

# 컴퓨터 그래픽스

## 4. 2차원 그래픽스의 윈도우와 뷰포트

2024년 2학기

# 학습 내용

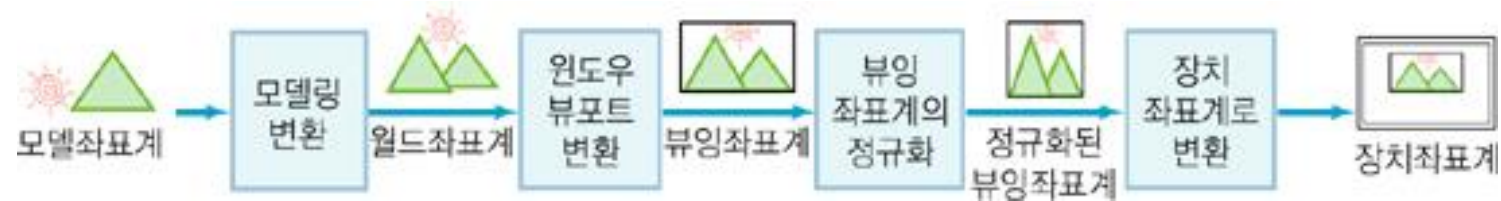
- 윈도우와 뷰포트
  - 2차원 뷰잉 파이프라인: 윈도우, 뷰포트
  - 클리핑 알고리즘

# 윈도우와 뷰포트

## • 2차원 그래픽스의 뷰잉 파이프라인

- 뷰잉 파이프라인: 그림을 출력장치에 나타내야 할 때, 특정 부분만을 출력시키기도 하는데 이때 적절한 변환을 적용해야하는데 이 과정을 뷰잉 파이프라인이라고 한다.

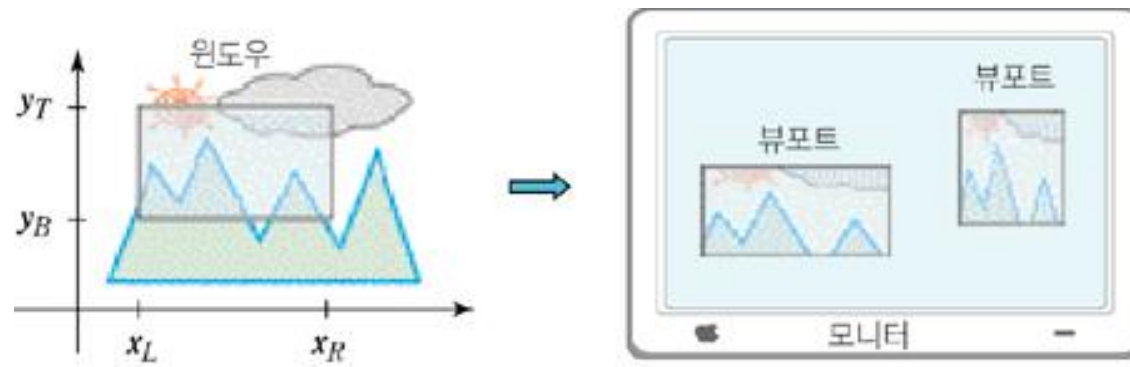
- 뷰잉 파이프라인 과정:



- 모델 좌표계: 개별 객체를 표현하기 위해 사용되는 좌표계
- 월드 좌표계: 각 모델 좌표계의 통합된 좌표계
- 뷰잉 좌표계: 출력장치에 출력 위치 및 크기 설정하여 뷰포트에 출력, 뷰포트 좌표계
- 정규 좌표계: 정규화된 좌표계
- 장치 좌표계: 출력하려는 장치 좌표계

# 윈도우와 뷰포트

- **Window**
  - 출력 장치에 표시하기 위해 선택된 세계 좌표 영역
- **Viewport**
  - 윈도우가 사상되는 출력 장치의 영역
- **윈도우-뷰포트 변환에 의한 효과**
  - Zooming 효과: zoom-in 또는 zoom-out 효과
  - Panning 효과: 카메라 각도를 돌려가면서 비디오 촬영하는 것과 같은 효과
  - Multi viewport 효과: 한번에 여러 개의 화면을 가질 수 있다.



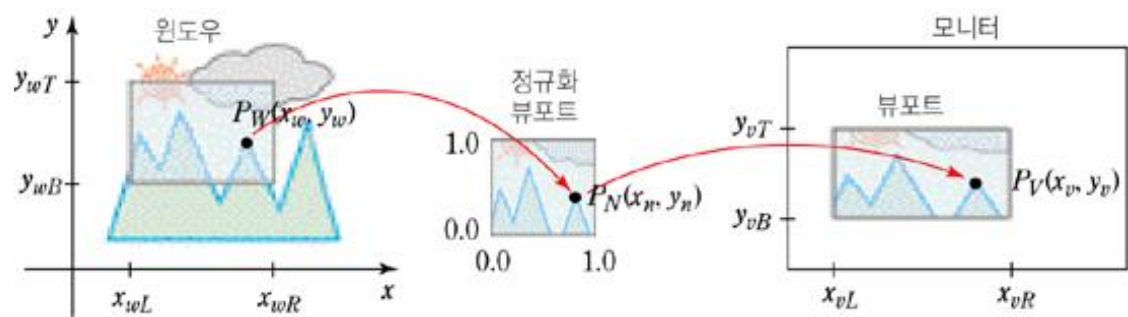
# 원도우와 뷰포트

- 원도우-뷰포트 좌표 변환
  - $(x_w, y_w)$ : 원도우 내의 점       $(x_v, y_v)$ : 뷰포트 안의 점
  - 행렬

- $x_v = x_{vL} + (x_w - x_{wL})s_x,$
- $y_v = y_{vB} + (y_w - y_{wB})s_y,$

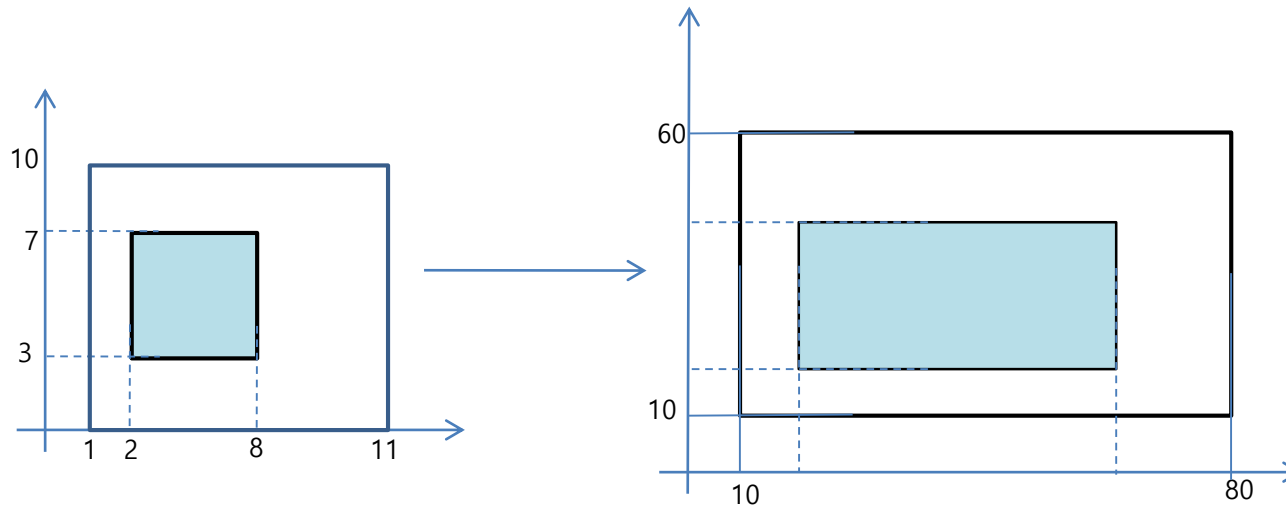
$$s_x = \frac{(x_{vR} - x_{vL})}{(x_{wR} - x_{wL})}$$
$$s_y = \frac{(y_{vT} - y_{vB})}{(y_{wT} - y_{wB})}$$

$x_{wR}, x_{wL}$ : 원도우의 x방향 최대값, 최소값  
 $y_{wT}, y_{wB}$ : 원도우의 y방향 최대값, 최소값



# 윈도우와 뷰포트

- 예) 다음의 도형에 대하여 윈도우-뷰포트 변환이 주어졌을 때 변환 좌표 값
  - 원도우 (1, 0) (11,10) → 뷰포트 (10, 10) (80, 60)
  - 도형 좌표: (2, 3) (8, 7)로 이루어진 사각형이 윈도우-뷰포트 변환 후 좌표값:



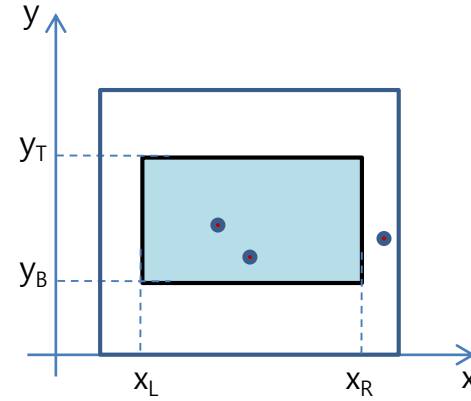
# Clipping (클리핑)

- Clipping이란

- 윈도우-뷰포트 변환 시, 출력장치에 표시되어서는 안될 그림영역을 제거한 뒤, 나머지 그림영역을 출력화면에 나타내는 것
- 월드 좌표 클리핑:
  - 윈도우를 설정할 때 윈도우 바깥 영역을 제거하여 윈도우 내부 영역만 뷰포트로 매핑시키는 방법
- 뷰포트 클리핑:
  - 월드 좌표계를 표현된 그림 전부를 뷰포트로 매핑시킨 후 뷰포트 외부에 위치한 객체나 그림의 일부를 제거하는 방법
- 두 클리핑이 모두 결과는 같다.

# Clipping (클리핑): 점

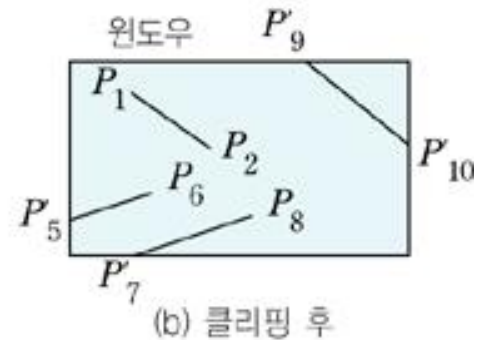
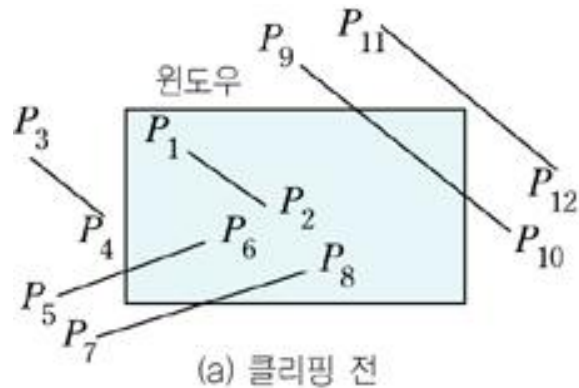
- 점 클리핑
    - 클리핑 되는 객체가 점
    - 한 점  $P(x, y)$ 는
      - $x_L \leq x \leq x_R, \quad y_B \leq y \leq y_T$
- 이면 그려진다.





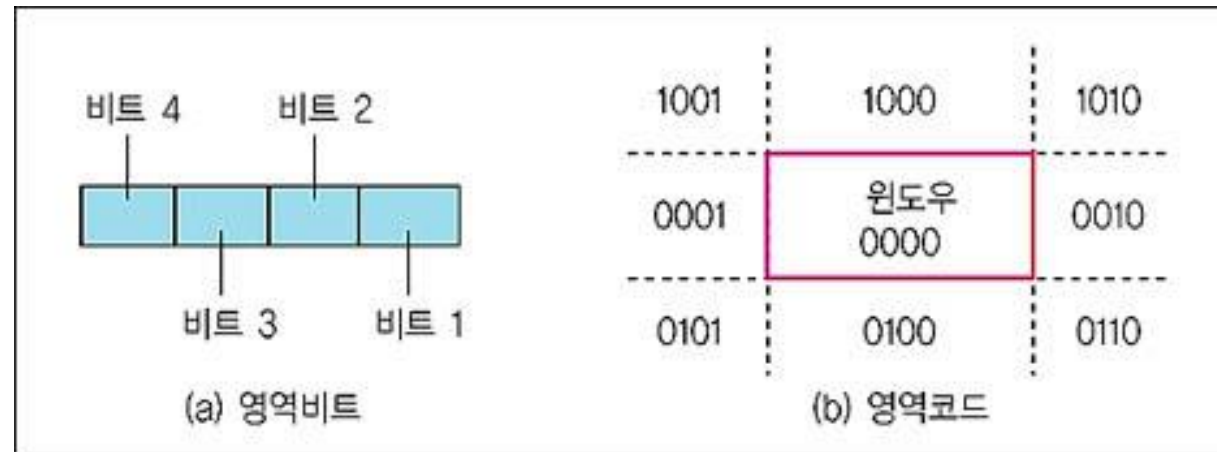
# Clipping (클리핑): 선

- 선 클리핑
  - 클리핑 되는 객체가 선분
  - 선분에 대하여
    - 선분이 클리핑 영역의 내부 또는 외부에 완전히 포함되는가/포함되지 않는가
    - 부분적으로 속하는가
    - 속한다면 교차점은 어떻게 구하는가



# Clipping (클리핑): 선 - Cohen-Sutherland 알고리즘

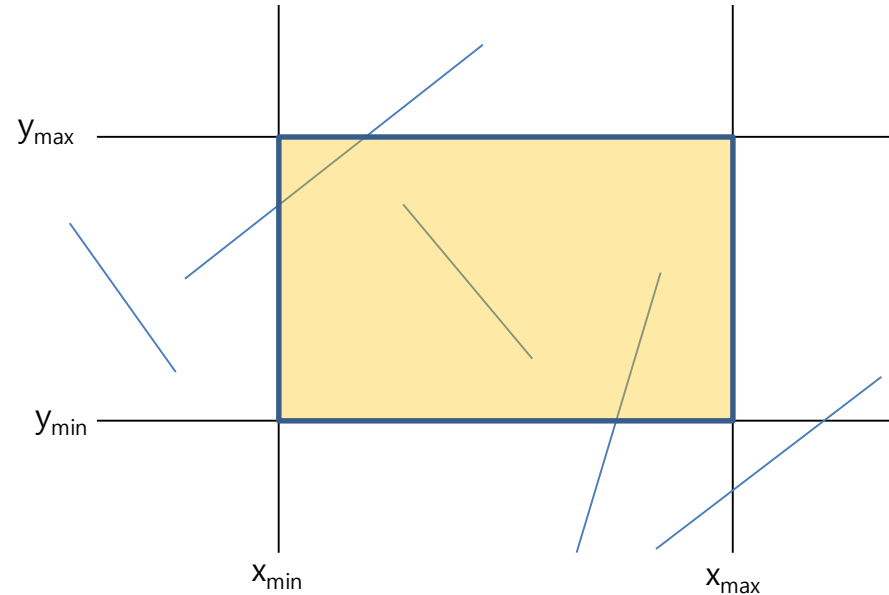
- Cohen-Sutherland 알고리즘
  - 윈도우를 중심으로 전체 그림 영역을 9개 영역으로 구분
  - 각 영역에 4비트를 사용하여 영역코드를 부여한다.
    - 비트 1: 윈도우의 왼쪽에 있으면 1
    - 비트 2: 윈도우의 오른쪽에 있으면 1
    - 비트 3: 윈도우의 아래쪽에 있으면 1
    - 비트 4: 윈도우의 위쪽에 있으면 1



# Clipping (클리핑): 선 - Cohen-Sutherland 알고리즘

## - 알고리즘 수행 과정

- ① 양 끝점의 코드가 모두 0000이면 →
- ② 양 끝점의 코드 중 한 쪽 코드는 0이고 다른 쪽 코드는 0이 아니면 →
- ③ 양 끝점 코드가 모두 0이 아니고, 양 끝점 코드간 AND 연산이 0이 아니면 →
- ④ 양 끝점 코드가 모두 0이 아니고, 양 끝점 코드간 AND 연산이 0이면 →



# Clipping (클리핑): 선 - Cohen-Sutherland 알고리즘

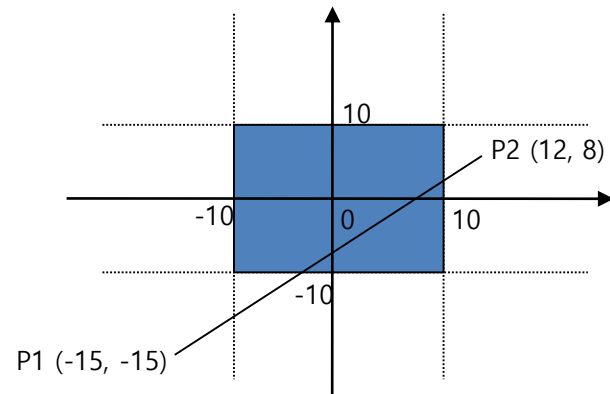
- 주어진 선분에 대한 교차점 구하기

- 수직 경계:  $x = x_L$  또는  $x = x_R$   
 $y = y_1 + m(x - x_1), \quad m = \frac{y_2 - y_1}{x_2 - x_1}$

- 수평 경계:  $y = y_B$  또는  $y = y_T$   
 $x = x_1 + \frac{(y - y_1)}{m}, \quad m = \frac{y_2 - y_1}{x_2 - x_1}$

$x_1, y_1$ : 선분의 끝 점  
 $x_L, x_R, y_B, y_T$ : 윈도우의 경계

- 예)



# Clipping (클리핑): 선 - Liang-barskey 알고리즘

- 매개변수 방정식을 이용하여 선분을 윈도우 경계에 대하여 자르는 알고리즘

- 선을 나타내는 매개변수 방정식은

- $$p(u) = (1-u)p_1 + up_2$$
$$= p_1 + (p_2 - p_1)u \quad 0 \leq u \leq 1$$

- 즉,  $x = x_1 + (x_2 - x_1)u$

- $y = y_1 + (y_2 - y_1)u$

- 끝점이  $P1 = (x_1, y_1)$   $P2 = (x_2, y_2)$  인 선분일 때

- 매개변수 방정식 사용하여 임의의 점  $P(x, y)$  을 표시

- $x = x_1 + (x_2 - x_1)u \quad 0 \leq u \leq 1 \quad (x_2 - x_1 \rightarrow d_x)$

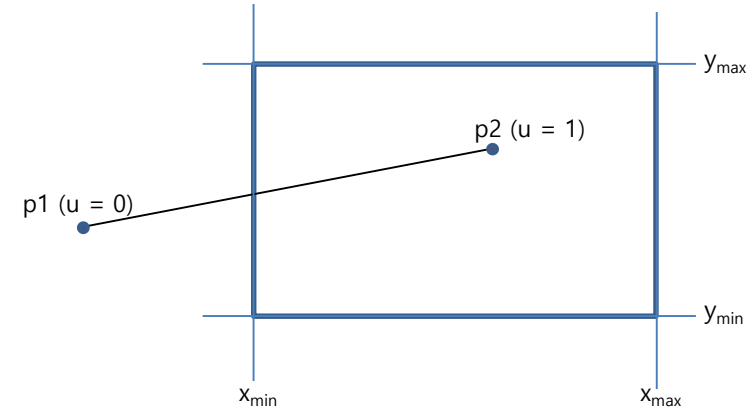
- $y = y_1 + (y_2 - y_1)u \quad 0 \leq u \leq 1 \quad (y_2 - y_1 \rightarrow d_y)$

- $u = 0 \rightarrow x = x_1, y = y_1$

- $u = 1 \rightarrow x = x_2, y = y_2$

- 선분 위에 있는 모든 점들은 아래의 조건을 만족

- $x_{\min} \leq x \leq x_{\max}, \quad y_{\min} \leq y \leq y_{\max}$



# Clipping (클리핑): 선 - Liang-barskey 알고리즘

- 매개 변수 방정식으로 다시 작성하면

$$\begin{aligned} \blacksquare x_{\min} &\leq x_1 + d_x u \leq x_{\max} & \text{--- ①} & (d_x = x_2 - x_1) \\ \blacksquare y_{\min} &\leq y_1 + d_y u \leq y_{\max} & \text{--- ②} & (d_y = y_2 - y_1) \end{aligned}$$

- ① 은 다음과 같다

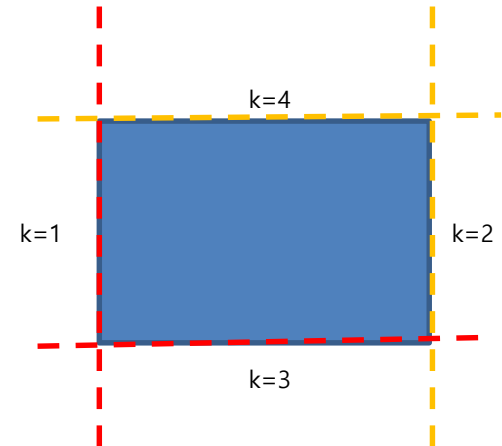
$$\begin{aligned} \gg \text{왼쪽 가장자리에 대하여} & -d_x u < x_1 - x_{\min} \\ \gg \text{오른쪽 가장자리에 대하여} & d_x u < x_{\max} - x_1 \end{aligned}$$

- ② 도 같은 방식으로 바꿀 수 있다.

$$\begin{aligned} \gg \text{아래쪽 가장자리에 대하여} & -d_y u < y_1 - y_{\min} \\ \gg \text{위쪽 가장자리에 대하여} & d_y u < y_{\max} - y_1 \end{aligned}$$

- 위의 식은,  $up_k < q_k$   $k = 1, 2, 3, 4$

$$\begin{aligned} - p_1 &= -d_x & q_1 &= x_1 - x_{\min} & \rightarrow \text{left} \\ - p_2 &= d_x & q_2 &= x_{\max} - x_1 & \rightarrow \text{right} \\ - p_3 &= -d_y & q_3 &= y_1 - y_{\min} & \rightarrow \text{bottom} \\ - p_4 &= d_y & q_4 &= y_{\max} - y_1 & \rightarrow \text{top} \end{aligned}$$



# Clipping (클리핑): 선 - Liang-barskey 알고리즘

1)  $p_k == 0$  일 때,

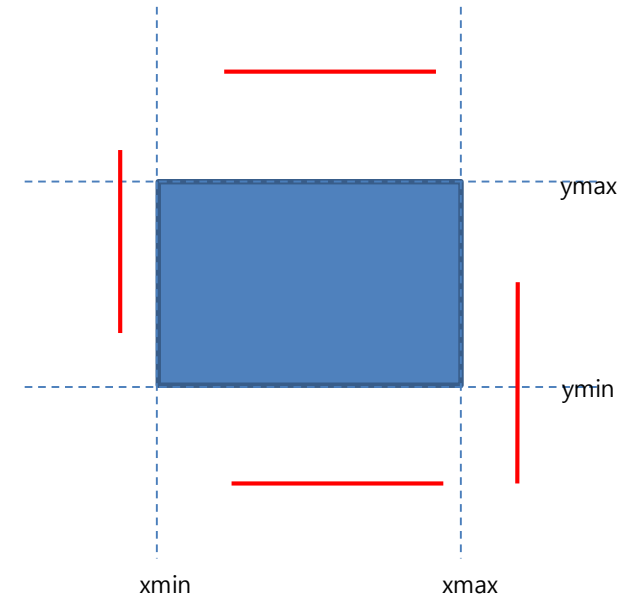
–  $p_k$ 가 0 이면,  $k$ 번째 가장자리와 평행

- $p_1 = -dx = -(x_2 - x_1) = 0 \rightarrow x_2 == x_1$
- $p_3 = -dy = -(y_2 - y_1) = 0 \rightarrow y_2 == y_1$

•  $p_k = 0$  이고,  $q_k < 0$ 이면, 그 가장자리 영역 밖에 있다.

- a.  $q_1 < 0 \rightarrow q_1 = (x_1 - x_{\min}) < 0 \rightarrow x_1 < x_{\min} \rightarrow$  영역 밖에 있다.
- b.  $q_2 < 0 \rightarrow q_2 = (x_{\max} - x_1) < 0 \rightarrow x_{\max} < x_1 \rightarrow$  영역 밖에 있다.
- c.  $q_3 < 0 \rightarrow q_3 = (y_1 - y_{\min}) < 0 \rightarrow y_1 < y_{\min} \rightarrow$  영역 밖에 있다.
- d.  $q_4 < 0 \rightarrow q_4 = (y_{\max} - y_1) < 0 \rightarrow y_{\max} < y_1 \rightarrow$  영역 밖에 있다.

$\rightarrow p_k == 0$  이고  $q_k < 0$  이면, 영역 밖에 있으므로 안 그린다.



# Clipping (클리핑): 선 - Liang-barskey 알고리즘

1)  $p_k == 0$  일 때,

–  $p_k$ 가 0 이면,  $k$ 번째 가장자리와 평행

- $p_1 = -dx = -(x_2 - x_1) = 0 \rightarrow x_2 == x_1$

- $p_3 = -dy = -(y_2 - y_1) = 0 \rightarrow y_2 == y_1$

- $p_k = 0$  이고,  $q_k > 0$  이면, 평행한 가장자리외의 가장자리와 만날 수 있다.

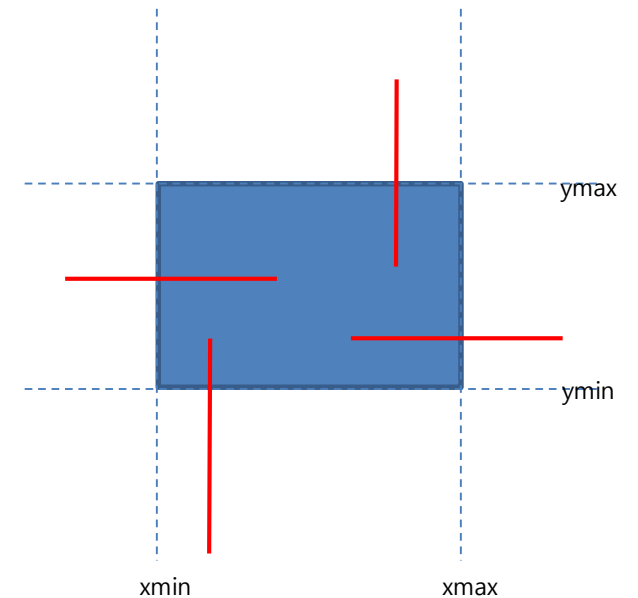
- a.  $q_1 > 0 \rightarrow q_1 = (x_1 - x_{\min}) > 0 \rightarrow x_1 > x_{\min}$

- b.  $q_2 > 0 \rightarrow q_2 = (x_{\max} - x_1) > 0 \rightarrow x_{\max} > x_1$

- c.  $q_3 > 0 \rightarrow q_3 = (y_1 - y_{\min}) > 0 \rightarrow y_1 > y_{\min}$

- d.  $q_4 > 0 \rightarrow q_4 = (y_{\max} - y_1) > 0 \rightarrow y_{\max} > y_1$

$\rightarrow p_k == 0$  이고  $q_k > 0$  이면, 평행한 가장자리외의 가장자리와 만날 수 있다.





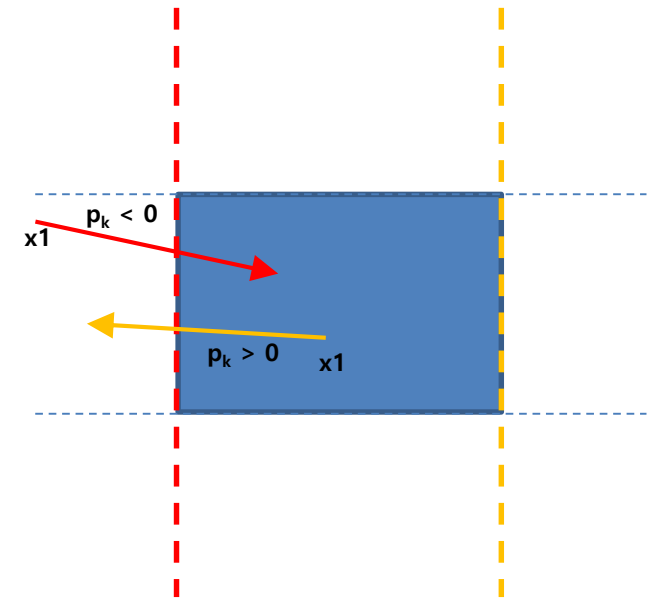
# Clipping (클리핑): 선 - Liang-barskey 알고리즘

2)  $p_k \neq 0$  일 때,

–  $p_k \neq 0$ 이면, 선분이 경계선 중 하나와 평행하지 않다.

→ 그 선분의 무한한 연장선은 원도우의 네 개의 경계선과 어디에선가 교차한다.

- $p_k < 0$  이면,
  - »  $p_1 < 0 \rightarrow p_1 = -dx = -(x_2 - x_1) < 0 \rightarrow x_2 - x_1 > 0 \rightarrow x_2 > x_1$ 
    - $p_1 < 0$  이면  $p_2 = dx \rightarrow p_2 > 0$
  - »  $p_3 < 0 \rightarrow p_3 = -dy = -(y_2 - y_1) < 0 \rightarrow y_2 - y_1 > 0 \rightarrow y_2 > y_1$ 
    - $p_3 < 0$  이면  $p_4 = dy \rightarrow p_4 > 0$
- $p_k > 0$  이면,
  - »  $p_1 > 0 \rightarrow p_1 = -dx = -(x_2 - x_1) > 0 \rightarrow x_2 - x_1 < 0 \rightarrow x_2 < x_1$ 
    - $p_1 > 0$  이면  $p_2 = dx \rightarrow p_2 < 0$
  - »  $p_3 > 0 \rightarrow p_3 = -dy = -(y_2 - y_1) > 0 \rightarrow y_2 - y_1 < 0 \rightarrow y_2 < y_1$ 
    - $p_3 > 0$  이면  $p_4 = dy \rightarrow p_4 < 0$



# Clipping (클리핑): 선 - Liang-barskey 알고리즘

2)  $p_k \neq 0$  일 때,

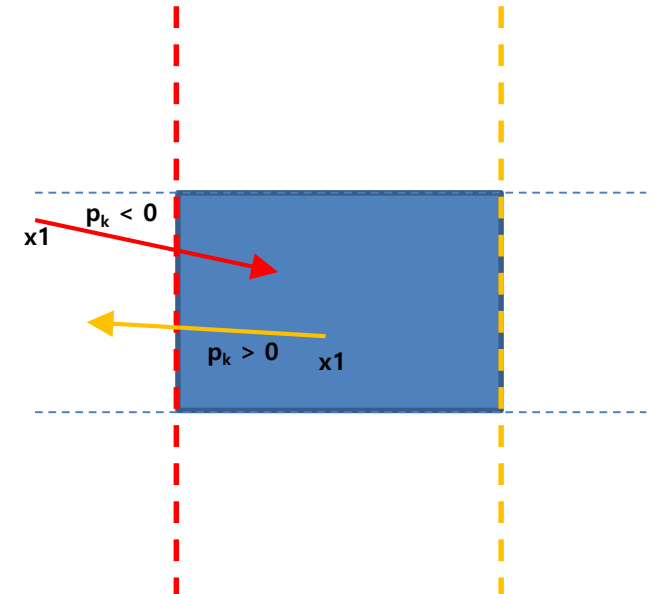
- 만약  $p_k < 0$ 이면, 직선은 밖  $\rightarrow$  안으로 진행
- 만약  $p_k > 0$ 이면, 직선은 안  $\rightarrow$  밖으로 진행

• 0이 아닌  $p_k$ 에 대하여, 매개변수  $u$ 의 값으로 가장자리와의 교차점을 찾을 수 있다. 즉,

- $u_k = q_k / p_k$  ( $k = 1, 2, 3, 4$ )
  - $k$ 에 대한  $u$ 의 값에 대하여,
    - »  $p_k < 0$ 이면,  $u_{\text{start}} = \max(u_k, 0)$
    - »  $p_k > 0$ 이면,  $u_{\text{end}} = \min(u_k, 1)$

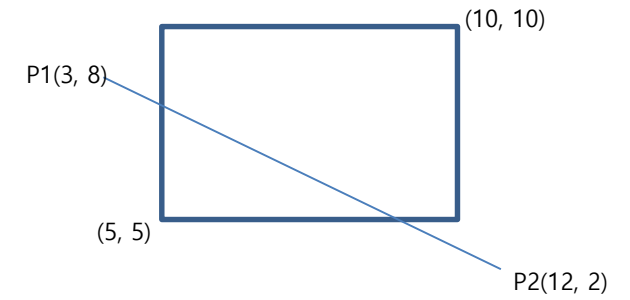
• 가장자리와의 교차점인 새로운 매개변수  $u_{\text{start}}$ 와  $u_{\text{end}}$ 에 대해서,

- if  $u_{\text{start}} > u_{\text{end}}$   $\rightarrow$  reject
- if  $u_{\text{start}} < u_{\text{end}}$   $\rightarrow$  새로운 좌표값
  - »  $\text{new\_x1} = x1 + u_{\text{start}} * dx,$        $\text{new\_y1} = y1 + u_{\text{start}} * dy$
  - »  $\text{new\_x2} = x1 + u_{\text{end}} * dx,$        $\text{new\_y2} = y1 + u_{\text{end}} * dy$



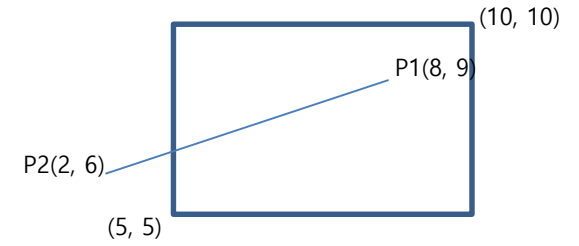
# Clipping (클리핑): 선 - Liang-barskey 알고리즘

- 예) 윈도우 영역 (5, 5) (10, 10),  $p1=(3, 8)$   $p2=(12, 2)$



# Clipping (클리핑): 선 - Liang-barskey 알고리즘

- 예) 윈도우 영역 (5, 5) (10, 10),  $p1=(8, 9)$   $p2=(2, 6)$

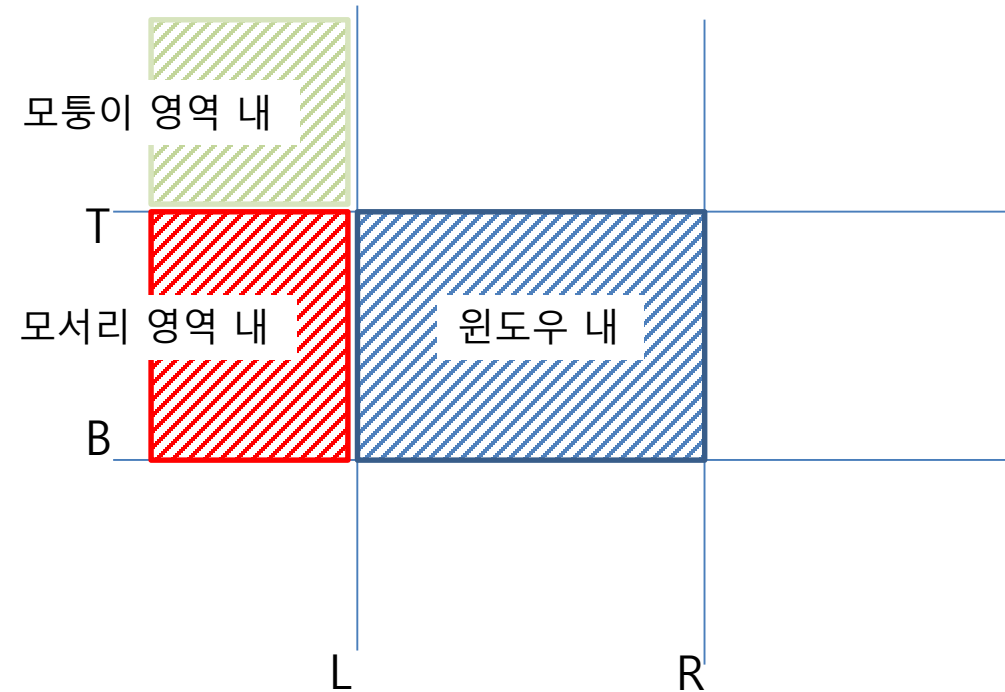


# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- 코헨 서덜랜드 알고리즘은
  - 가장자리를 한 번 이상의 만나게 되는 경우가 발생한다.
  - 이 경우를 피할 수 있는 알고리즘 → Nicholl-Lee-Nicholl (NLN) 알고리즘
- Nicholl-Lee-Nicholl (NLN) 알고리즘
  - 클리핑 윈도우 주위에 더 많은 영역을 만든다.
  - 교차점 위치 계산 전에 영역 검사를 한다 → 다중 교차점 연산을 하지 않는다.
  - 코헨 서덜랜드 알고리즘보다 더 적은 나눗셈을 하여 효과적이다.
  - 그러나 3차원으로는 확장될 수 없다.

# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

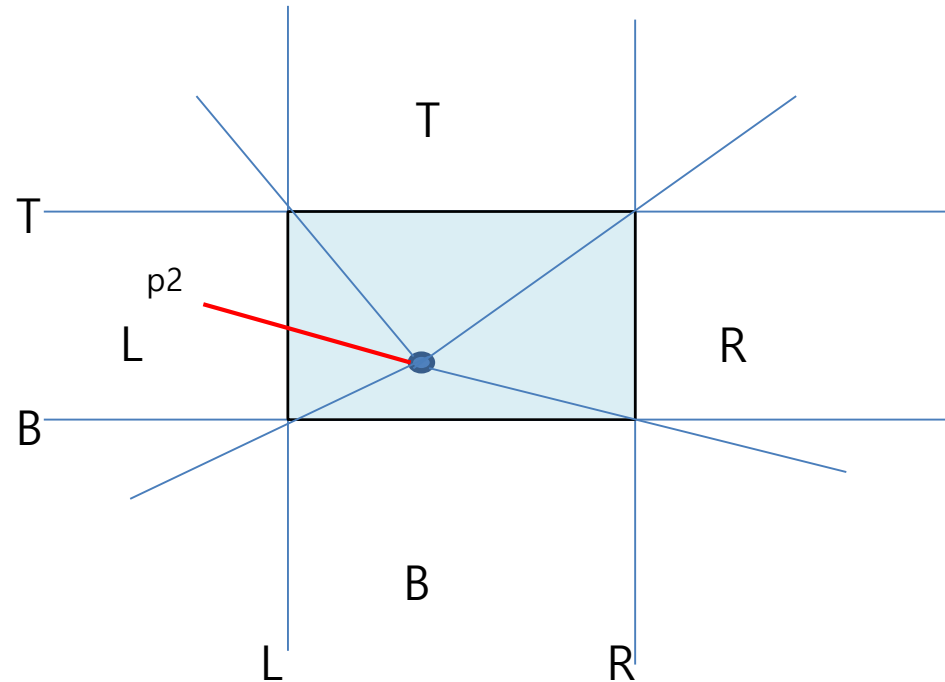
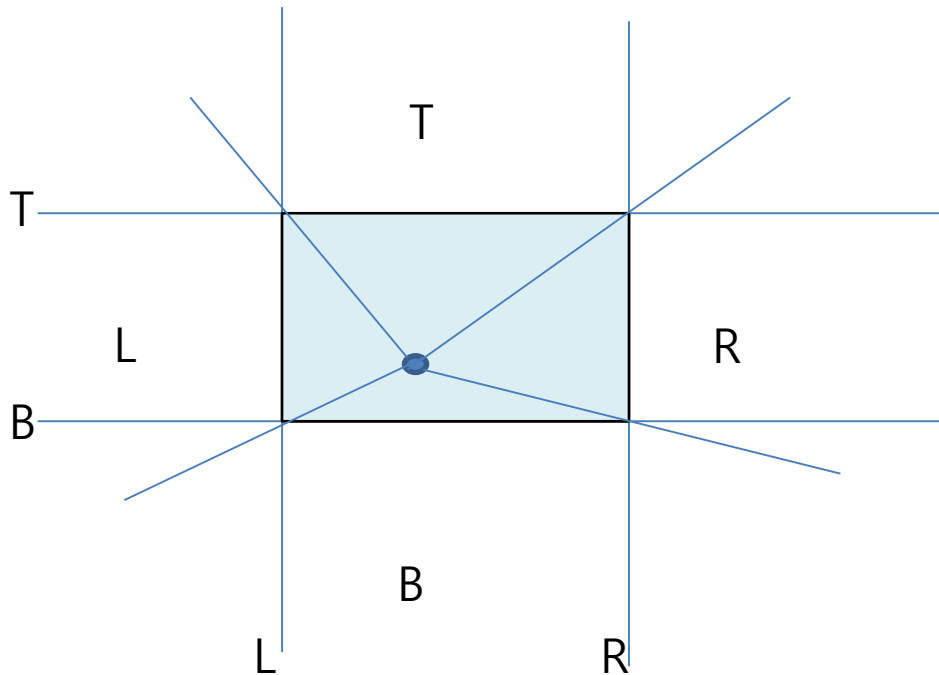
- NLN 알고리즘에서
  - 세 개의 영역에 대하여 고려한다
    - 윈도우 내
    - 모서리 영역 내
    - 모퉁이 영역 내
  - 이외의 영역은 변환을 적용한다.



- P1에 대한 P2의 상대적 위치 결정 → p1을 고려하여 평면상의 새로운 영역을 생성한다.
  - 새로운 영역의 경계: P1에서 시작해서 윈도우의 모서리를 지나가는 반 무한 직선

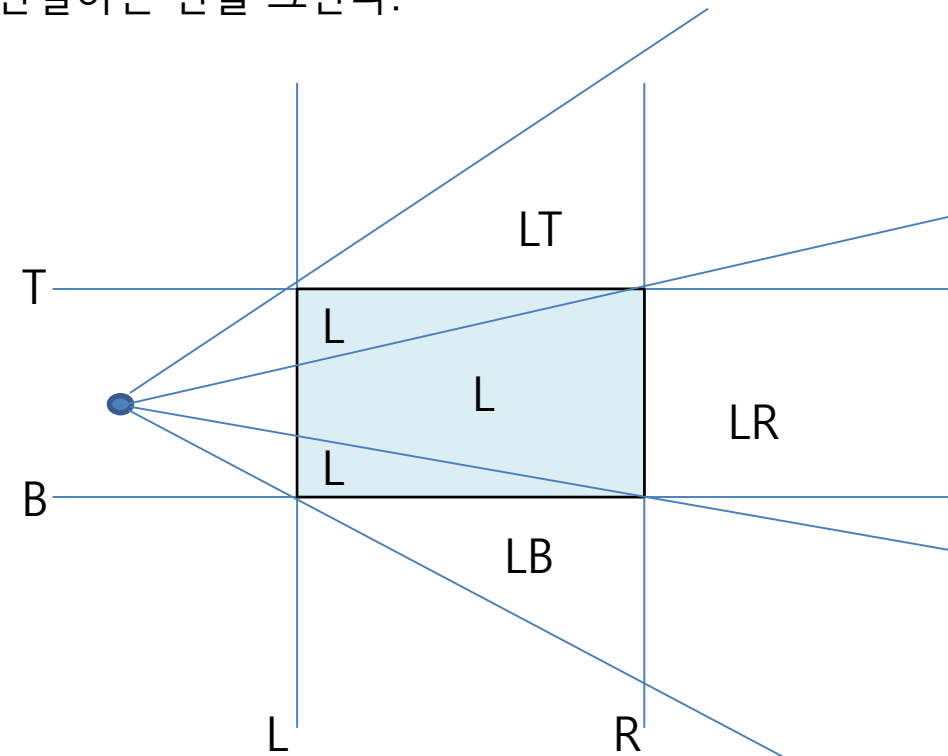
# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- 두 점 P1, P2가 모두 윈도우 내에 있을 때,
  - 두 점을 연결한 선을 그대로 그린다.
- P1이 내부에, P2가 외부에 있으면
  - P1과 네 모퉁이 점을 연결하여 영역을 (L, B, R, T) 정한다.
  - 4개의 영역 중에서 P2를 포함하는 한 영역에 대응하는 윈도우 경계와의 교차점을 결정하여 클리핑 한다.



# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

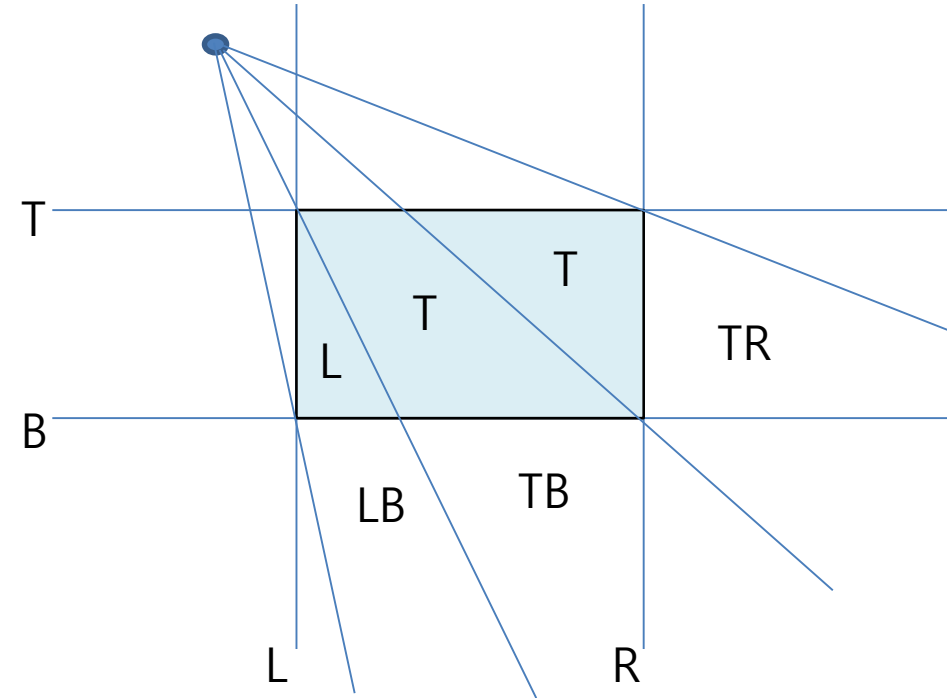
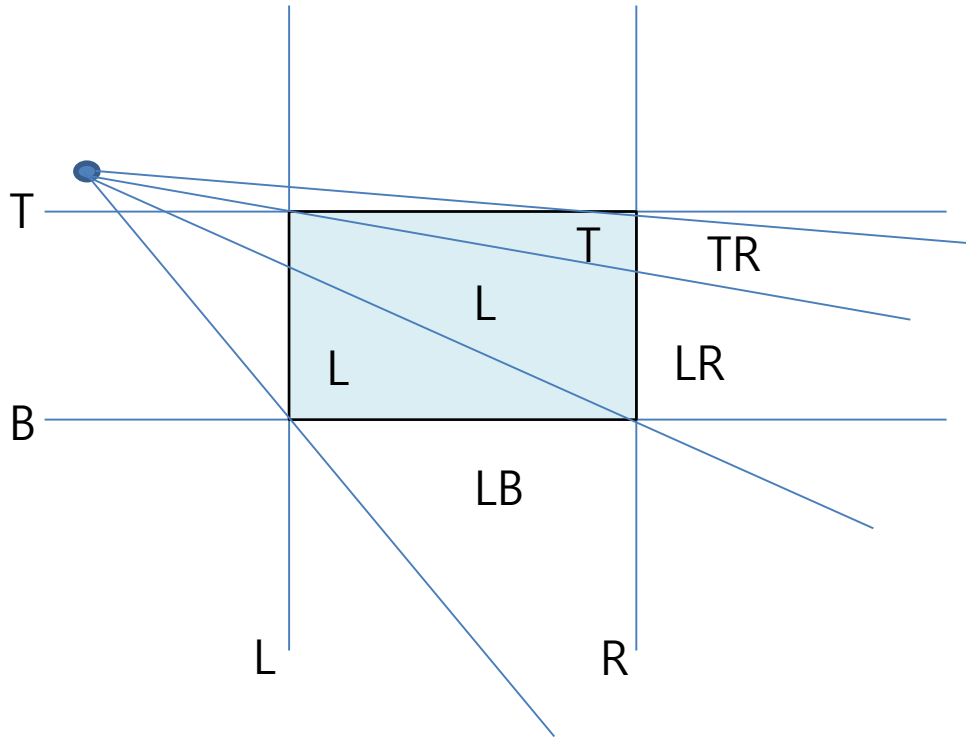
- P1이 외부에 있는데, 윈도우의 왼쪽 영역에 있으면
  - P1과 네 귀퉁이를 연결하여 L, LT, LR, LB 네 영역을 설정
  - 네 개의 영역 중, P2가 속한 영역에서 클리핑 되는 경계를 결정한다.
    - P2가 L 영역에 속한다면, 왼쪽 경계선과 P2를 연결하는 선을 그린다.
    - P2가 LT 영역에 속한다면, 왼쪽 경계선 ~ 위쪽 경계선을 연결하는 선을 그린다.
    - P2가 LR 영역에 속한다면, 왼쪽 경계선 ~ 오른쪽 경계선을 연결하는 선을 그린다.
    - P2가 LB 영역에 속한다면, 왼쪽 경계선 ~ 아래쪽 경계선을 연결하는 선을 그린다.
  - P2가 이 외의 영역에 있다면, 선분은 안 그린다.





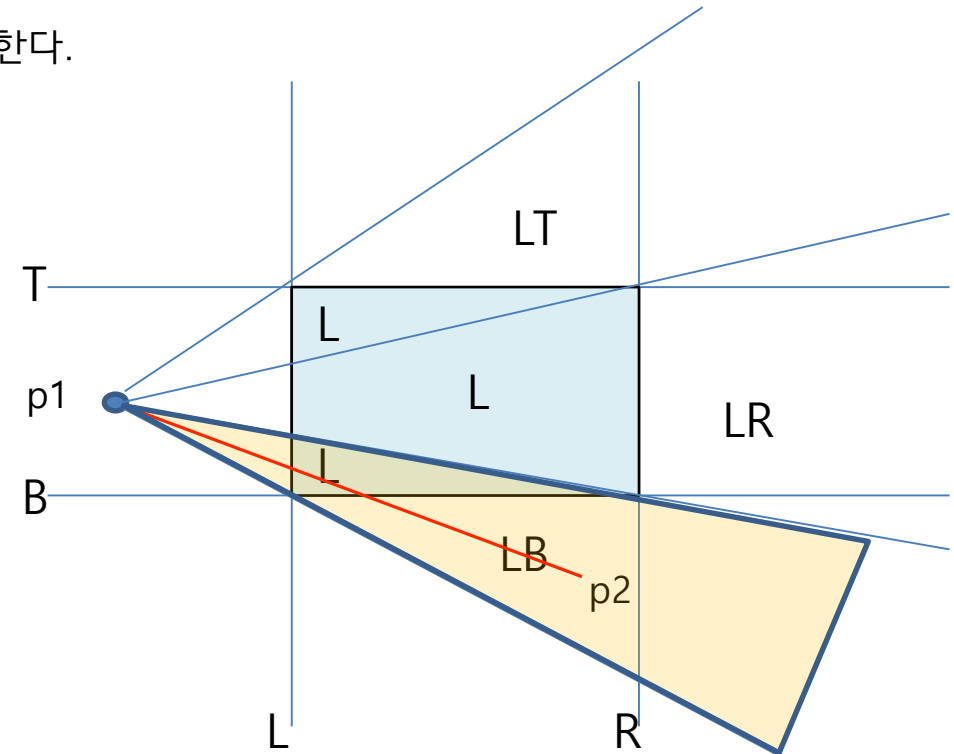
# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- P1이 외부에 있는데, P1이 윈도우의 왼쪽 위에 있을 때
  - 두 가지 가능성을 갖고 있다.
    - 영역이: T, L, TR, TB, LB 가 될 수 있다.
    - P2가 위의 영역 중 한 군데 있으면 경계선과 교차점을 구해서 그린다.
    - P2가 위의 영역 외에 있다면, 그 선은 그리지 않는다.



# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- P2가 놓여진 영역을 결정하기 위해서, 영역을 구성하는 두 선의 기울기를 비교한다.
  - $\text{slope boundary1} < \text{slope P1P2} < \text{slope boundary2}$ 
    - Boundary1은 영역의 첫번째 경계로 P1과 영역을 구성하는 모퉁이의 점을 연결하는 직선의 기울기
    - Boundary2는 영역의 두번째 경계로 P1과 영역을 구성하는 모퉁이의 점을 연결하는 직선의 기울기
- 예를 들어, P1이 윈도우 영역 왼쪽에 있다면
  - P1P2의 기울기가 포함된 가장 작은 영역을 구성하는 두 선을 구한다.
  - 오른쪽의 예에서는 slope LR과 slope LB
    - P2가 영역 내에 있지 않다면 → P2는 LB에 있다
    - 왼쪽 ~ 오른쪽 경계와 만나는 점을 연결한 선분을 그린다.



# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- 기울기를 체크하여 선분이 어느 영역을 지나는지 알아낸다 → 교차점을 찾아낸다.

- P2 좌표값과 모서리의 좌표값을 비교하여 알아낸다
  - 좌측/우측 경계선은 x값과 비교하고, 위/아래 경계선은 y 값과 비교한다.
- 선분이 위의 영역에 있지않으면, 그 선분은 모두 삭제한다.
- 그렇지 않으면 경계선과의 교차점을 계산한다.
  - 해당 영역의 경계선과의 교차점은 매개변수 방정식으로 찾을 수 있다.

$$x = x1 + (x2 - x1) u$$

$$y = y1 + (y2 - y1) u$$

- 좌측 경계선  $x = xL$  또는 우측 경계선  $x = xR$  일 때,  $u = \frac{(xL - x1)}{(x2 - x1)}$  또는  $u = \frac{(xR - x1)}{(x2 - x1)}$

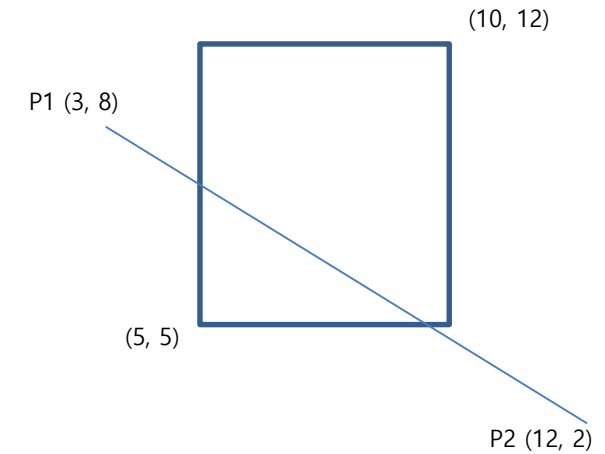
$$\rightarrow y = y1 + (xL - x1) * \frac{(y2 - y1)}{(x2 - x1)}$$

- 비슷하게, 위쪽 경계선  $y = yT$  또는 아래쪽 경계선  $y = yB$  일 때,  $u = \frac{(yT - y1)}{(y2 - y1)}$  또는  $u = \frac{(yB - y1)}{(y2 - y1)}$

$$\rightarrow x = x1 + (yT - y1) * \frac{(x2 - x1)}{(y2 - y1)}$$

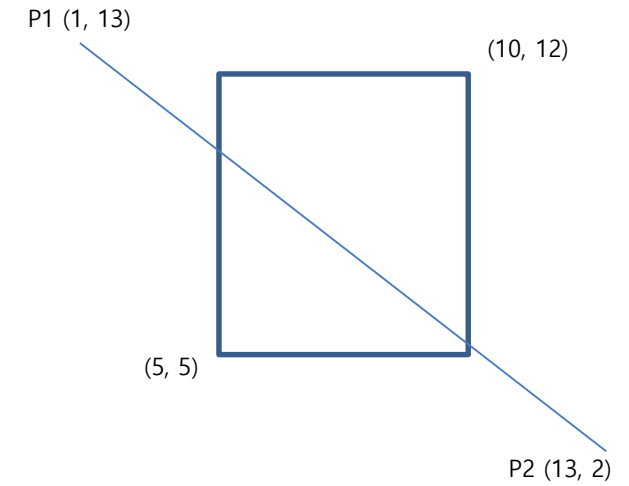
# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- 예) 윈도우 영역 (5, 5) (10, 12),  $p1=(3, 8)$   $p2=(12, 2)$



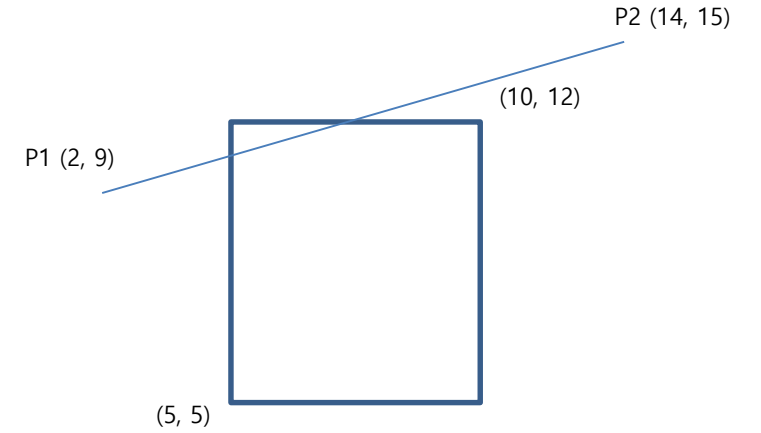
# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- 예) 윈도우 영역 (5, 5) (10, 12),  $p1=(1, 13)$   $p2=(13, 2)$



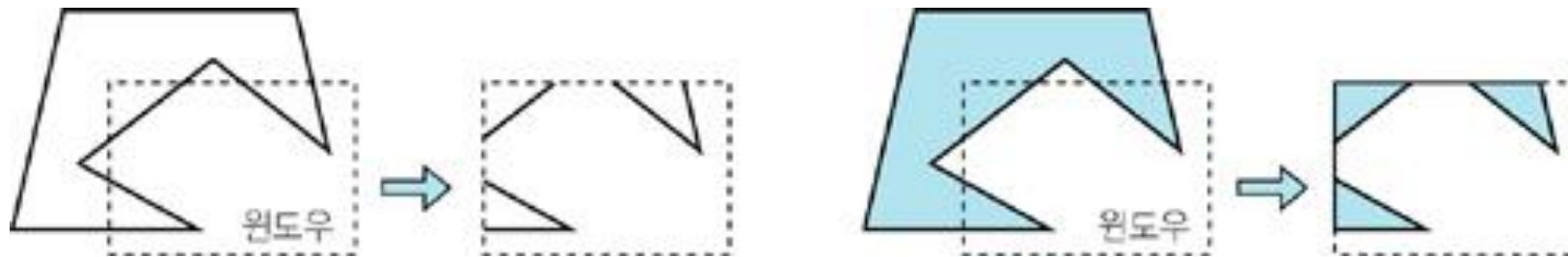
# Clipping (클리핑): 선 - Nicholl-Lee-Nicholl (NLN) 알고리즘

- 예) 윈도우 영역 (5, 5) (10, 12),  $p1=(2, 9)$   $p2=(14, 15)$



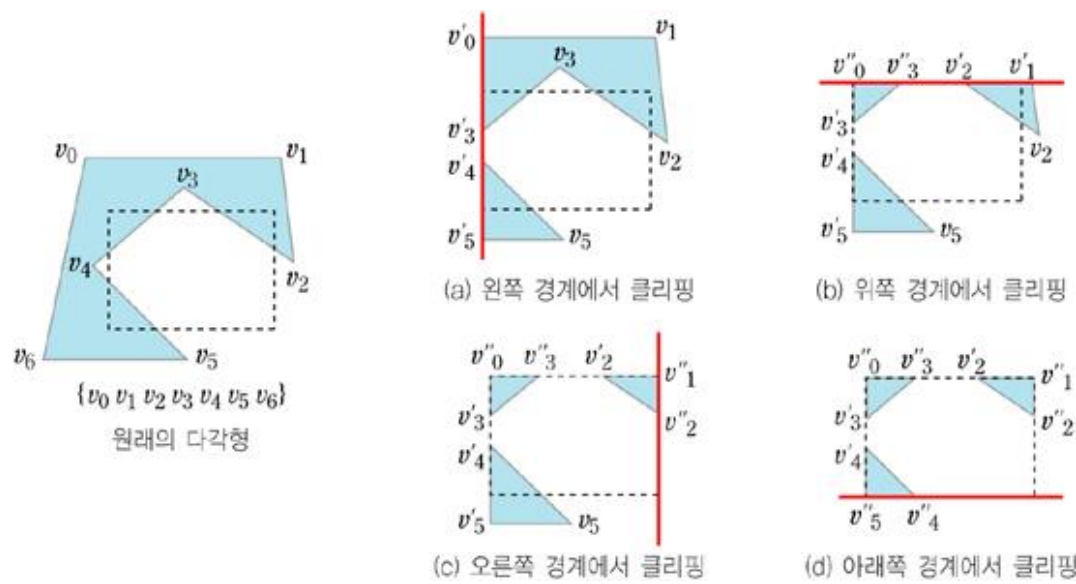
# Clipping (클리핑): 다각형 - Sutherland-Hodgeman 알고리즘

- 속이 빈 다각형(Hollow polygon) :
  - 선 클리핑 알고리즘 적용
- 속이 찬 다각형 :
  - 몇 개의 Closed filled polygon 생성



# Clipping (클리핑): 다각형 - Sutherland-Hodgeman 알고리즘

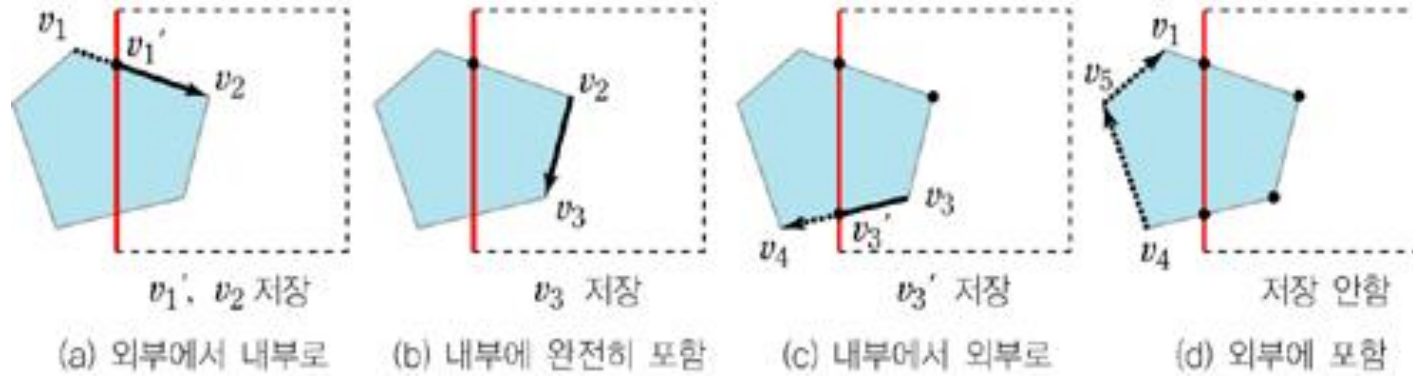
- Sutherland-Hodgeman 알고리즘
  - 다각형의 모든 꼭지점이 윈도우의 내부 또는 외부에 완전히 포함되는지를 결정하여 다각형 전체를 제거하거나 선택하고 그 외의 경우에는 다음 알고리즘을 적용하여 다각형을 클리핑
  - 한 경계변을 기준하여 이 변이 윈도 바깥쪽 영역에 속하는 다각형 부분은 클리핑 소거





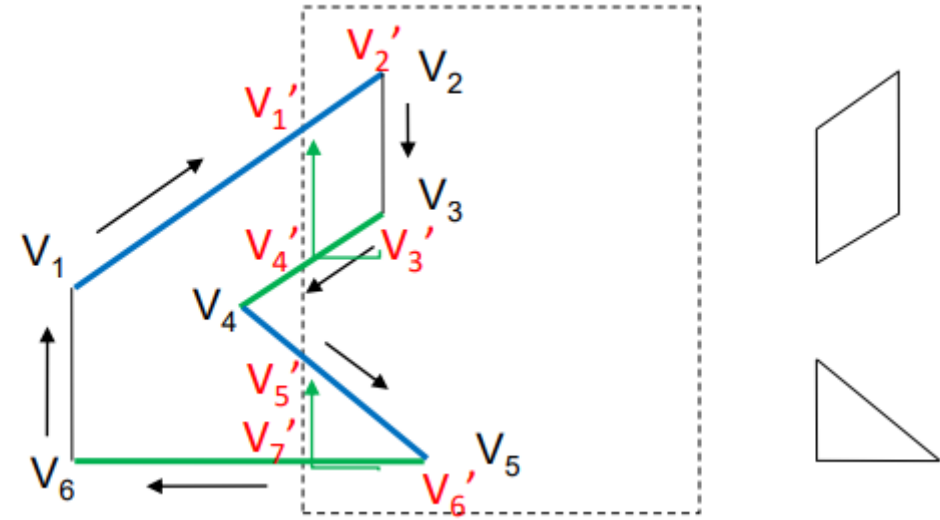
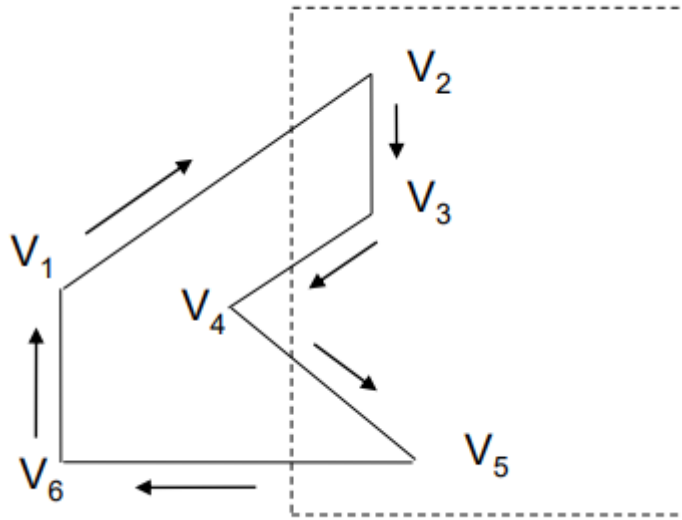
# Clipping (클리핑): 다각형

- Sutherland-Hodgeman 알고리즘
  - 각 윈도우 경계 (상하좌우)에 대하여 다음 알고리즘을 적용
  - 4가지 경우로 구분하여 다각형 꼭지점을 재구성



# Clipping (클리핑): 다각형 - Weiler-Atherton 알고리즘

- Weiler-Atherton 알고리즘
  - 오목 다각형 클리핑 시 발생할 수 있는 Sutherland-Hodgeman 알고리즘의 문제점을 해결
    - 시계방향으로 꼭지점을 따라간다
    - 외부 → 내부: 폴리곤 경계를 따라간다.
    - 내부 → 외부: 폴리곤 경계를 시계방향으로 따라간다.



## 이번 주에는

- 윈도우와 뷰포트
- 클리핑
  - 점 클리핑
  - 선 클리핑
  - 다각형 클리핑
- 다음 시간에는
  - 3차원 그래픽스의 기하변환