



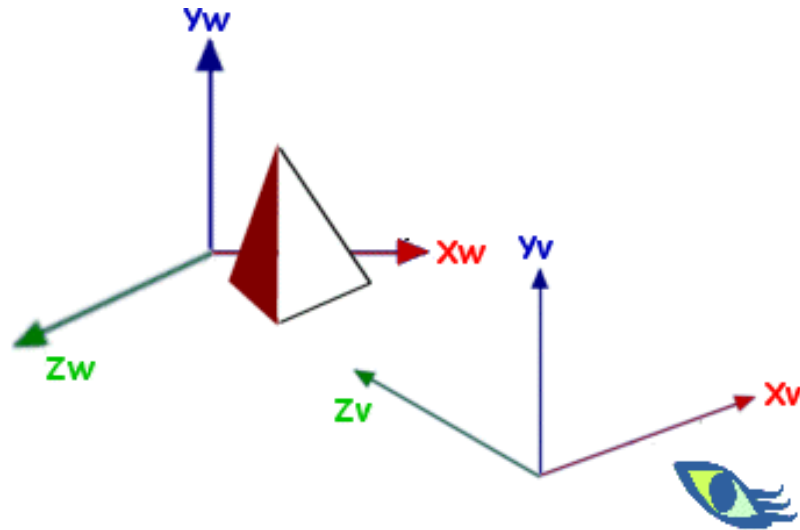
# 관측 변환 및 은면/은선 제거

# 관측 변환(Viewing Transformation)



## 1단계: 관측면(View Plane)의 지정

- 사용자는 관측면을 정의함으로써 어떤 장면의 특정 관측을 정한다.
- 관측면 : 한 물체를 투영해서 나타나게 되는 평면
  - > 마치 원하는 장면을 찍기 위해서 방향을 잡고, 자리를 정한 카메라내에 위치한 필름과 같은 것
  - > 관측평면은 관측 좌표체계(viewing coordinate system)를 정의함에 따라서 만들어짐

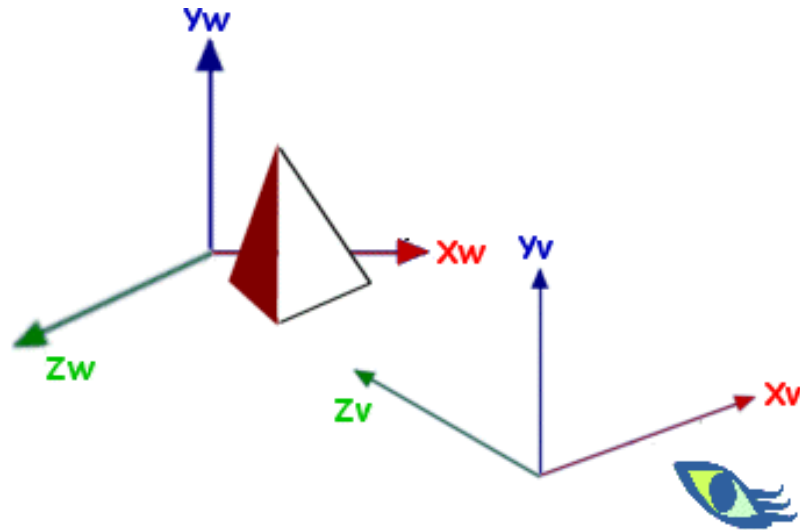


# 관측 변환(Viewing Transformation)



## 관측면(View Plane)의 지정

- 사용자는 관측면을 정의함으로써 어떤 장면의 특정 관측을 정한다.
- 관측면 : 한 물체를 투영해서 나타나게 되는 평면
  - > 마치 원하는 장면을 찍기 위해서 방향을 잡고, 자리를 정한 카메라내에 위치한 필름과 같은 것
  - > 관측평면은 관측 좌표체계(viewing coordinate system)를 정의함에 따라서 만들어짐

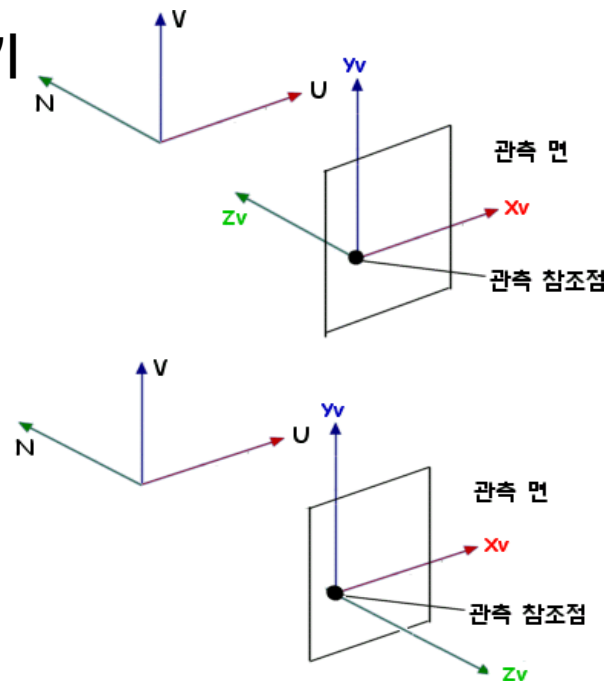


# 관측 변환(Viewing Transformation)



## 관측 좌표 체계

- 관측 좌표를 정하면 세계 좌표위치(world coordinate position)는 이에 따라 재 정의되고, 이 관측 좌표축에 따라서 상대적으로 표현된다.
- 관측참조점(view reference point) : 관측좌표를 만들기 위해서 세계좌표에서 한 점을 선정  
> 관측좌표의 원점
- 관측평면의 방향 : 관측 평면정규벡터  
(view plane normal vector)  
에 의해서 정의
- 관측 평면정규벡터 : 관측좌표체계에서 z축의 양수 방향을 표시한다.
- 관측상향벡터(view up vector)라고 하는 수직벡터  
V : y축의 방향에 해당한다.

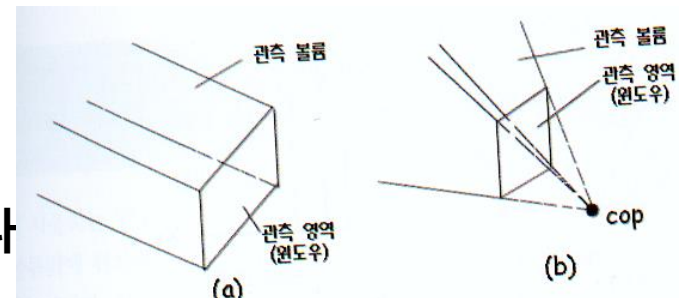
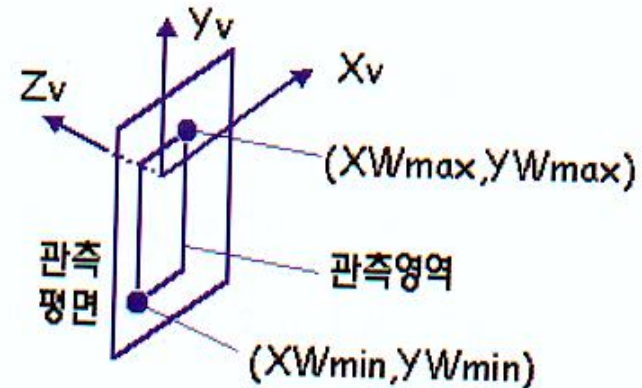


# 관측 변환(Viewing Transformation)



## 관측 볼륨(Volume)

- 카메라에서 사용되는 렌즈의 형태에 따라 얼마만한 크기의 장면이 필름에 노출되는 지가 결정되는 것처럼 3D 관측에서는 투영 관측 영역(Projection window)가 같은 효과를 나타낸다
  - > 광각렌즈에서 큰 장면을 찍을 수 있듯이, 투영 관측 영역을 조정하여 큰 화면 생성
  - > 관측 영역은 3D관측에서는 투영 관측 평면상의  $x, y$ 축에서 최소, 최대 값에 의해서 정의
- 관측 볼륨은 투영 관측 영역에 의하여 결정
  - > 평행 투영법에서는 관측 볼륨이 4면에 의해서 무한길이의 직육면체 형태로 나타나고 원근 투영법에서는 투영 중심점을 정점으로 하여 사각 추의 형태로 나타남
  - > 사각추는 관측영역에 의해서 잘려지고 잘려진 이하가 관측 볼륨이 되고 이를 frustum(각추대)라고 함



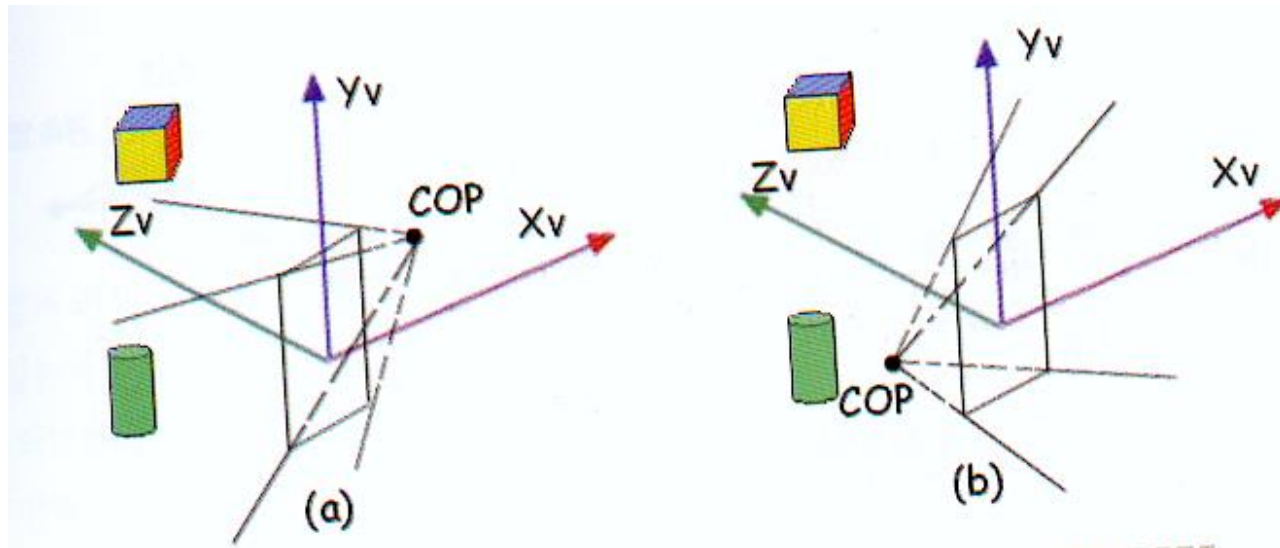
평행 투영의 관측 볼륨 원근 투영의 관측 볼륨

# 관측 변환(Viewing Transformation)



## 원근 투영에 의한 관측 볼륨

- 투영 중심점이 관측 좌표 공간에서의 임의의 점에 위치할 수 있다고 하는 경우, 아래 그림은 관측 축을 기준으로 두 개의 다른 방향을 갖는 관측 볼륨 사각 추를 보여주는 예시
- (a)의 경우는 일반적인 경우
- (b)의 경우는 투영 중심점과 대상 물체가 관측 평면을 기준으로 같은 쪽에 위치하고 있는 경우로 이때 화면에 출력되는 것은 ?

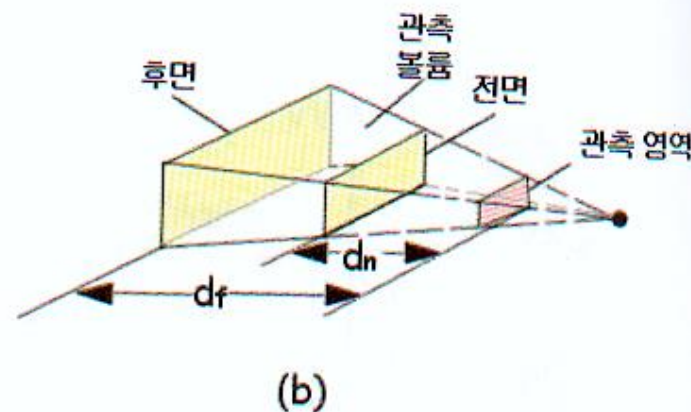
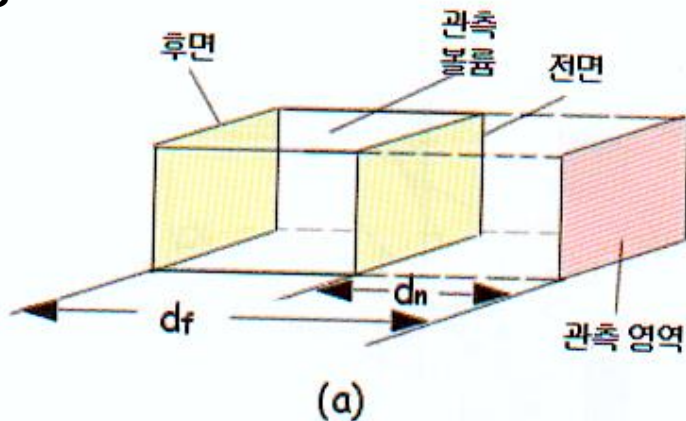


# 관측 변환(Viewing Transformation)



## 관측 볼륨: Near plane, Far plane

- 관측 볼륨을 정의하기 위해서 한 두개의 평면이 더 사용되는 경우도 있다.
- 유한 관측 볼륨: 근접평면(전면: NP)과 원접평면(후면: FP)에 의해서 6개의 면으로 구성된 관측 볼륨  
> NP, FP는 항상 관측 평면과 평행이고, 관측 좌표상의 관측 평면과의 상대거리에 의해서 결정



원근 투영시 관측 평면에서 멀리 떨어져 있는 물체는 단지 하나의 점으로 보이게 할 수도 있고, 매우 가까이 있는 물체는 다른 물체에 가려서 보이지 않게 할 수도 있다

# 관측 변환(Viewing Transformation)

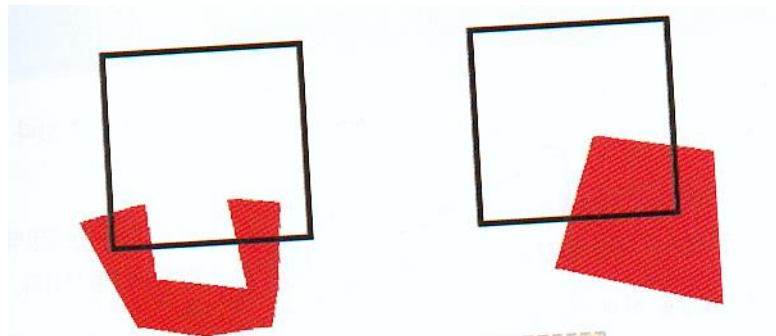
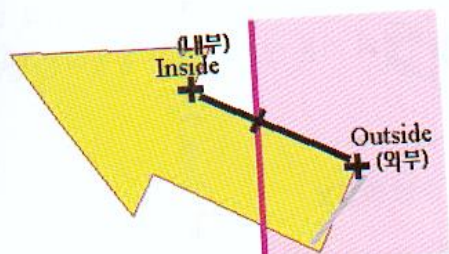


## 절단 작업 (Clipping)

- 3D절단 작업에 사용하는 알고리즘은 관측 평면에 투영되는 관측 볼륨내의 모든 선분을 알아내어 저장하는 것으로 이때 관측 볼륨밖에 위치하는 모든 선분은 삭제
- 2D 선분 절단 과정의 확장

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

선분 끝점의 좌표 값을 다각면 평면공식에  
대입하여 선분의 내, 외부를 판단하고  
외부에 있는 선분 요소를 절단



관측 영역에서의 절단

(1) 식이 양수이면 경계면 밖에 위치하는 것이고, 음수이면 내부에 위치. 교차하는 경우는 당연히 0.

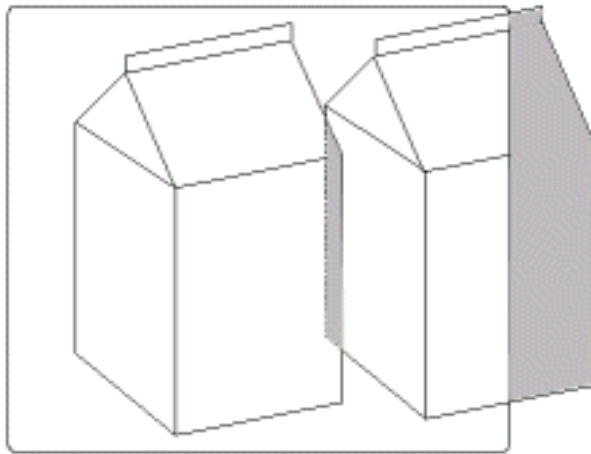


# 은면 및 은선 제거

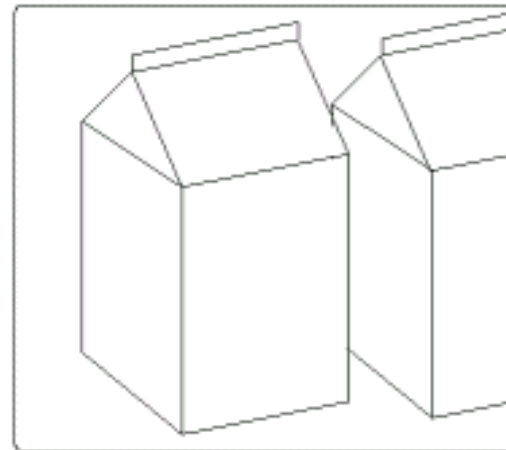


## 은선 제거? 은면 제거?

- 사실적 표현 : 보이지 않는 그림의 부분을 제거
- 은선 제거 : 선으로만 그린 이미지에서 가려진 선을 제거하는 것
- 은면 제거 : 면 자체를 표시한 이미지에서 가려진 면을 제거하는 것



■ 제거할 면



은면을 제거한 결과

# 은면 및 은선 제거



## 알고리즘 분류

- 물체공간법(object space method): 물체를 정의한 단계에서 적용하는 알고리즘
  - > 물체와 물체의 부분들을 서로 비교하여, 어떤 면들과 선들이 보이지 않는가를 결정함
- 이미지 공간법(image-space method): 투영된 이미지에서 은면과 은선을 찾아 처리하는 알고리즘
  - > 투영평면에서 각 픽셀별로 가시성 여부를 정한다.
- 대부분의 은면제거 알고리즘은 **이미지 공간법**을 사용하지만, 어떤 경우에는 물체 공간법이 보다 효과적으로 사용될 수도 있다.
- 알고리즘 배경
  - > 정렬방법 : 관측평면으로부터 각각의 선이나 면 또는 물체와의 거리를 오름차순으로 배열 하여 어느 것이 뒤에 위치하고 있는지를 쉽게 알아보는데 사용된다.

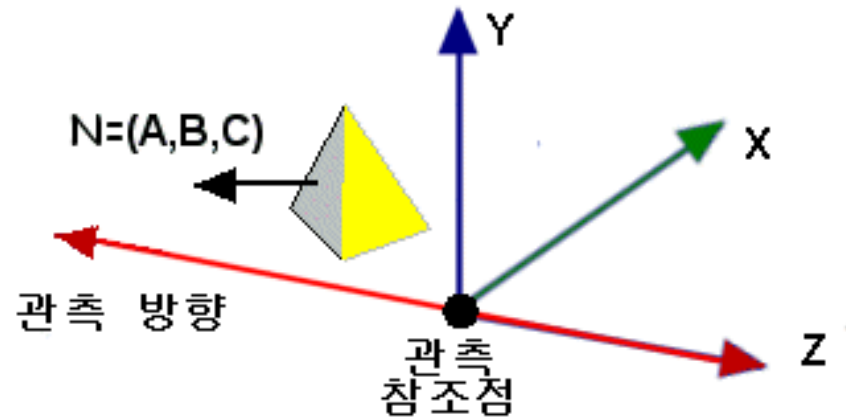
# 은면 및 은선 제거



## 뒷면 제거법(Back-Face Removal)

- 물체 공간법, 평면공식을 사용하여 3D 그래픽 객체의 앞, 뒷면을 구분한다.
- 오른손 좌표체계에서 정의된 임의의 한 점(x, y, z)는 다음의 부등식을 만족시키는 경우에 평면의 안쪽(inside)에 위치한다.
  - > 만약 점 (x', y', z')가 관측점(view point) 또는 관측참조점이라면, 바라보는 평면은 뒷면이다.

$$Ax' + By' + Cz' + D < 0$$

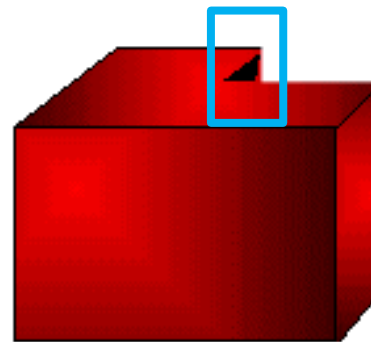


# 은면 및 은선 제거



## 뒷면 제거법(Back-Face Removal)

- 한 물체를 이루는 여러 면들에 대한 평면계수를 조사함으로써, 뒷면여부에 대해서 알아낼 수 있다.
- 피라미드 형태와 같은 볼록한 형태의 물체 : 물체의 은면(hidden surface)을 모두 알아 낼 수 있다. 이것은 각 면이 전체가 보이거나, 또는 전체가 보이지 않거나 하는 경우만 존재하기 때문이다.
- 그 외의 물체 : 한 면이 전체로 보이는지 부분적으로 보이는 지의 여부를 더 조사해야 한다. 또한 그 물체가 다른 물체에 의해서 부분적으로 가려졌는지, 전부 다 가려졌는지의 여부를 판단해야 할 경우도 있다.

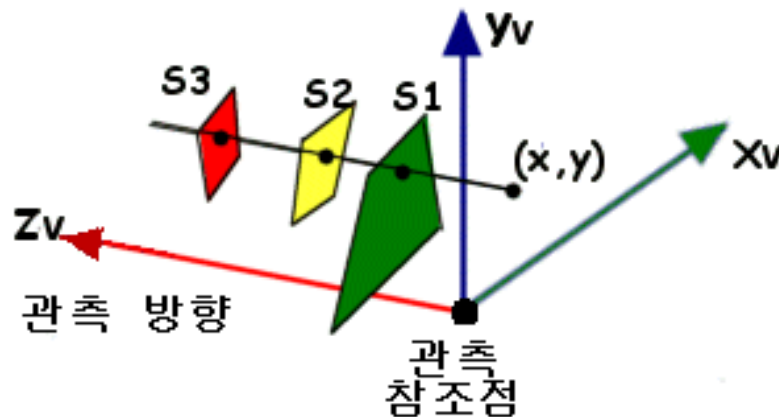


# 은면 및 은선 제거



## Z-버퍼 측정법(Z Buffer, Depth-Buffer)

- 이미지 공간법
- 버퍼심도측정법
  - > 한 표면이 보이는지의 여부를 한번에 한 픽셀씩 조사한다. 관측평면상의 각 픽셀지점( $x, y$ )에 대해서  $z$ 축 값이 가장 작은 평면이(왼손 좌표계에서) 해당 픽셀 지점에서 보이는 평면이 되는 것이다.
- 아래 그림은 왼손 관측좌표계의 한 점( $x, y$ )에 위치하고, 서로 다른 심도( $z$ 값)를 갖는 세 개의 면을 보고 있는 경우이다. 이 점에 표면  $s_1$ 이 가장 작은  $z$ 값을 갖고 있으므로,  $s_1$ 에 관계된 밝기 값이 점( $x, y$ )에 할당된다.



# 은면 및 은선 제거



## Z-버퍼 측정법(Z Buffer, Depth-Buffer)

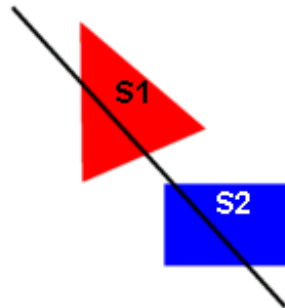
- 2개의 버퍼 필요
  - > z버퍼 : 점(x, y)에서 표면들을 비교하기 위한 z값을 저장하기 위한 버퍼
  - > 재생 버퍼 : 지점에서의 밝기를 저장하기 위한 버퍼
- 알고리즘 개요
  - > 처음에 z-버퍼(심도버퍼)를 모두 1로 초기화
  - > 밝기를 저장하는 버퍼(refresh buffer, frame buffer)를 배경의 색과 동일하게 만든다
  - > 각각의 표면을 처리하되, 한번에 한 주사선씩, 각각의(x, y)점에 대하여 z값을 계산해 나간다.
  - > z-버퍼(심도버퍼)내의 값과 계산된 값을 비교한다.
  - > 계산한 z값이 z-버퍼(심도버퍼)에 저장된 값보다 작으면, 새로운 z값 즉 계산된 z값을 저장하고, 그 지점에 해당하는 프레임(재생) 버퍼에다 그 표면의 밝기 값을 저장한다.

# 은면 및 은선 제거



## Z-버퍼 알고리즘 (1)

1. z(심도) 버퍼와 프레임 버퍼를 초기화 한다. 즉 모든 좌표(x, y)에 대해서  $depth(x, y)=1$ 이고  $refresh(x, y)=background$ 이다.



1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Z- 버퍼

B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B

프레임 버퍼

2. 각 표면상의 점에 대하여 z(심도)버퍼에 저장된 값과 점의 z(심도)값을 비교하여 보이는지의 여부를 결정한다.
  - a. 표면상의 점 (x, y)에서의 z값을 계산한다.
  - b.  $z < depth(x, y)$ 이면,  $depth(x, y)=z$ 로 하고  $refresh(x, y)=i$ 로 놓는다. 여기서 i는 점 (x, y)에서의 해당 표면의 밝기를 나타낸다.

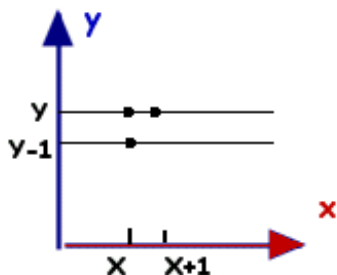
# 은면 및 은선 제거



## Z-버퍼 알고리즘 (3)

각 표면의 평면공식에 의하여 한 점(x, y)에서의 z(심도) 값을 계산해 낼 수가 있다.

$$z = \frac{-Ax - By - D}{C}$$



옆의 그림에서 그려진 주사선을 보면, x좌표는 주사선을 따라서 1씩 바뀌고, y축 값은 주사선이 바뀔 때 마다 1씩 바뀐다.

점(x, y)에서의 심도가 z에 의해 결정 될 때에 다음에 위치한 점(x+1, y)에서의 심도는 다음식에 의해서 결정된다.

$$z' = \frac{-A(x+1) - By - D}{C}$$

$$z' = z - \frac{A}{C}$$

비슷한 방법으로 주사선 사이에서 z(심도)값을 측정할 수 있다. 점(x, y)에서 심도값 z를 갖는다고 하자. 그러면 점(x, y-1)은 바로 다음의 주사선이고, z(심도)값은 평면공식에 의해서 다음과 같이 계산된다.

$$z'' = \frac{-Ax - B(y-1) - D}{C}$$

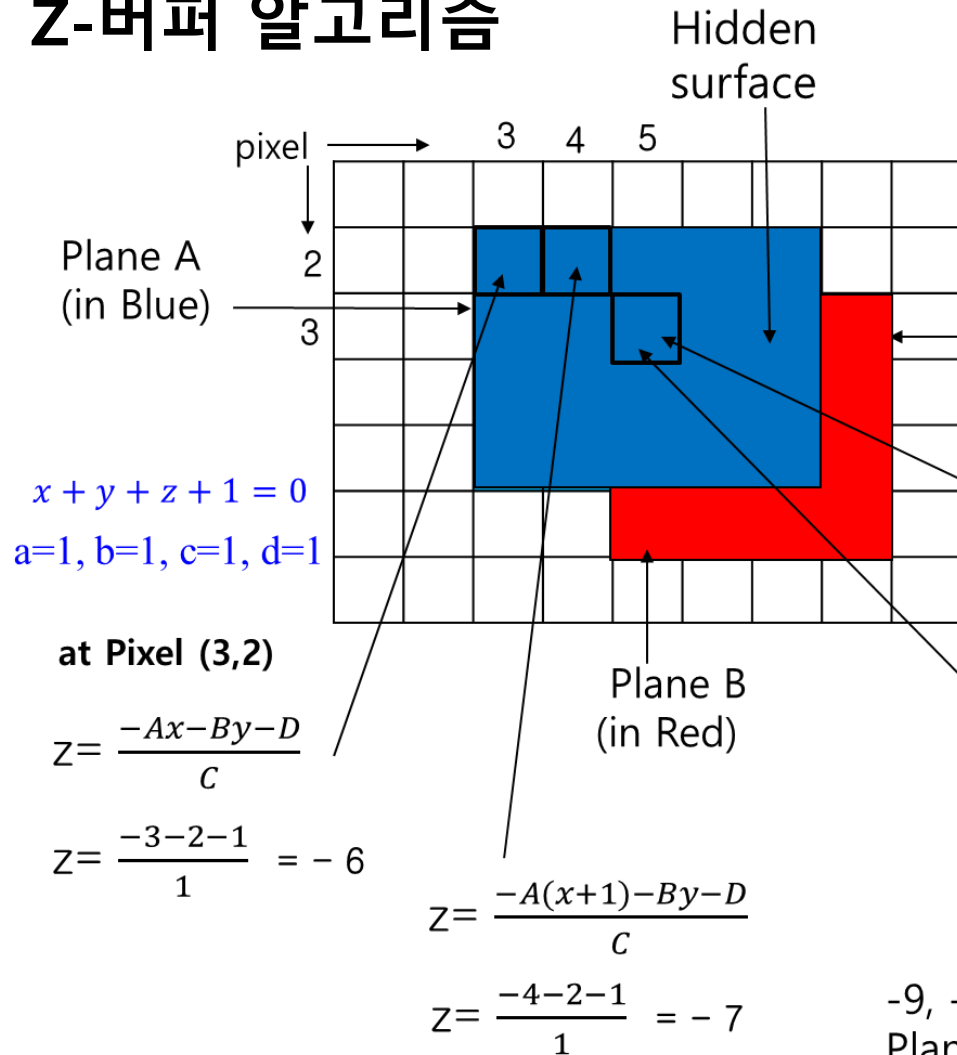
$$z'' = z + \frac{B}{C}$$



# 은면 및 은선 제거



## Z-버퍼 알고리즘



Three depth values

- Background (z=1)
- Blue plane A
- Red plane B

$$x + 3y + z = 0$$

$$a=1, b=3, c=1, d=0$$

$$Z = \frac{-A(x+2) - B(y+1) - D}{C}$$

$$Z = \frac{-5 - 9}{1} = -14$$

$$a=1, b=1, c=1, d=1$$

$$Z = \frac{-A(x+2) - B(y+1) - D}{C}$$

$$Z = \frac{-5 - 3 - 1}{1} = -9$$

-9, -14 reflected value: 9, 14  
Plane A and Plane B -> unit Z  
Background z=1

# 은면 및 은선 제거



## Z-버퍼 알고리즘

- 실제 구현하기가 쉽고, 정렬작업이 필요하지 않다.
- 프레임 버퍼 이외에도 z(심도) 버퍼가 필요.
  - > 해상도  $1024 \times 1024$ 인 출력장치를 위해서는 z(심도)버퍼를 위하여 백만개 이상의 자리가 추가로 필요하고, 각 자리를 z축의 수를 충분히 나타낼 수 있는 정도의 비트를 포함하여야 한다.
  - > 메모리의 크기를 줄이는 방법중의 하나는 작은 z(심도) 버퍼를 사용하여 하나의 그림을 작은 부분으로 나누어 하나씩 처리하는 것이다. 즉 그림의 한 부분이 처리되면 z(심도)버퍼를 그림의 다른 부분에 적용하여 그려 나가는 것이다.

# 은면 및 은선 제거



## 주사선법 (Scan Line Method)

- 이미지 공간법
- 다각형의 내부를 채우는 주사선 알고리즘을 확장
  - > 여러 개의 표면을 동시에 취급
  - > 각각의 주사선을 처리할 때 마다, 주사선과 교차하는 다각형 표면(polygon surface)을 조사하여 어느 것이 보이는지를 알아낸다.
  - > 한 주사선상의 모든 점에 대해서 각 표면에 대한 심도를 계산한 후에 어느 표면이 관측평면에 보다 가까운지를 결정한다.
  - > 보이는 표면이 결정되면, 그 지점에 보이는 표면의 밝기를 해당 프레임 버퍼에 저장하는 것이다.
  - > 하나의 주사선과 만나는 표면을 쉽게 찾기 위해서, 변 테이블에서 현재 액티브 변에 대한 목록을 만든다.
  - > **액티브 목록(active list)** :
    - .. 현재의 주사선과 만나는 변(edge)들만을 포함
    - .. x값에 따라 오름차순으로 정렬되어 있다.
    - .. 주사선상의 점이 표면의 내부에 위치하고 있는지를 on/off로 설정하는 표시자(flag)를 포함한다. 그리고 주사선들은 왼쪽에서 오른쪽으로 순서대로 처리한다. 표면의 맨 왼쪽 경계선에 이르면 표면상태 표시를 on 시키고, 맨 오른쪽 경계선에 이르면 off 시킨다.

# 은면 및 은선 제거

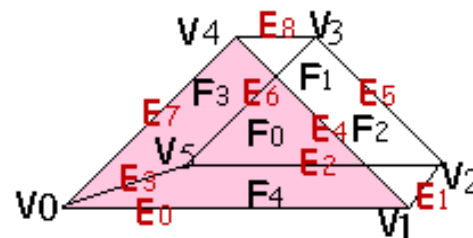
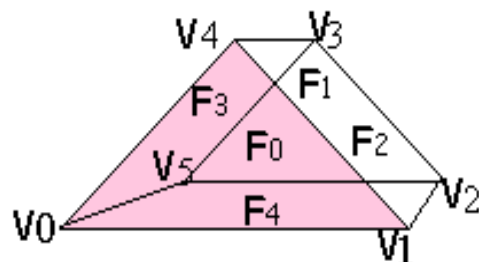


## 다각형 테이블 (Polygon Tables)

- 다각형 테이블
  - > 다각형 표면이 사용자에게 의해서 한번 정의가 되면, 그래픽스 패키지는 이 값을 테이블에 저장
  - > 이 테이블을 이용하여 한 표면이 처리되어, 화면에 보이게 된다.
  - > 대상물체의 기하학적 성질과 속성을 포함하고 있으며, 컴퓨터 내에서의 처리가 용이하게끔 구성됨
  - > 기하학적(또는 도형) 데이터 테이블(geometric data table)은 경계선 좌표와 다각형 표면의 공간에서의 방향을 나타내는 매개변수 값을 포함
  - > 물체의 속성에 관한 정보 : 색깔이나 명암등에 관한 것
- 기하학적 데이터를 구성하는 방법
  - > 꼭지점 테이블(vertex table)
  - > 변 테이블(edge table)
  - > 다각형 테이블(polygon table)

# 은면 및 은선 제거

## 🔍 다각형 테이블 (Polygon Tables)



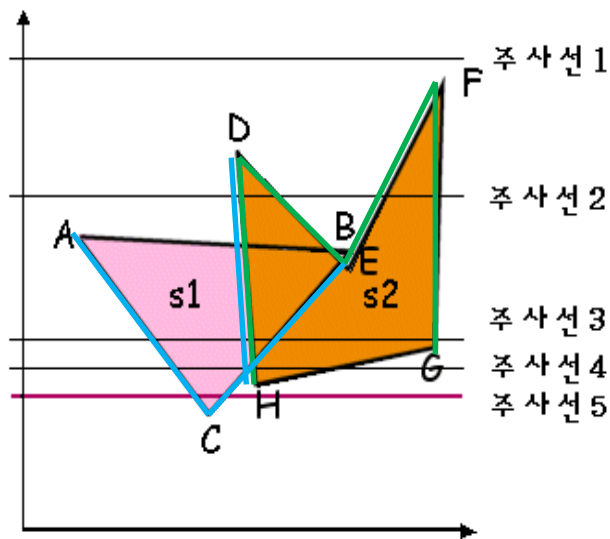
꼭지점 테이블	다각형면 테이블
$V_0 : x_0, y_0, z_0$	$F_0 : V_0, V_1, V_4$
$V_1 : x_1, y_1, z_1$	$F_1 : V_3, V_2, V_5$
$V_2 : x_2, y_2, z_2$	$F_2 : V_1, V_2, V_3, V_4$
$V_3 : x_3, y_3, z_3$	$F_3 : V_0, V_3, V_4, V_5$
$V_4 : x_4, y_4, z_4$	$F_4 : V_0, V_1, V_2, V_5$
$V_5 : x_5, y_5, z_5$	

변 테이블
$E_0 : V_0, V_1, F_0, F_4$
$E_1 : V_1, V_2, F_2, F_4$
$E_2 : V_2, V_3, F_2, F_4$
$E_3 : V_0, V_5, F_3, F_4$
$E_4 : V_1, V_4, F_0, F_2$
$E_5 : V_2, V_3, F_1, F_2$
$E_6 : V_3, V_5, F_1, F_3$
$E_7 : V_0, V_4, F_0, F_3$
$E_8 : V_3, V_4, F_2, F_3$

# 은면 및 은선 제거



## 주사선법(Scan Line Method)



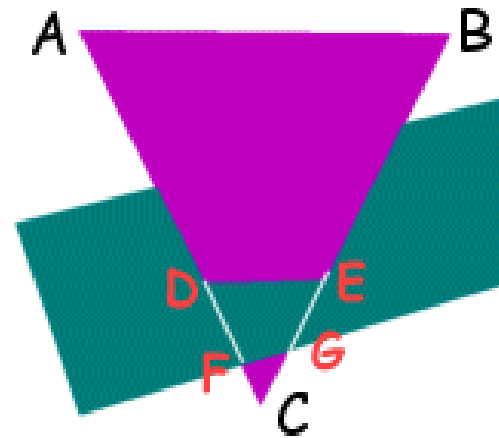
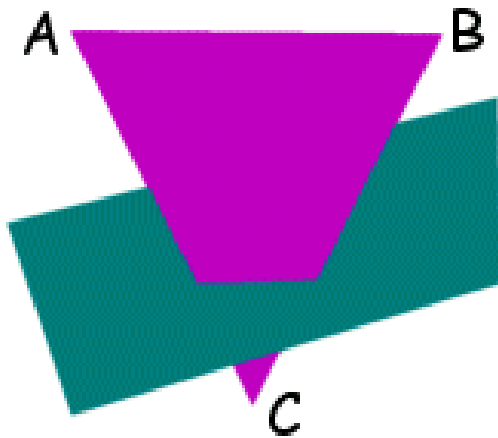
- 주사선 1  
> 어떤 표면도 지나지 않으므로 주사선의 모든 점에서의 밝기 값(색상)은 배경의 밝기(색)가 된다.
- 주사선 2 : DH, DE, EF, FG  
> 변 DH와 변 DE 사이의 주사선상의 점에서는 표면 s2만이 on, 표면 s2에 대한 밝기 값을 다각형 테이블에서 찾아서 프레임 버퍼에 저장  
> 변 EF와 FG에서는 표면 s2만이 on, 주사선 2와 교차하는 다른 표면은 없으므로, 나머지 부분의 점들에 대해서는 배경의 밝기로 된다.
- 주사선 3 : AC, DH, BC, FG  
> AC에서 변 DH까지는 표면 s1만이 on으로 정해지고 변 DH와 BC사이에는 표면 s1과 표면 s2 모두 on으로 된다. 이때에는 두 표면에 대한 평면공식을 이용하여 심도 계산을 하여야 한다. 이 예에서는 **표면 s2의 심도가 표면 s1보다 크다고 가정하면 표면 s1의 밝기 값이** 경계선 BC를 만날때 까지 프레임 버퍼에 저장된다.

# 은면 및 은선 제거



## 주사선법(Scan Line Method)

- 상대방의 표면을 가리는 경우
  - > 표면을 나누어서 생각
  - > 각각을 별개의 평면으로 취급하면 어느 두 평면도 상대방을 보이고 안보이게 하는 경우가 생기지 않는다.



# 은면 및 은선 제거



## 심도정렬법(Depth-Sorting Method)

- 이미지공간법과 물체공간법
  1. 내림차순의 심도값에 의하여 표면을 정렬한다 -> 이미지공간법
  2. 가장 큰 심도값을 갖는 표면을 시작으로, 표면을 순서대로 주사하여 그려나간다 -> 물체공간법
- 화가의 방법 (Painter's Algorithm)
  - > 캔버스에 먼저 배경의 색을 칠한다. => 가장 먼 곳에 위치한 물체를 그림=> 마지막으로 가장 근접한 물체를 좀 더 멀리 떨어진 물체 위에다 그려나감
  - > 관측평면으로 떨어진 거리에 따라서 표면들을 배열시킨다. =>가장 멀리 떨어진 표면의 밝기를 프레임 버퍼에 저장시킨다. => 다음의 표면 밝기를 프레임 버퍼에 저장시켜 나감
  - ..결과적으로 한 표면의 밝기를 바로 전에 처리한 표면의 밝기 위에다 프레임 버퍼 상에서 덧칠하는 효과.

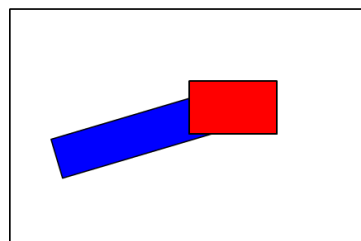
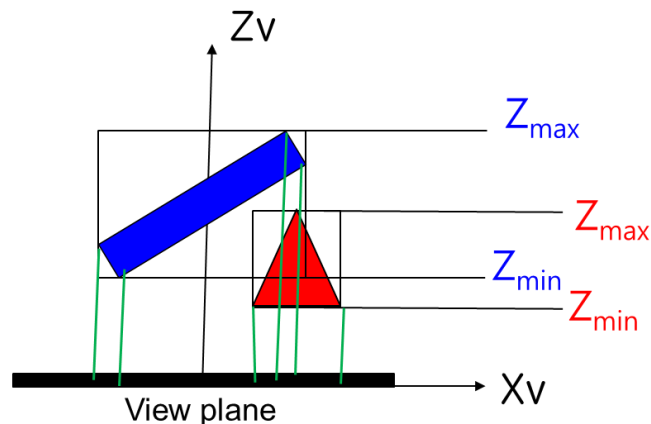
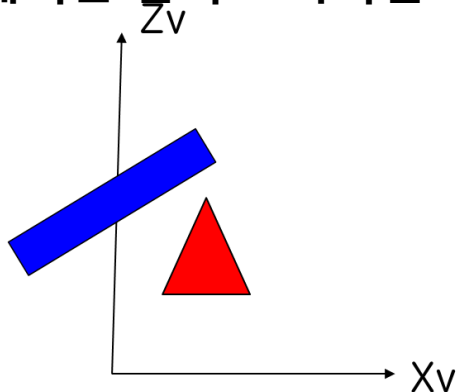
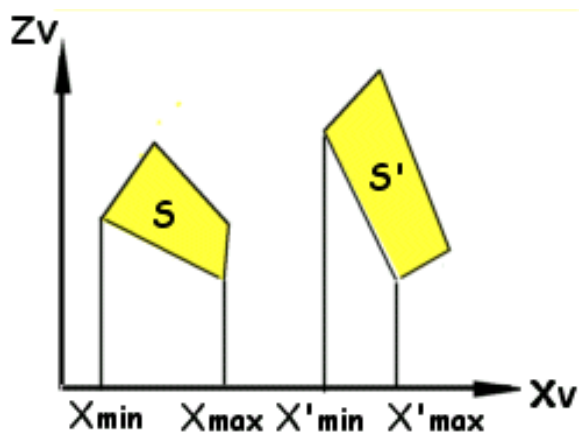


# 은면 및 은선 제거



## 심도정렬법(Depth-Sorting Method)

- $xy$ 평면상에서 생기는 둘레(한계사각형 Minimum Bounding Rectangle: MBR)가 서로 겹치지 않는 경우  
>  $x$ 축 방향에서 겹치는지 여부를 조사하고, 다음에  $y$ 축 방향에서 조사한다.  
이 양방향에서 서로 겹치지 않으면, 두 평면은 서로 상대방을 가리지 않고 있다.
- 다음 그림은  $z$ 방향에서는 겹쳐 보이지만  $x$ 방향에서는 겹쳐지지 않는 두 평면의 예이다.



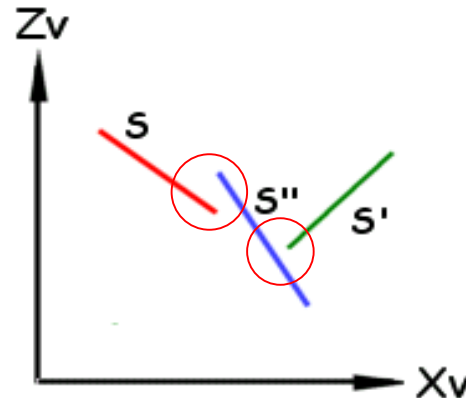
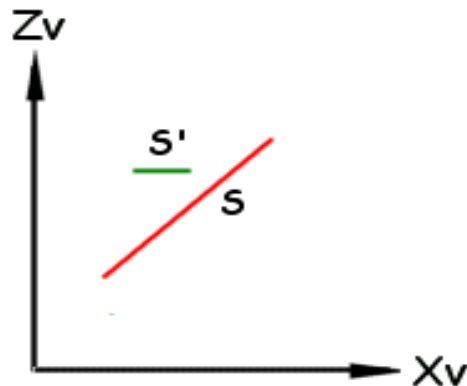
Viewport

# 은면 및 은선 제거



## 심도정렬법(Depth-Sorting Method)

- 하나의 중첩표면  $s'$  에서, 위 4 개의 조사결과가 모두 실패하면, 표면  $s$  와 중첩표면  $s'$  를 정렬된 목록에서 순서를 바꾸어준다.
- 왼쪽그림은 이와 같은 절차에 의해서 순서를 바꾸어 주어야 할 두 표면의 예: 표면  $s$  는  $z$ (심도)는 크지만 표면  $s'$  를 가리고 있다.

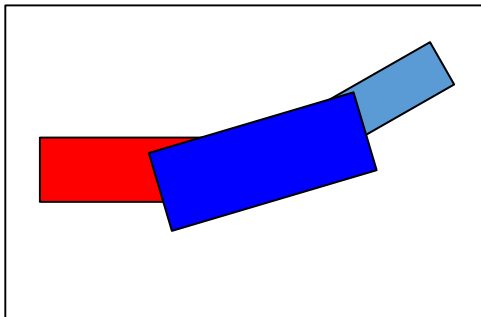
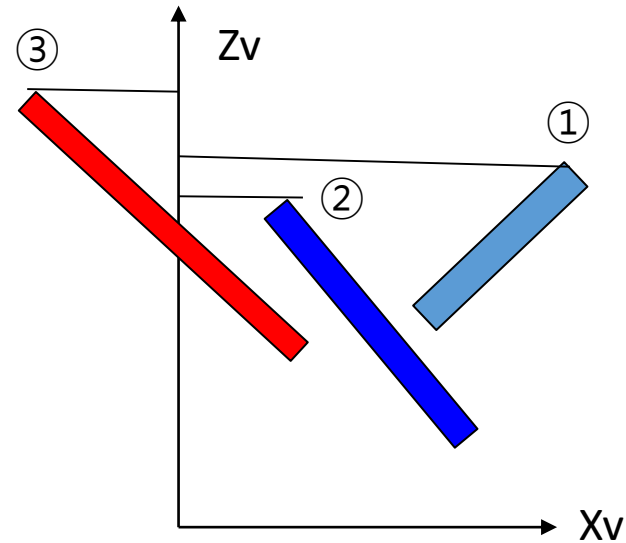
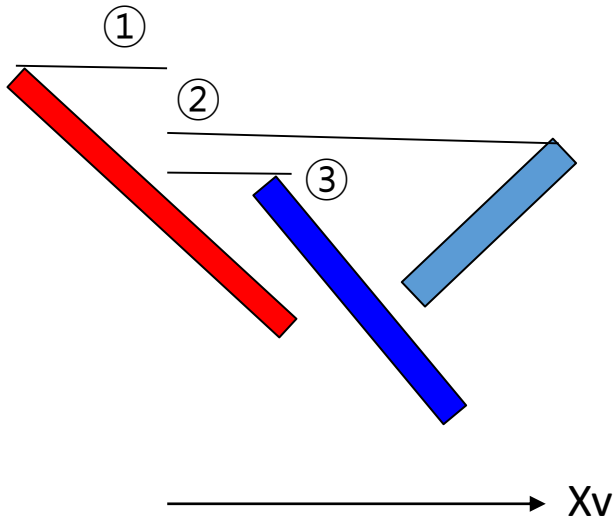


- 오른쪽 그림에서 먼저  $s$  와  $s'''$  을 바꾸어야 한다. 그러나  $s'''$  은  $s'$  의 부분을 가리고 있으므로,  $s'''$  과  $s'$  를 다시 바꾸어야 만, 3개의 표면이 목록에서 제대로의 순서로 정렬이 된다. 따라서 목록에서 자리를 바꾸는 경우에 바꾼 표면에 대해서 다시 위의 조사과정을 재 반복하여야 한다.
- $s, s', s''$  로 정렬되지만 실제로는  $s', s'', s$  로 정렬되어야 한다.

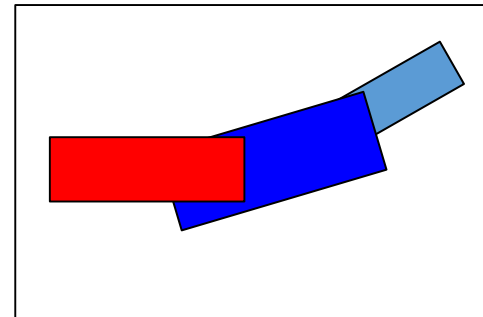
# 은면 및 은선 제거



## 심도정렬법(Depth-Sorting Method)



Viewport



Viewport

# 은면 및 은선 제거



## 은면 제거법 비교

- 은면 제거법의 효율성은 응용분야의 특색에 좌우된다.
  - > 한 장면에 있는 표면들이 z축 방향에서 퍼져있고, 서로 중첩되는 경우가 거의 없는 경우 => 심도 정렬법
  - > 표면들이 수평적으로 분산되어 있는 경우 => 주사선 법
  - > 많지 않은 표면들이 하나의 장면을 구성할 때, 서로 겹치는 표면이 거의 없으므로 => 심도 정렬법
  - > 하나의 장면이 적은 수의 표면으로 구성되어 있는 경우 => 주사선 법
  - > 수 천개의 표면에 의하여 하나의 장면이 구성되거나, 복잡한 그림을 표현하고자 할 때 => Z-버퍼 법

