf{f} - \mathbf{x}) < 0$$

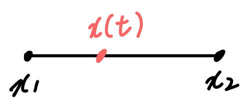
② Finding Intersection Points

- inside → outside, outside → inside 경우, intersection point i 를 구하는 방법
- 선분 (edge) 과 평면이 만나는 좌표를 찾아야 함

(방법)

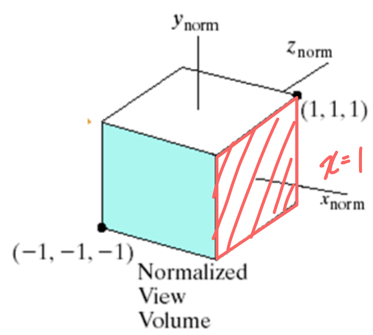
1. 매개변수 t 를 사용해, x_1 와 x_2 를 연결하는 선분을 수식으로 표현

$$\mathbf{x}(t) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)t, \qquad 0 \le t \le 1$$

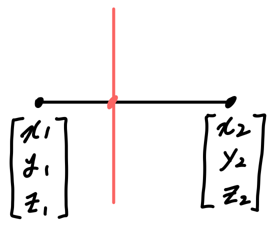


2. 평면의 방정식 정의
   - (일반적인 평면의 방정식의 형태: ax + by + cz + d = 0)
   - **view volume 은 정육면체 형태이므로, clip plane 은 x, y, z 축에 수직**

     ⇒ $x_s = \pm1$, $y_s = \pm1$, $z_s = \pm1$



3. 직선과 평면의 intersection 을 구해야 함
   - 평면의 방정식: x = a, 점 $\mathbf{x}_1$ 을 (x_1, y_1, z_1), 점 $\mathbf{x}_2$ 를 (x_2, y_2, z_2) 라고 해보자



$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} \mathrm{t} = \begin{matrix} x(t) \\ y(t) \\ z(t) \end{matrix}$$

$$x_1 + (x_2 - x_1)\mathrm{t} = \mathrm{a} \qquad \mathrm{t} = \frac{a - x_1}{x_2 - x_1}$$

   - t 를 식에 다시 대입하면, intersection 점 $\mathbf{x}_i$ 를 구할 수 있음

$$\mathbf{x}_i = \left(a, y_1 + \frac{(y_2 - y_1)}{(x_2 - x_1)}(a - x_1), z_1 + \frac{(z_2 - z_1)}{(x_2 - x_1)}(a - x_1)\right)$$

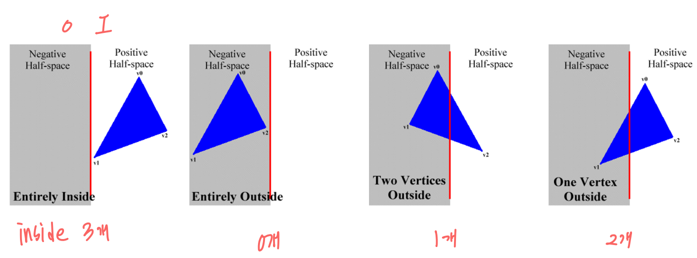
   - **Similar forms for *y = a, z = a***

Sutherland-Hodgman: Triangle

- input polygon 이 삼각형인 경우
  - 앞에서는 일반적인 polygon 에 대한 알고리즘을 배움
  - input polygon 이 삼각형이면 앞의 알고리즘 압축시켜 짧게 구현 가능

(방법)

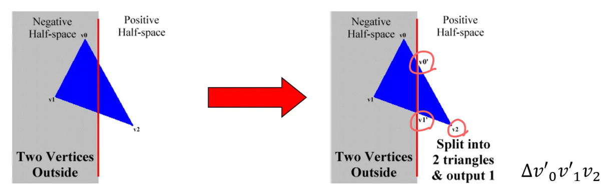
1. **inside-outside testing** 으로, **inside 점 개수 count**
   - 삼각형은 vertex 가 3개이므로, outside/inside 인 점의 개수가 몇 개인지 세어보면 4가지 경우 존재



2. inside vertex 3개: trivially accepted 이므로 다 keep

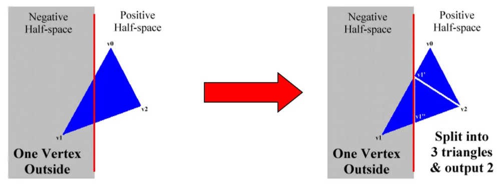
   inside vertex 0개: trivially reject 이므로 다 날림

   inside vertex 1개:
   - 바깥쪽 점들과 연결해서 만나는 점 v_0', v_1' 을 구하고 $\Delta v'_0 v'_1 v_2$ 을 새로운 output 으로 출력



inside vertex 2개:

- 바깥쪽 점들과 연결해서 만나는 점 v_1', v_1'' 을 구하면, 사각형 $v_0v_1'v_1''v_2$ 가 나옴
  - → output 도 삼각형으로 만들어주기 위해, 사각형을 쪼개줌
  - → $\Delta v_0v'_1v_2$, $\Delta v'_1v''_1v_2$ 가 만들어짐



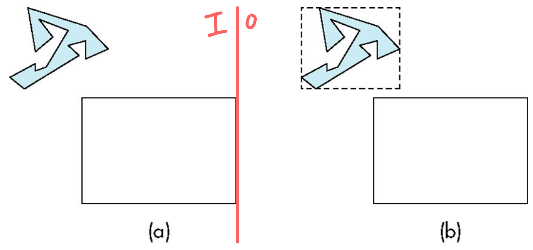
$\Delta v_0v'_1v_2$ $\Delta v'_1v''_1v_2$

Clipping Acceleration

- Bounding area(2D)/volume(3D) 에 **먼저 clipping 을 적용해 test 하는 것**
- Bounding area: 복잡한 input polygon 을 감싸는 간단한 shape (box, sphere)

💡 사용하는 이유

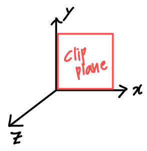
- trivial reject case 인 경우 → 복잡한 polygon 에 대해 test 하려면 시간을 많이 낭비하게 됨
- bounding area 에 먼저 clipping 을 적용해 trivial reject 한 경우면 날림
- 간단한 shape 이므로, test 가 훨씬 간단하고 시간 절약 가능



Quiz) clipping

Consider a specialized version of Sutherland-Hodgman clipping for a triangle T with three vertices v_1, v_2, v_3 against a clipping plane P_1.

- P_1: z=0 (+z is outside of the clip region)
- T: $v_1(0, 0, 1)$, $v_2(0, 0, -1)$, $v_3(-1, 0, -1)$



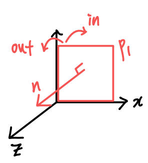
Find the normal vector of the plane P_1 that points at the positive z-axis.

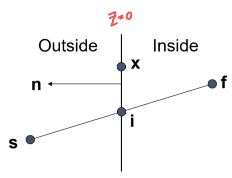
하나를 선택하세요.

- a. (0, 1, 0)
- b. (0, 0, -1)
- c. (0, 0, 1) ✔
- d. (1, 0, 0)

0x+0y+1z=0. The normal vector consists of the coefficients.

정답 : (0, 0, 1)





n = (0,0,1)

Find respectively whether each vertex v_1, v_2, v_3 is inside or outside of P_1.

하나를 선택하세요.

- a. outside, outside, inside
- b. outside, inside, inside ✔
- c. outside, outside, outside
- d. inside, outside, inside

Pick any point on z=0, say x=(0, 0, 0)

$(v_1 - x)\cdot(0,0,1) = 1 > 0 \to$ outside
$(v_2 - x)\cdot(0,0,1) = -1 < 0 \to$ inside
$(v_3 - x)\cdot(0,0,1) = -1 < 0 \to$ inside

정답 : outside, inside, inside

Inside-outside testing

$z=0$ 에서 임의 점 $x=(0,0,0)$

1) $(v_1-x)\cdot n = (0,0,1)\cdot(0,0,1) = 1 > 0$ out
2) $(v_2-x)\cdot n = (0,0,-1)\cdot(0,0,1) = -1 < 0$ in
3) $(v_3-x)\cdot n = (-1,0,-1)\cdot(0,0,1) = -1 < 0$ in

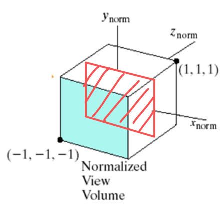
Find two intersecting points v_1' and v_3' between the three edges of the triangle T (v_1v_2, v_2v_3, v_3v_1) and the plane P_1; v_1' intersects v_1v_2 and P_1, and v_3' intersects v_3v_1 and P_1.

하나를 선택하세요.

- a. $v_1' = (0,0,0), v_3' = (-\frac{1}{2},0,0)$ ✔
- b. $v_1' = (0,1,0), v_3' = (\frac{1}{2},0,0)$
- c. $v_1' = (0,0,1), v_3' = (-\frac{1}{2},0,0)$
- d. $v_1' = (0,0,0), v_3' = (0,0,-\frac{1}{2})$

정답 : $v_1' = (0,0,0), v_3' = (-\frac{1}{2},0,0)$

Finding Intersection points



1. $z=0$
2. $z_1+(z_2-z_1)t=0$, $t=\frac{-z_1}{z_2-z_1}$
3. $\left(x_1+\frac{x_2-x_1}{z_2-z_1}(-z_1),\ y_1+\frac{y_2-y_1}{z_2-z_1}(-z_1),\ 0\right)$

1) V_1V_2 : $(0,0,1)$ $(0,0,-1)$

$V_1' = \left(0+\frac{0-0}{-1-1}(-1),\ 0+\frac{0-0}{-1-1}(-1),\ 0\right)$

$= (0,0,0)$

2) V_3V_1 : $(-1,0,-1)$, $(0,0,1)$

$V_3' = \left(-1+\frac{0-(-1)}{-1-1}(-1),\ 0+\frac{0-0}{-1-1}(-1),\ 0\right)$

$= (-\frac{1}{2}, 0, 0)$

Find the clipped triangle(s) of T as a result of the Sutherland Hodgman algorithm with respect to the clip plane P_1.

하나를 선택하세요.

- a. $\triangle(v_3', v_3, v_2), \triangle(v_1, v_3', v_2)$
- b. $\triangle(v_1, v_2, v_3)$
- c. $\triangle(v_3', v_3, v_2), \triangle(v_1', v_3', v_2)$ ✔
- d. Empty

정답 : $\triangle(v_3', v_3, v_2), \triangle(v_1', v_3', v_2)$

1개 inside 가 2개인 Triangle 이면 → 2개 삼각형으로

$V_1 \to V_2$: out → in
$V_3 \to V_1$: in → out

결과 polygon
$V_1' = (0,0,0)$
$V_3 = (-1,0,-1)$
$V_3' = (-\frac{1}{2},0,0)$