Computer Graphics

유주한

컴퓨터AI학부
동아대학교
2025년 01학기

컴퓨터 그래픽의 간략한 역사

- CRT(음극선 튜브, Cathode Ray Tube)

단순 Text ➔ 점, 선 ➔ 이미지

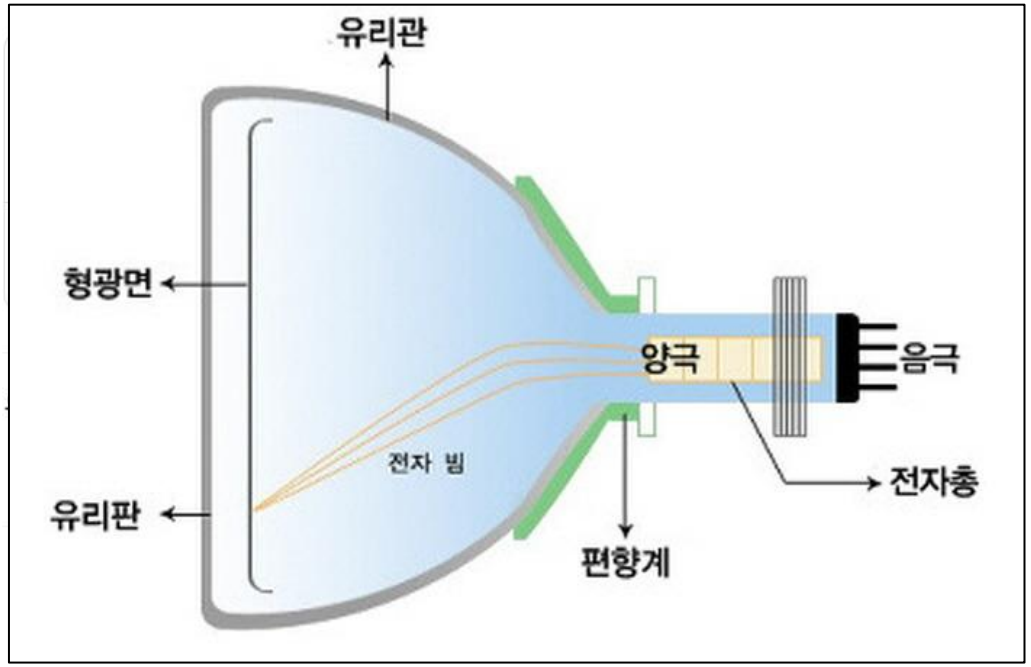
CRT 모니터 평면 게이밍 15인치 컴팩 신품 스크린 CRT 모니터

★ 리뷰 0개

10% ~~64,700원~~

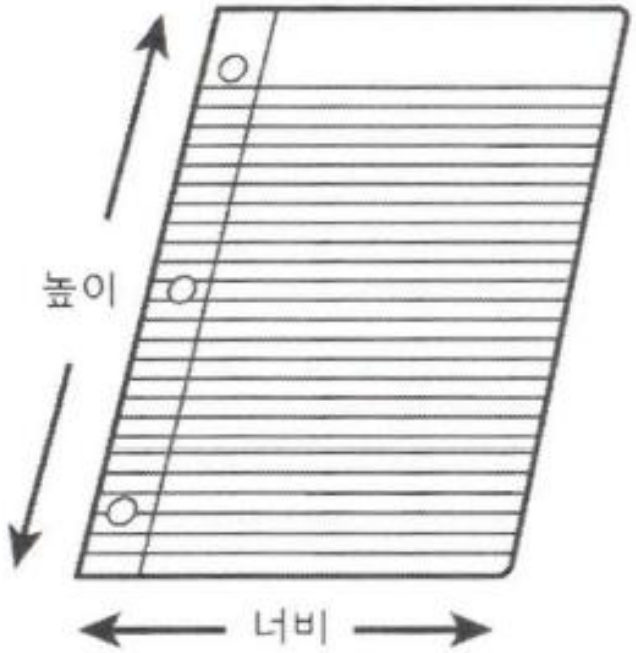
64,700원

58,230원 쿠폰적용가



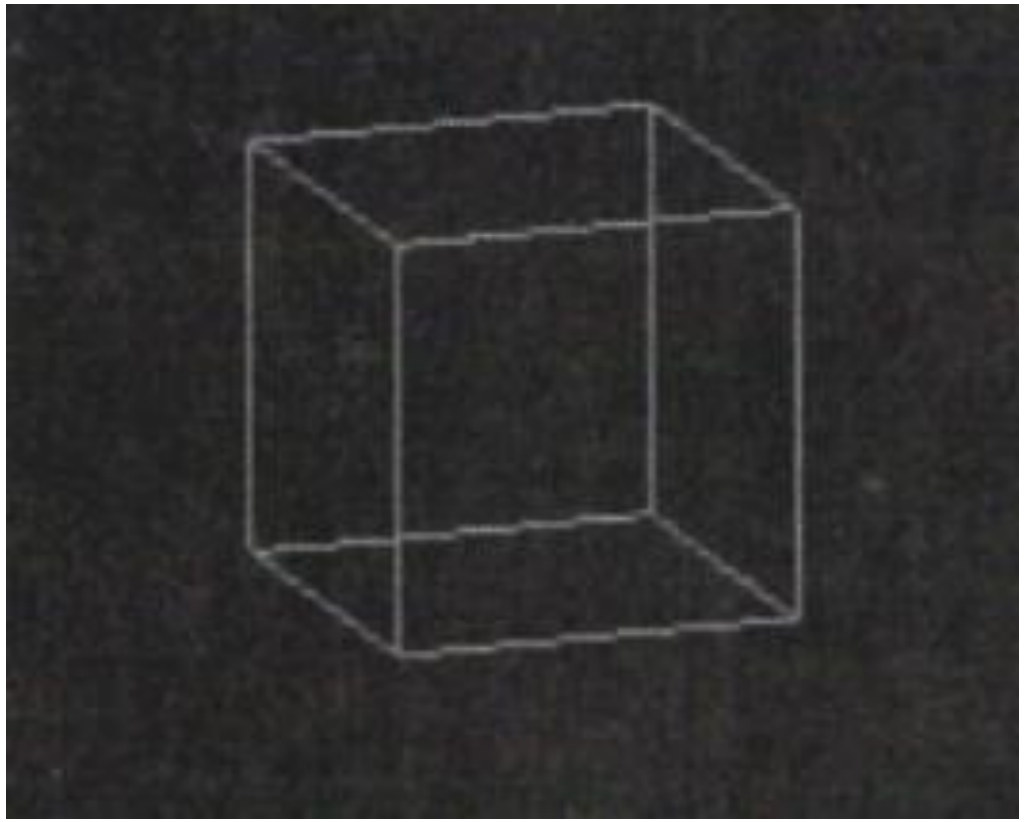
컴퓨터 그래픽의 간략한 역사

- 3D로 진화
  - 깊이 개념 추가 ➔ 물체의 볼륨 형성
- 시선의 방향에 따라
  - 어떤 길이가 물체의 너비 혹은 높이인지 달라짐



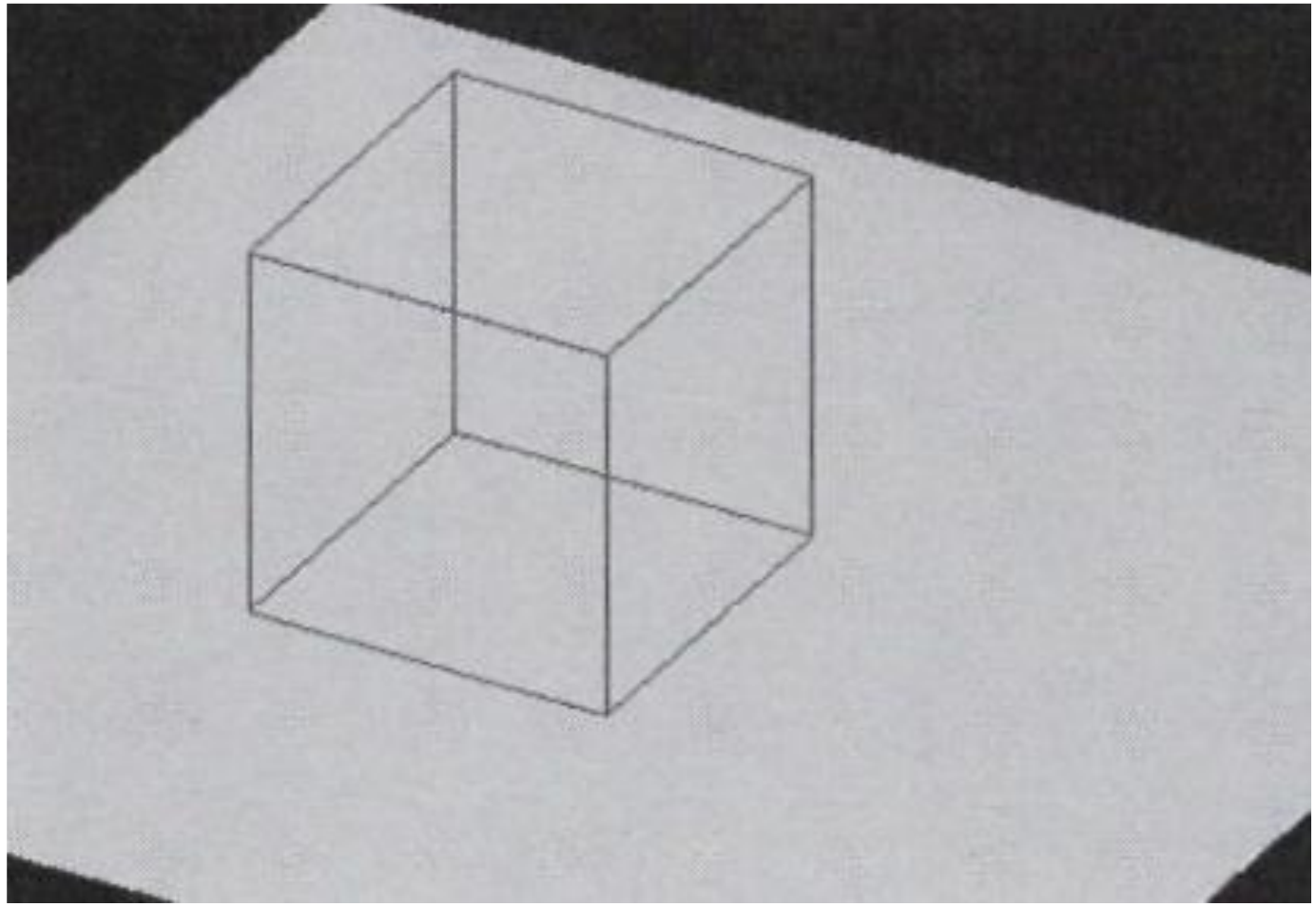
컴퓨터 그래픽의 간략한 역사

- 2D + 원근법 = 3D
- 12 개의 선으로 3차원 상자를 그린 것
  - 원근법을 사용하여 물체를 3차원으로 보이게 만든 예
  - 물체의 깊이 조절
    - ✓선 사이의 각도가 중요한 역할
- 원근법만으로 3D 물체를 표현 가능



컴퓨터 그래픽의 간략한 역사

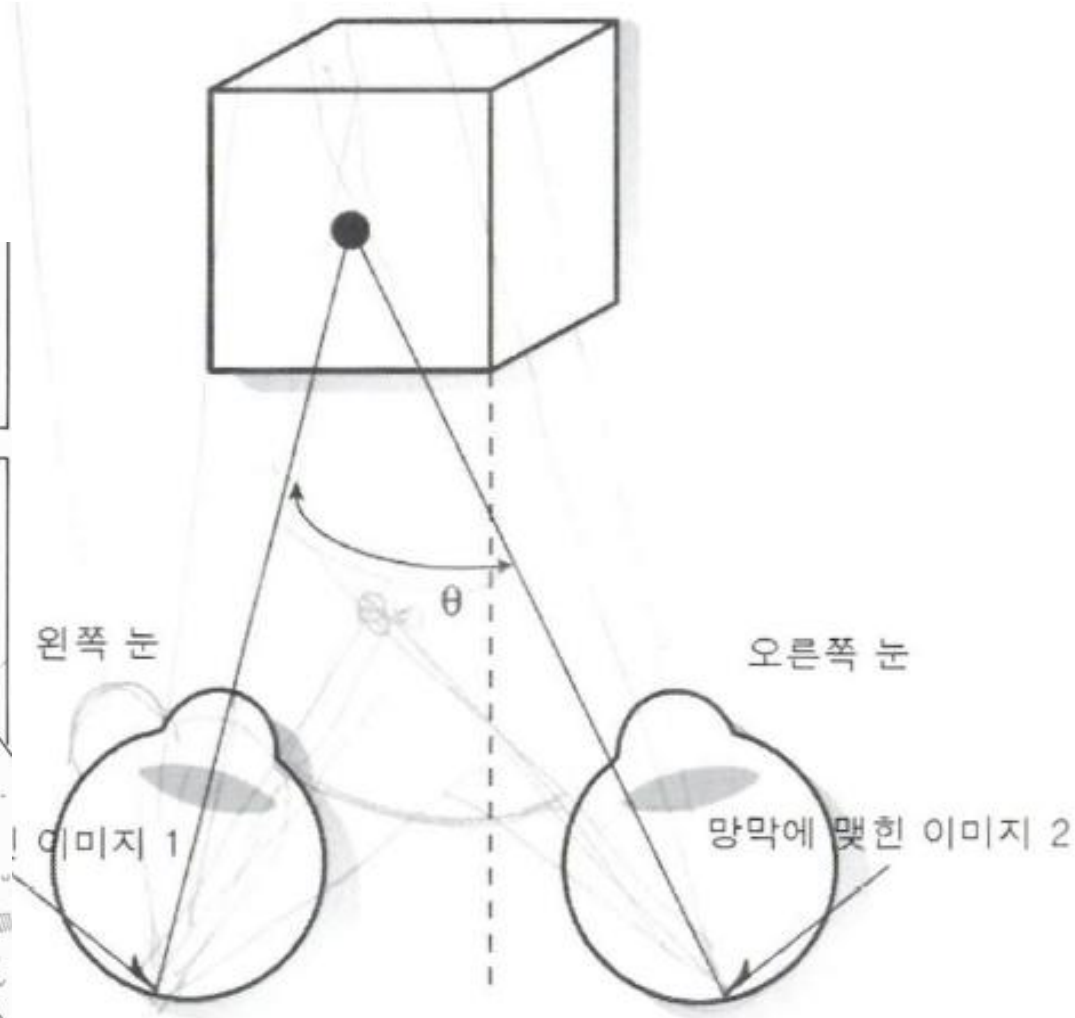
- 원근법만으로 3D 물체를 표현 가능한가?
  - 아래 그림을 오랫동안 보고 있으면 상자의 앞-뒤가 바뀌어 보이는 혼란 발생
  - 이런 현상을 '**팝핑(popping)**' 이라고 함



컴퓨터 그래픽의 간략한 역사

- 실제로 3D 이미지를 보기 위해서는 양쪽 눈에서 보여지는 내용이 결합되는 과정 필요
  - 실제로 3D 이미지를 보기 위해서는 양쪽 눈에서 보여지는 내용이 결합되는 과정이 필요

풍전고교 남훈

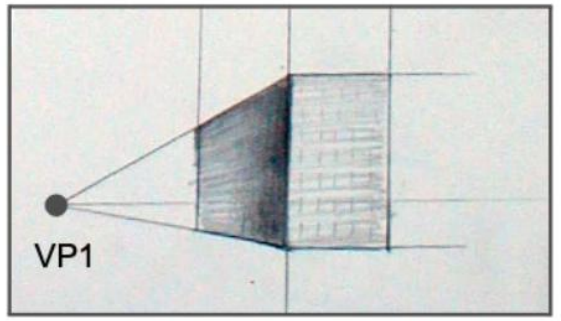


컴퓨터 그래픽의 간략한 역사

- 원근법(Perspective)
  - 한쪽 눈을 가리더라도 모든 3D 공간이 2D가 되지 않음
    - ✓가까이 있는 물체가 멀리 있는 물체보다 크다
      - ➔ 뇌는 눈에 보여지는 2D 효과를 3D로 인식
  - 여기에 색상의 변화, 텍스처, 조명, 쉐이딩 등 다양한 효과를 추가하여 보다 사실적인 3D 물체를 표현

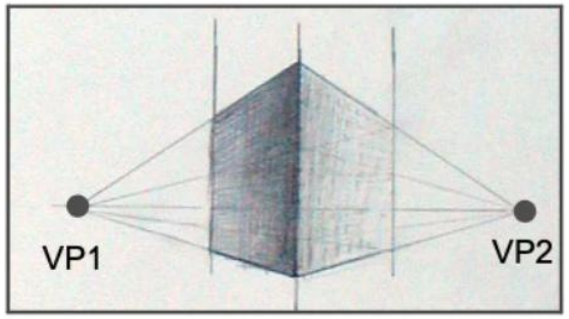
One point perspective

For one point perspective, make VP2 and VP3 a very long distance away (so the lines never meet).

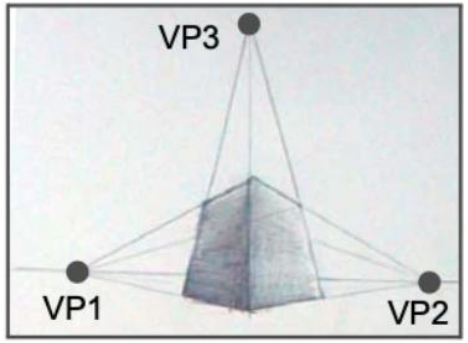


Two point perspective

For two point perspective, make VP3 a very long distance away (so the lines never meet).

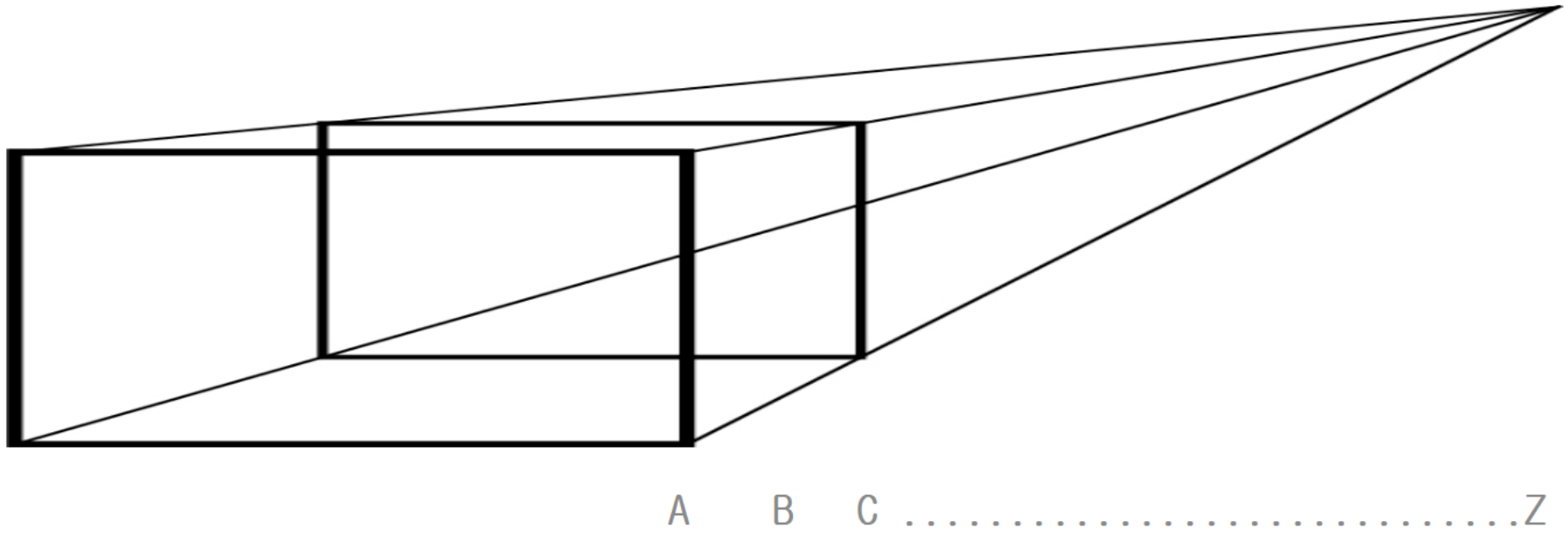


Three point perspective



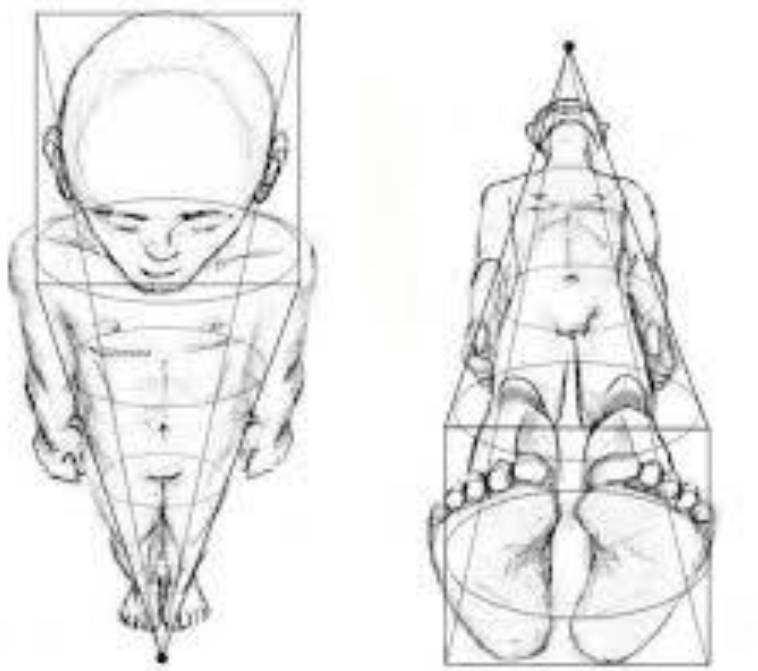
컴퓨터 그래픽의 간략한 역사

- 간단한 원근법
  - 선원근법
    - ✓대상물의 직선 1쌍 혹은 그 이상이 시점으로부터 멀어질수록 서로 가까워 짐 ➔ 소실점
    - ✓멀어질수록 작아 짐



컴퓨터 그래픽의 간략한 역사

- 원근법을 꼭 지켜야 하나?
  - 미술에서는 **강조**의 기법 중 하나로 **축소법(Foreshortening)**을 이용
  - 선원근법 원리를 잘~ 활용? (역으로?)
  - 앞(fore) 쪽으로 짧아(shorten) 보이게 만들다
  - 사물이 기울어졌을 때 원래 길이보다 "짧게" 보이는 것을 강조



컴퓨터 그래픽의 간략한 역사

- ‘에마우스에서의 저녁 식사’ 작품
  - 카라바조
  - 중앙에 있는 인물이 재림한 예수임을 알고 놀라는 제자들을 표현
  - 오른쪽 인물의 양 손의 원근법을 무시 ➔ 강조
    - ✓오른손과 왼손의 크기가 같음

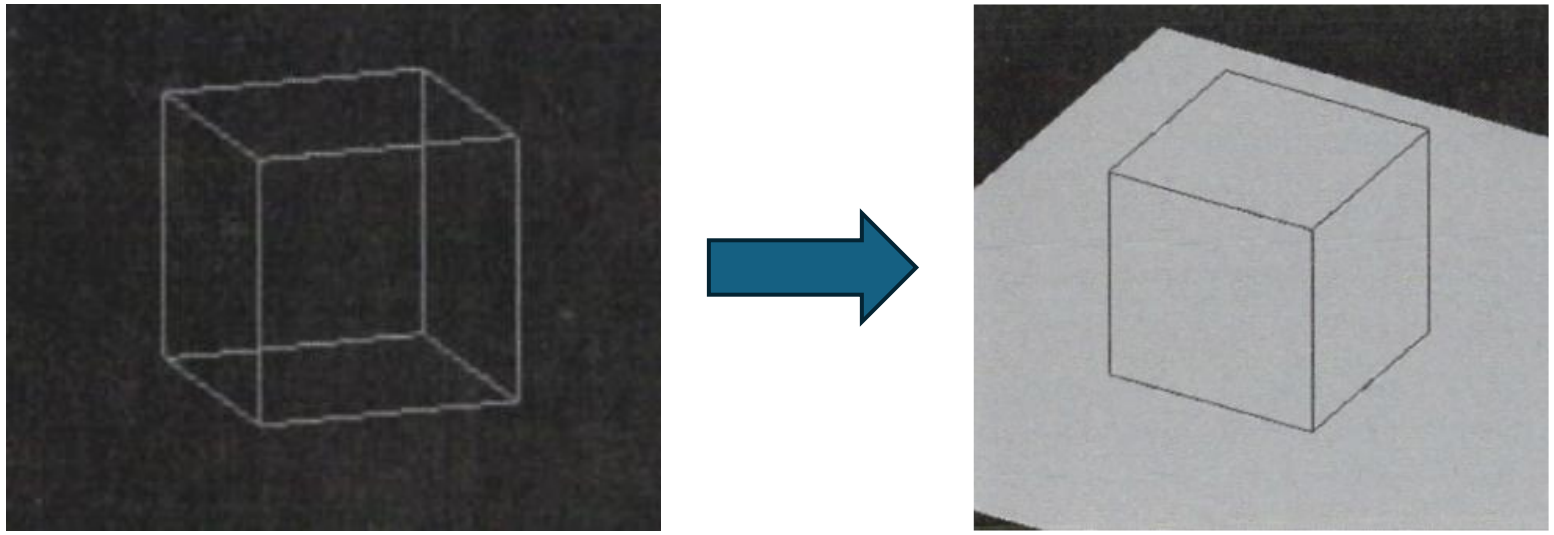
컴퓨터 그래픽의 간략한 역사

- ‘죽은 그리스도에 대한 애도’ 작품
  - 안드레아 만테냐
  - 시점에서 본 예수의 상체보다 하체가 더 작음
    ➔ 원근법 무시 ➔ 역행적 축소법

다양한 3D 효과들

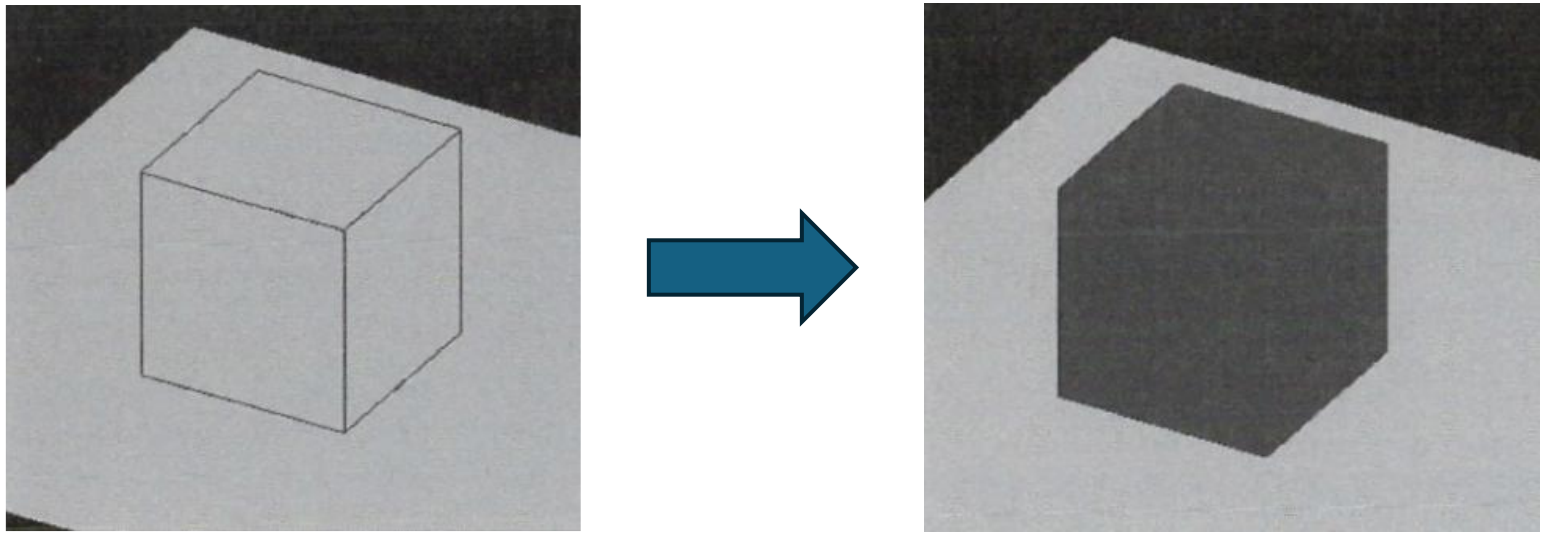
- 렌더링(rendering)
  - 3D 물체에 대한 도형적 정보를 2D 화면에 표시하는 과정

- 원근법(Perspective)
  - 선 사이의 각도를 조절하여 3차원 효과를 내는 기법
  - **팝핑 현상** 제거를 위해 보이지 않는 선들을 제거 ➔ '은면 제거'



다양한 3D 효과들

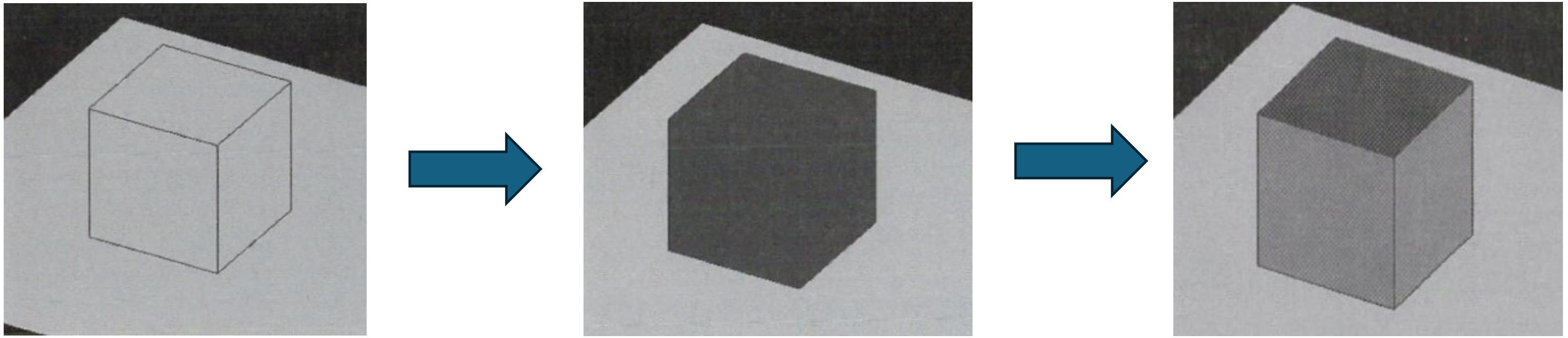
- 색상(Color)
  - 오랫동안 주시 하다보면 볼록 튀어나온 상자가 아니라 움푹 파인 형태로 보일 수 있음
  - 완벽한 3D 효과를 위해서는 각 면에 색상 추가 필요



  - But, 모든 면에 같은 색상을 칠하면 오히려 3D 효과가 사라짐

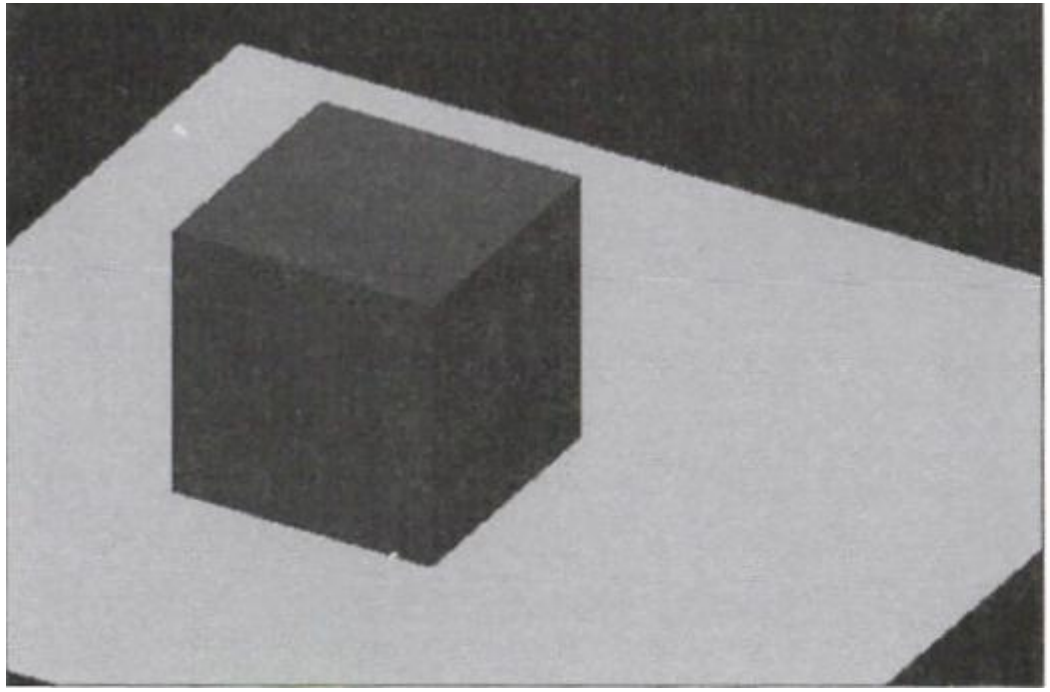
다양한 3D 효과들

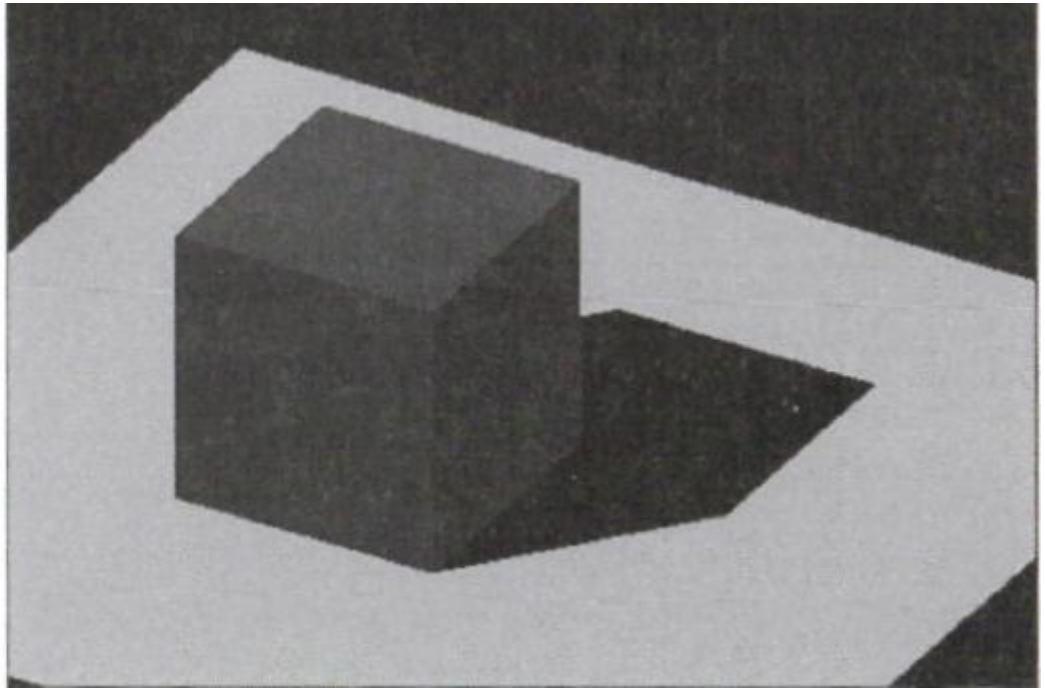
- 색상(Color)
  - But, 모든 면에 같은 색상을 칠하면 오히려 3D 효과가 사라짐
    ➔ 각 면에 서로 다른 색상을 적용(solid object)



다양한 3D 효과들

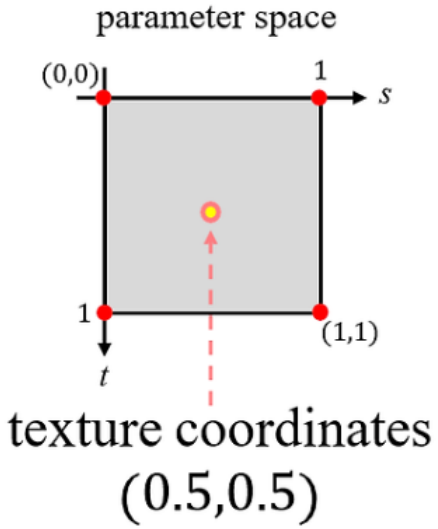
- 조명(Light)과 그림자(Shadow)
  - 보다 사실적인 3D효과
    - ✓조명의 효과를 적용하여 각 면의 색상 진하기를 조절
    - ✓조명의 위치와 물체의 위치 및 각도를 고려하여 적절한 그림자 삽입

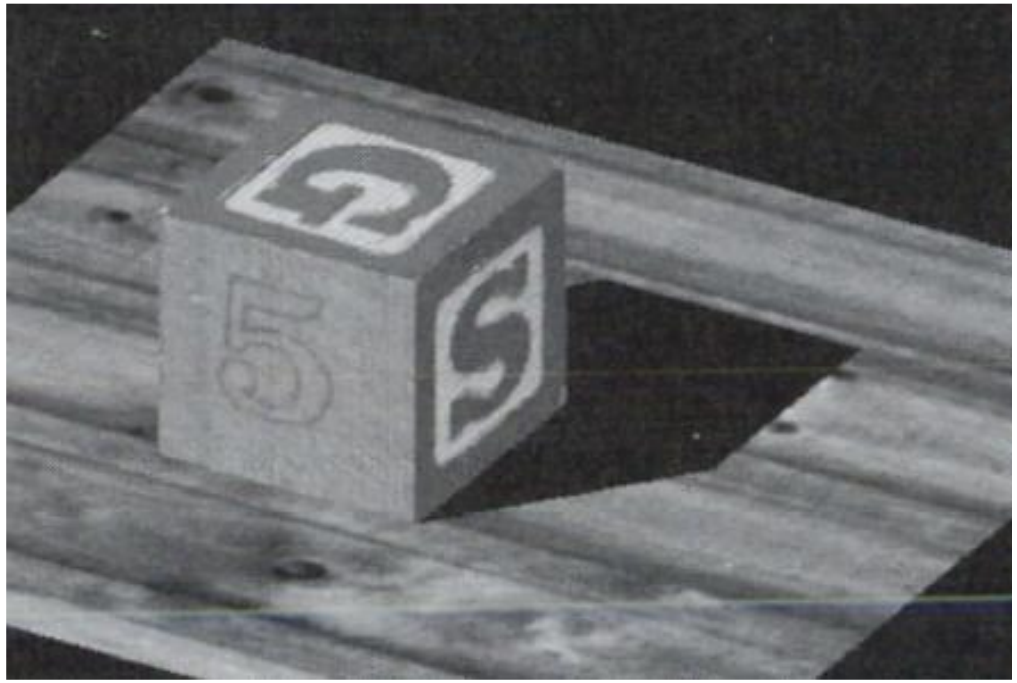




다양한 3D 효과들

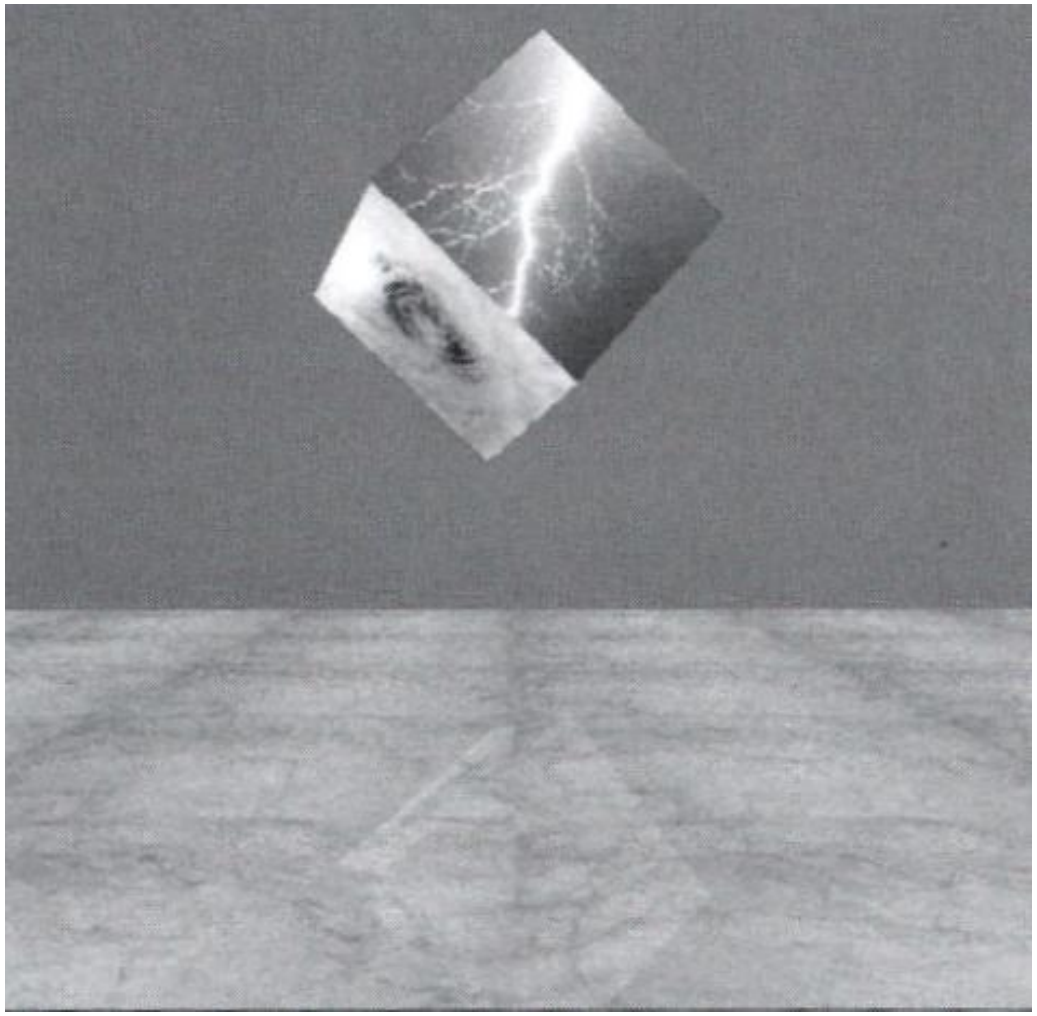
- 텍스쳐(Texture)
  - 3D 그래픽에서 객체의 표면에 적용하는 2D 이미지(또는 데이터)로, 물체의 외형을 더욱 현실감 있게 표현하는 데 사용되는 그래픽 요소
    - ✓나무 결, 천 의 재질, 벽 들 등과 같은 다양한 이미지를 3D 물체 의 각 면에 적절히 입혀주면 3D 효과를 극대화할 수 있음 ➔ 이 과정을 **Texture Mapping**이라고 함
    - ✓텍스쳐의 최소 단위 = 텍셀(texel)
      - pixel 좌표 = x,y
      - texel 좌표 = u,v (0~1)





다양한 3D 효과들

- 블렌딩(Blending) 및 투명(Transparency)
  - 두 이미지가 겹쳐는 효과
  - 물체의 반사 표현
    - ✓유리잔 뒤에 배경이 비치는 등의 효과



다양한 3D 효과들

- 쉐이딩(Shading)
  - **광원, 카메라, 물체의 재질** 등을 고려하여 ➔ **물체의 색상, 그림자, 반사, 블렌딩, 투명도** 등의 효과를 조정하여 보다 현실적인 그래픽을 표현하는 과정
- 쉐이더(Shader)
  - GPU에서 실행되는 작은 프로그램. Shading을 수행
- Vertex
  - 그래픽을 구성하는 가상의 공간에서 존재하는 데이터이며, 최종적으로 픽셀(Pixel)로 변환되어 화면에 출력
  - 2D, 3D 등의 형태 가능
- Pixel
  - 최종적으로 명시적인 화면 공간(디스플레이)에서 나타나는 색상 및 좌표 정보
  - 2D 형태, 3D 형태(Voxel) - 개념적

다양한 3D 효과들

▪ 2D Vertex와 2D Pixel의 일반적인 정의

기준	2D 버텍스(Vertex)	2D 픽셀(Pixel)
공간 유형	암묵적인 공간 (상상의 공간, 가상의 좌표계)	명시적인 공간 (실제 화면, 디스플레이 좌표계)
좌표계	모델 좌표(Model Space), 월드 좌표(World Space), 뷰 좌표(View Space)	스크린 좌표(Screen Space, 실제 픽셀 위치)
출력 여부	직접 화면에 보이지 않음 (도형을 구성하는 데이터)	최종적으로 화면에 표시되는 색상 단위
정보의 종류	위치 (X, Y), 색상(Color), UV 텍스처 좌표, 법선 (Normal) 등	위치 (X, Y), 색상(Color), 밝기(Brightness), 투명도(Alpha)

- **Vertices** exist in **a conceptual coordinate space** and must be transformed into screen space before rasterization occurs
- **Pixels** represent the **final rendered output on a display device**, whereas vertices define geometric primitives before rasterization

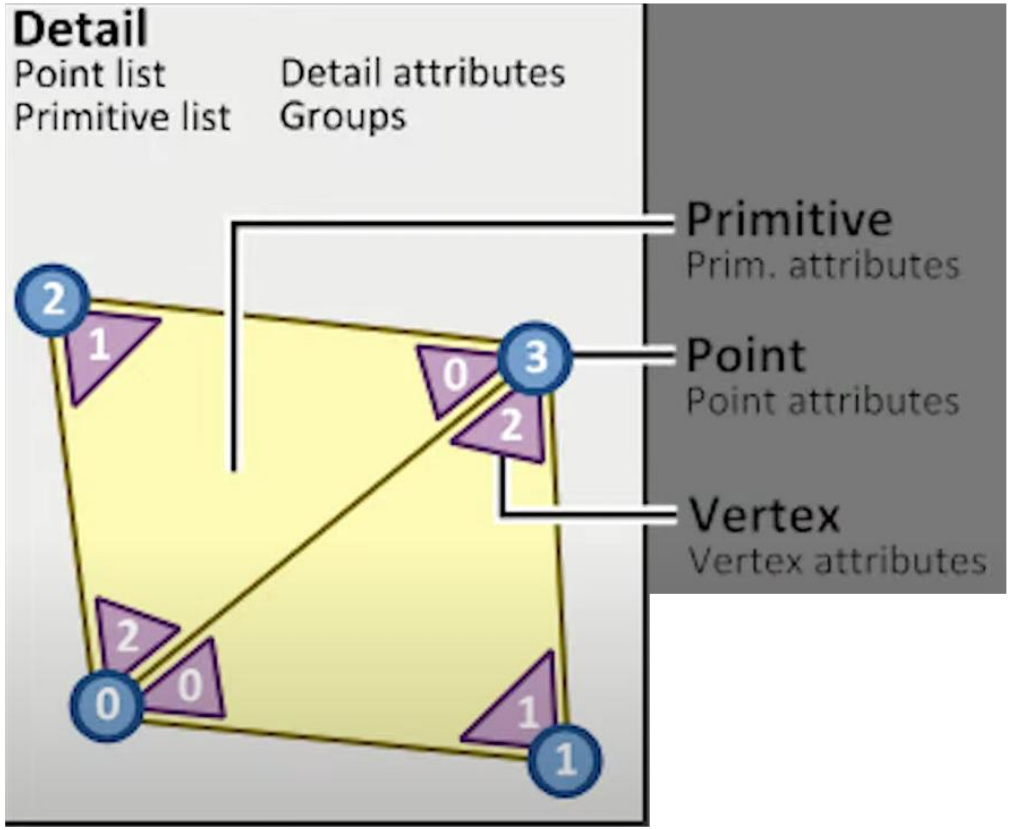
다양한 3D 효과들

- Vertex? Point? (3D 공간에서의 예)
  - Point : "3D 공간에서 한 점 (1,2,3)"아무런 그래픽 속성이 없으며, 단순한 좌표값
  - Vertex : " 3D 모델의 삼각형을 구성하는 꼭짓점 (1,2,3), 색상 (255, 0, 0), 노멀 (0,1,0), 텍스처 좌표 (0.5, 0.5) " 단순한 점이 아니라 그래픽적으로 의미 있는 정점
  - 3D 모델링에서의 예

    ✓어떤 속성을 가지고 있는냐?

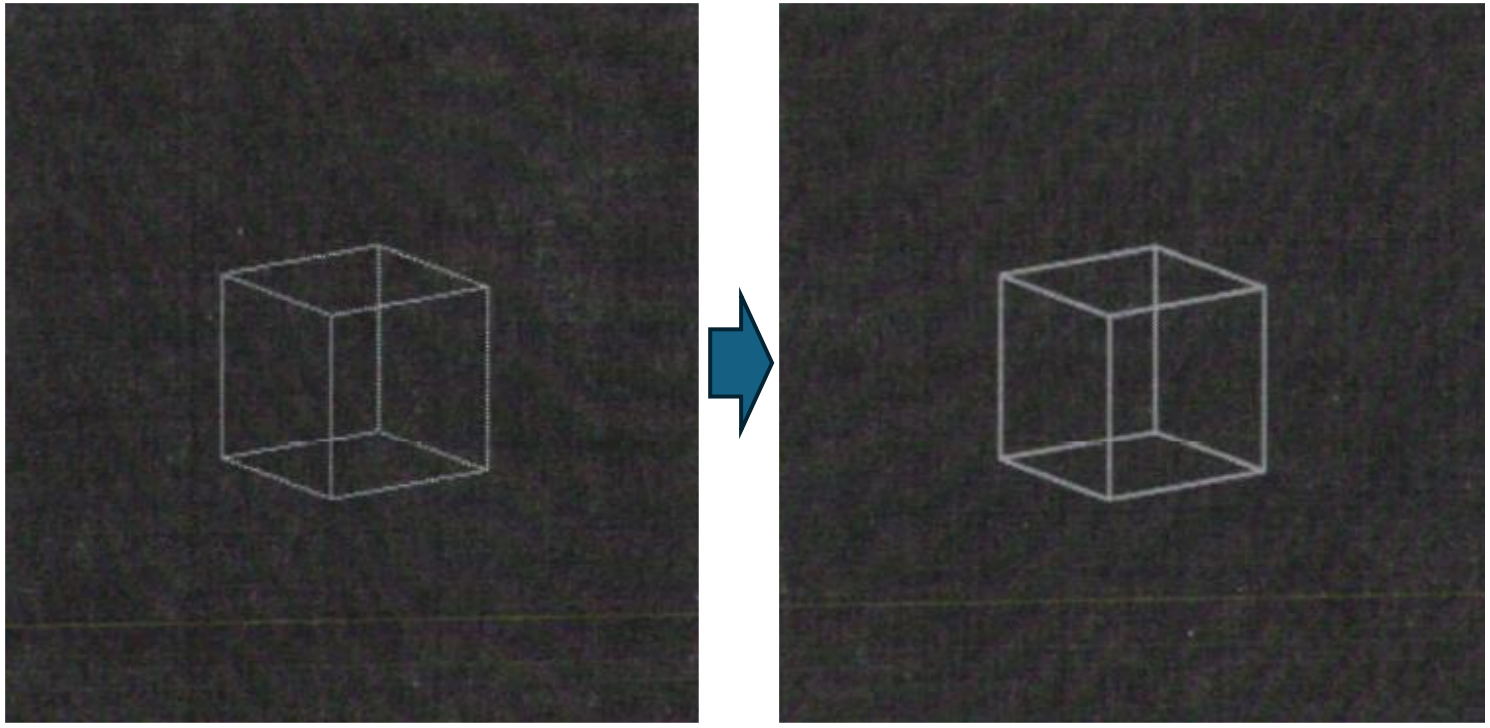
    - 왼쪽 삼각형에 속하나?
    - 오른쪽 삼각형에 속하나?

    ➔ 같은 Point라도 Vertex 정보는 다를 수 있음!



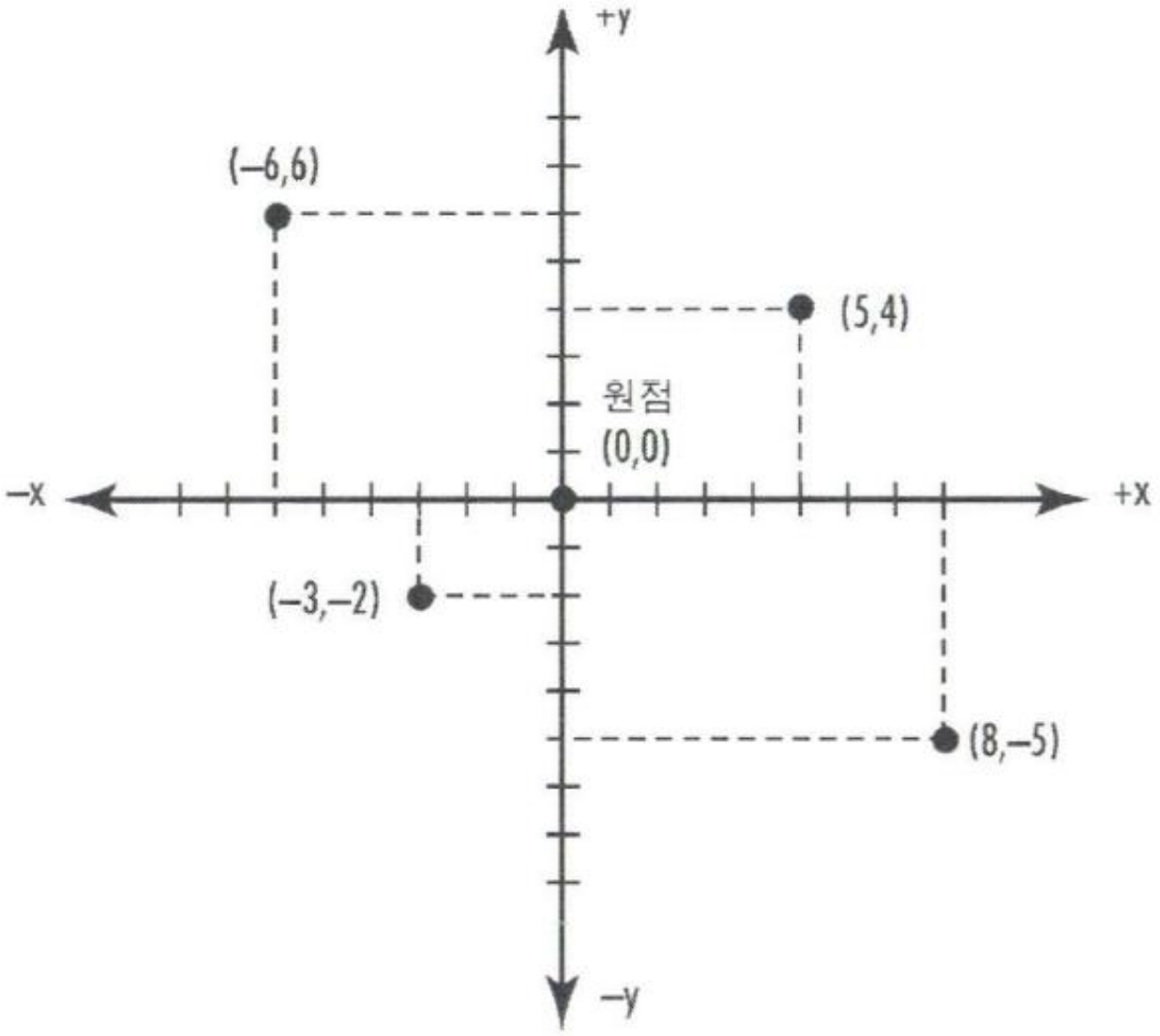
다양한 3D 효과들

- 알리아싱(Aliasing)
  - 픽셀은 정사각형이므로, 곡선이나 대각선이 "계단처럼 보이는(Aliasing)" 현상이 발생
  - 해상도가 낮을수록 이 계단 현상이 더욱 두드러짐
  - 텍스트, 폴리곤, 라인 렌더링, 텍스처 매핑 등에서 Anti-Aliasing 필요
- 안티 알리아싱(Anti-Aliasing)
  - Aliasing을 줄여서
    부드러운 이미지를 만드는 기법



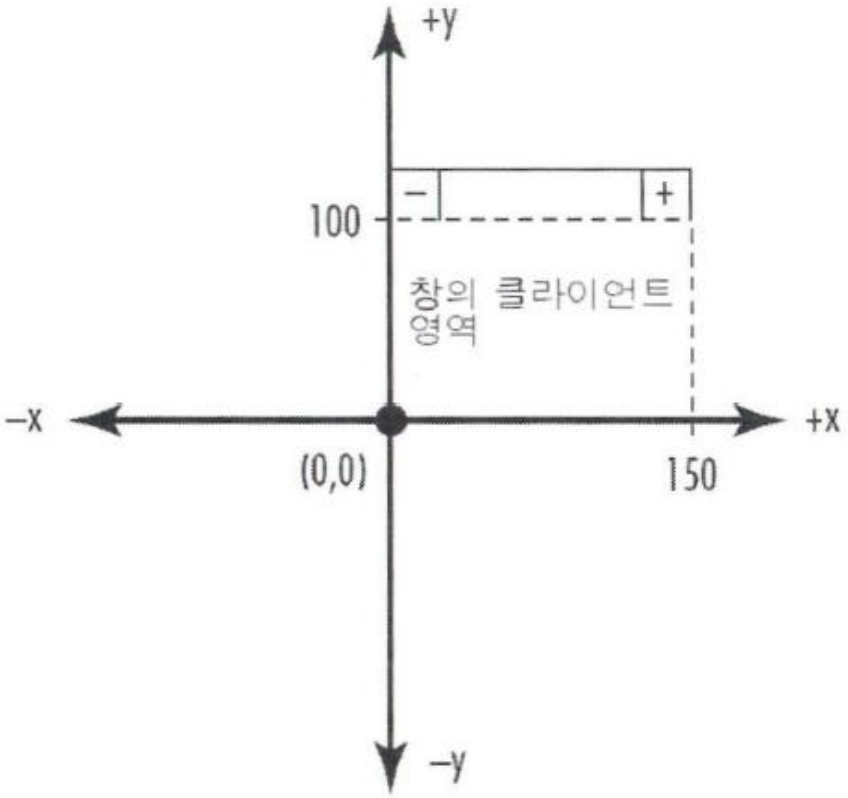
3D 프로그래밍의 기본 지식

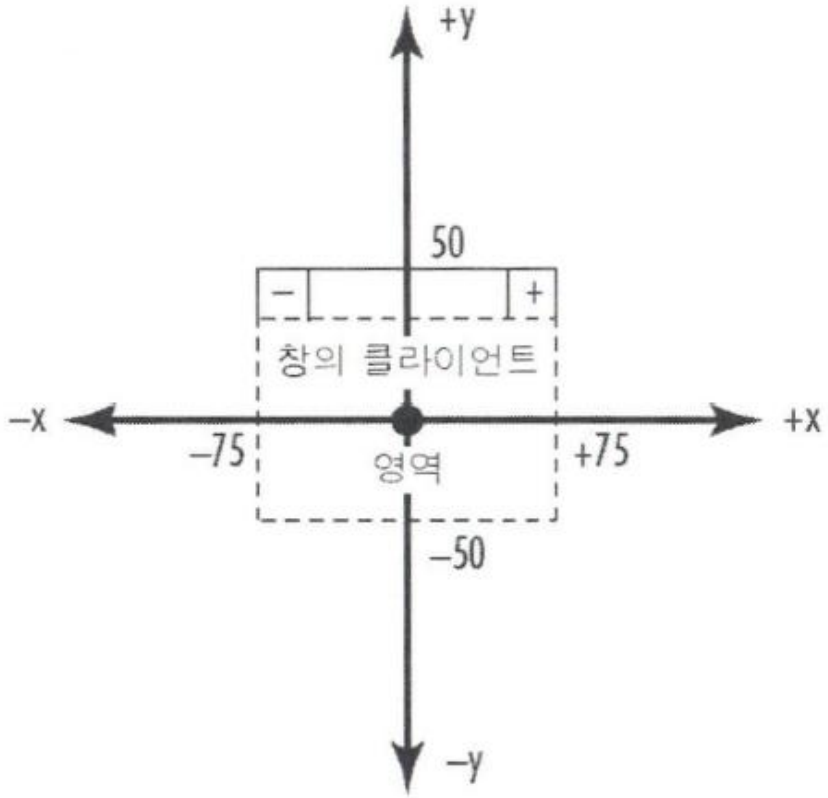
- 2D 직교 좌표계



3D 프로그래밍의 기본 지식

- 좌표의 클리핑(Clipping)
  - 직교 좌표계의 커다란 영역 중 일부를 잘라(clipping) PC의 화면, 즉 창(window)에 할당하는 것
  - 일반적으로 2개의 clipping영역이 이용



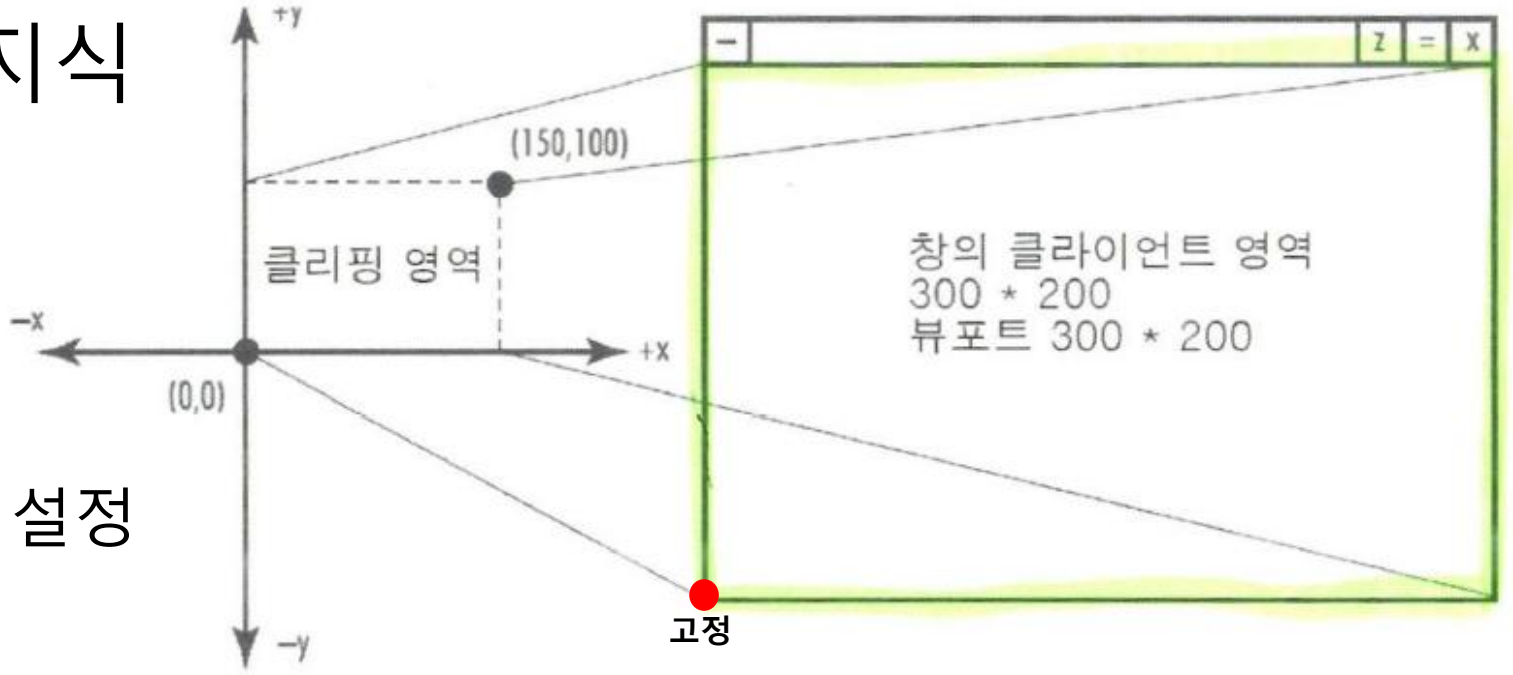


3D 프로그래밍의 기본 지식

- 뷰포트(Viewport)
  - 일반적으로 **클리핑 영역**과 **window의 픽셀 수**가 같지 않음
  - 논리적 직교 좌표를 물리적인 화면 픽셀 좌표로 전환하는 과정이 필요
    ➔ 뷰포트(viewport) 이용
  - window의 클라이언트 영역 내에 clipping 영역을 그리는 데 사용되는 영역
  - clipping 영역을 window에 적용시킴

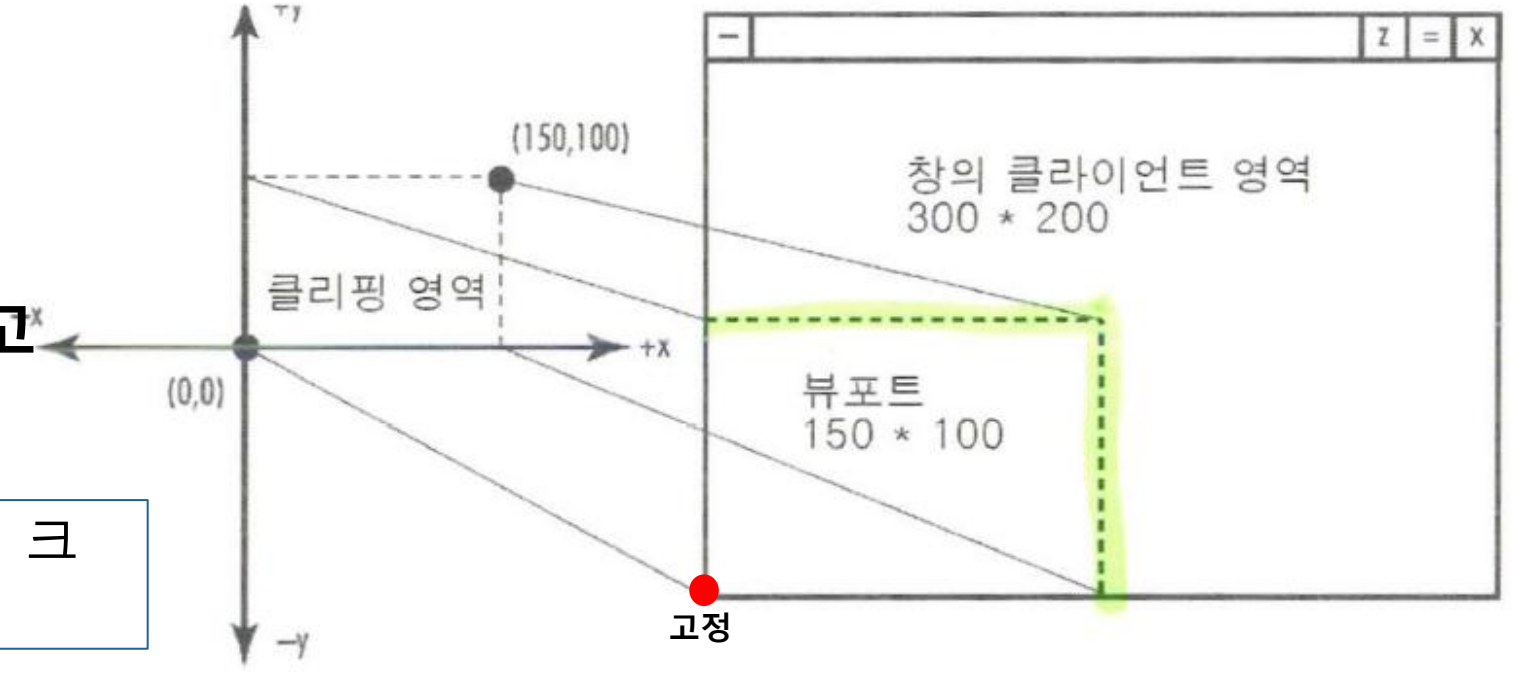
3D 프로그래밍의 기본 지식

- 뷰포트(Viewport)
  - clipping 영역은 150 X 100
  - window의 영역은 300 X 200
  - viewport는 **창의 전체 영역**으로 설정



  - clipping 영역은 150 X 100
  - window의 영역은 300 X 200
  - viewport는 **150 X 100**

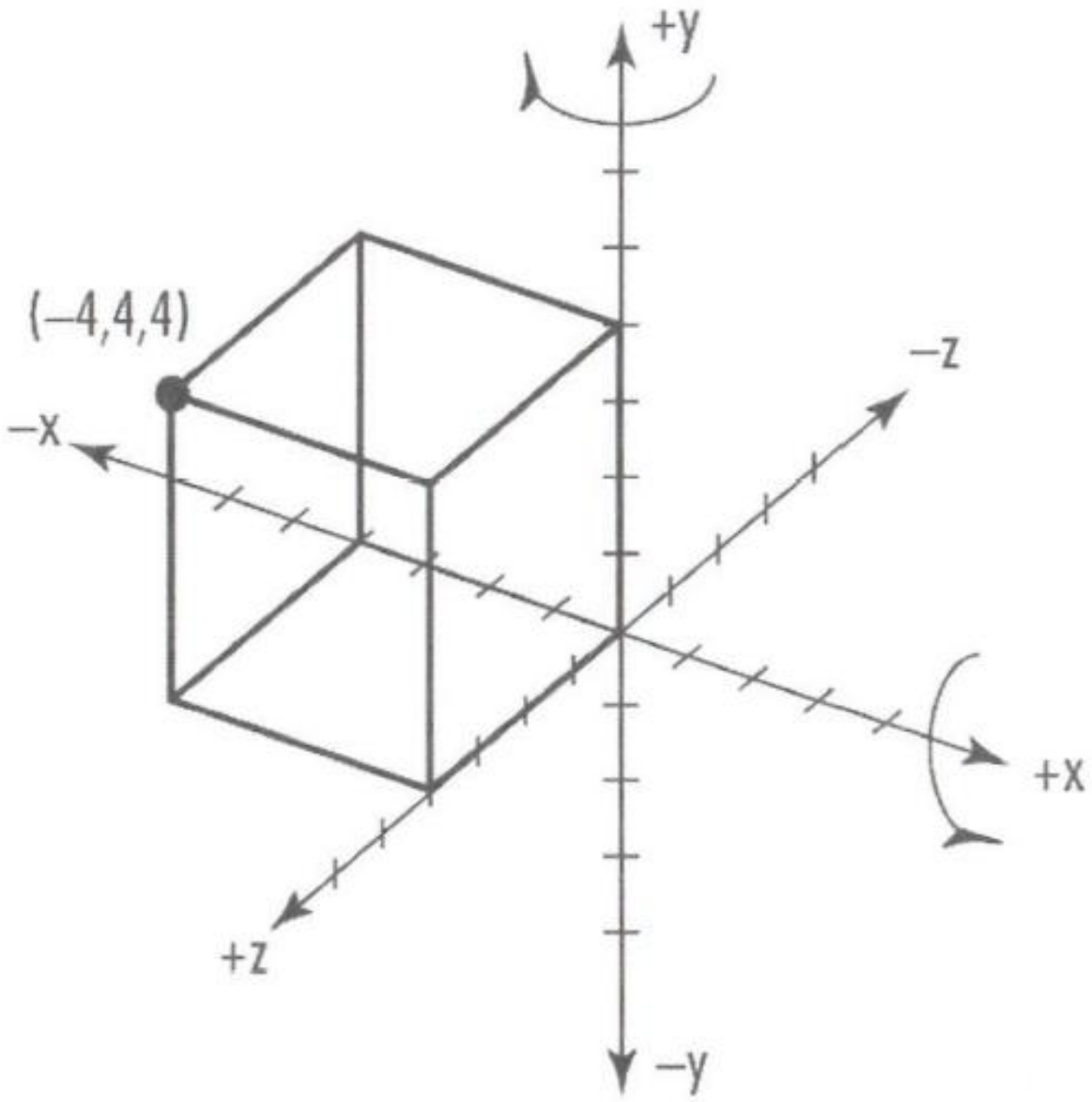
➔**창 영역 전체를 차지하지 못하고 좌측 하단만을 채움**



viewport는 window내에 표시될 이미지의 크기를 확대/축소하는데 사용될 수 있음

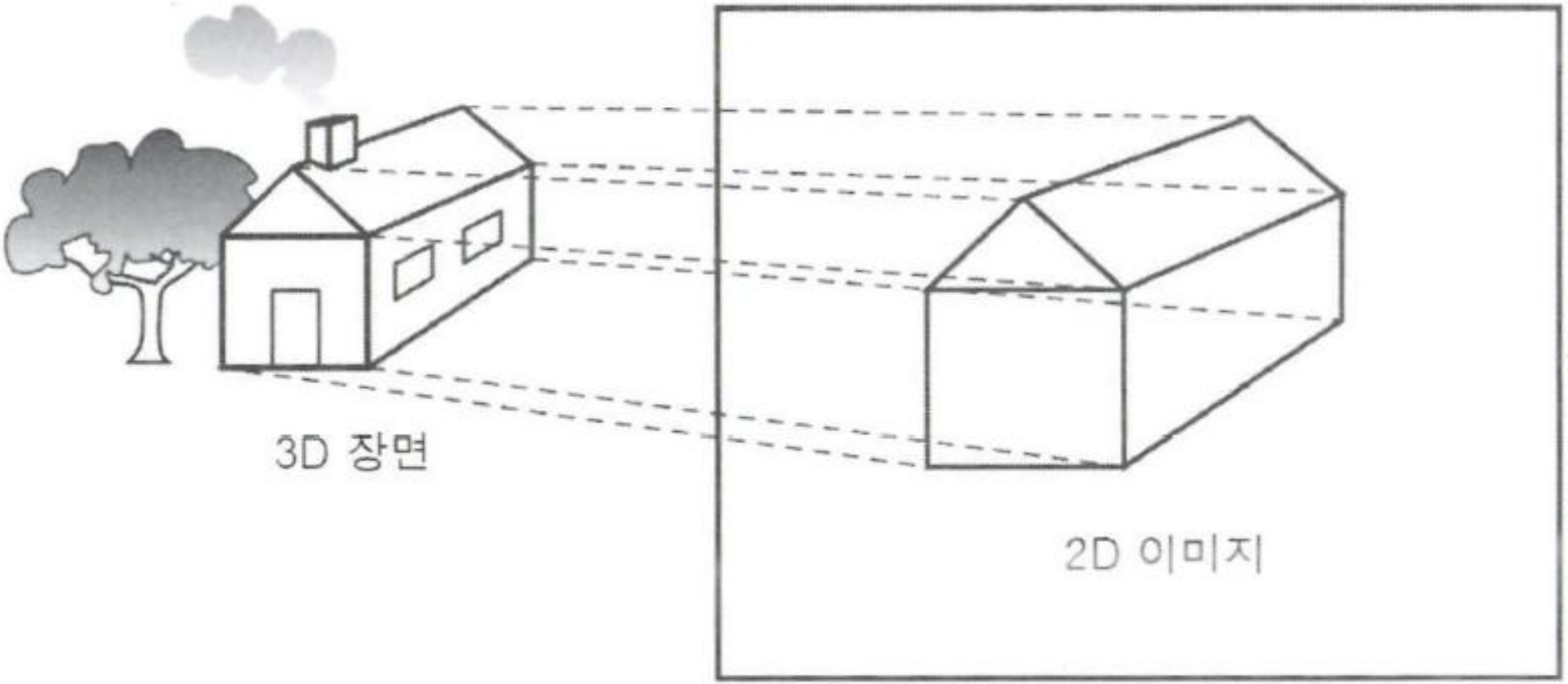
3D 프로그래밍의 기본 지식

- 3D 직교 좌표계



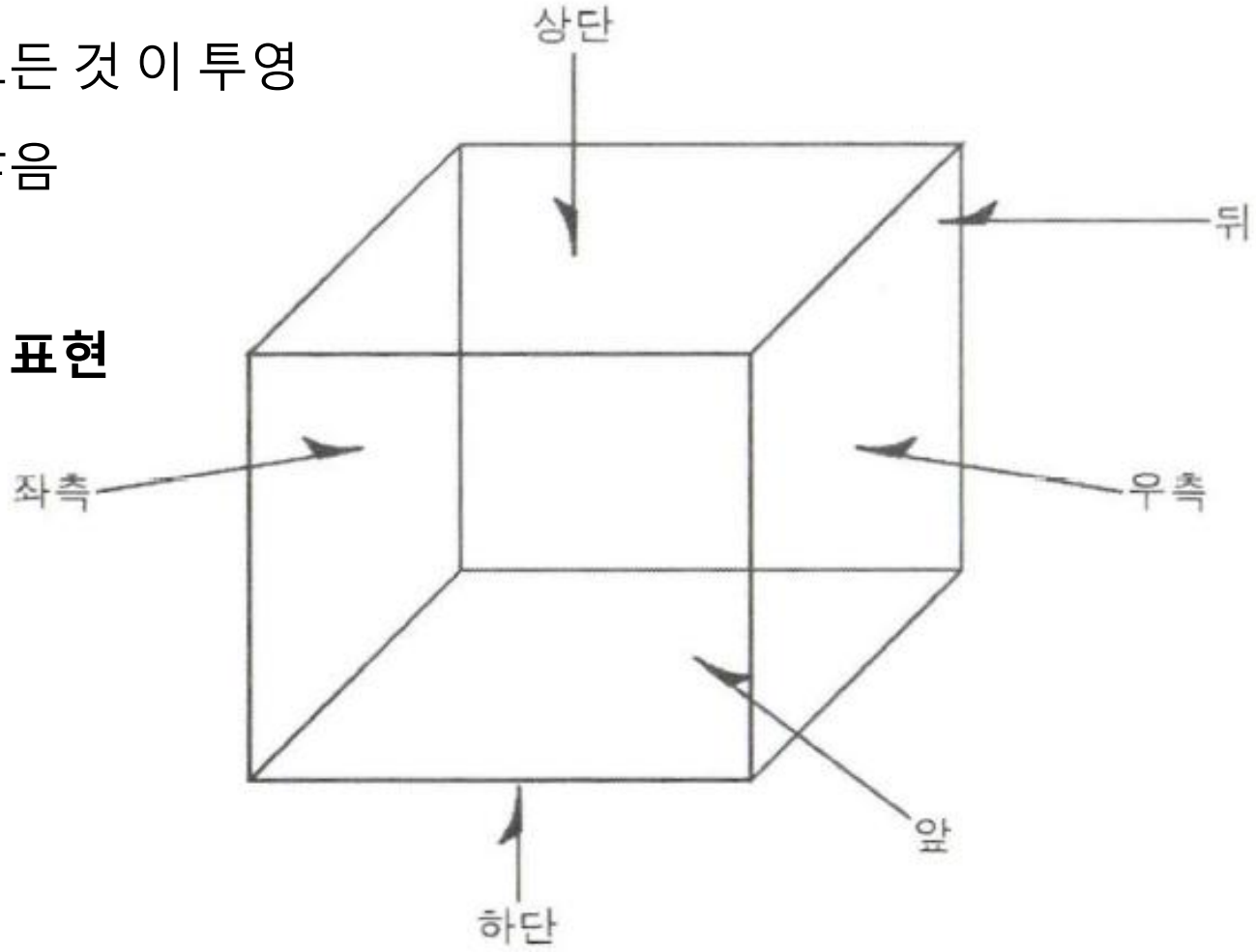
3D 프로그래밍의 기본 지식

- 투영(Projection) : from 3D to 2D



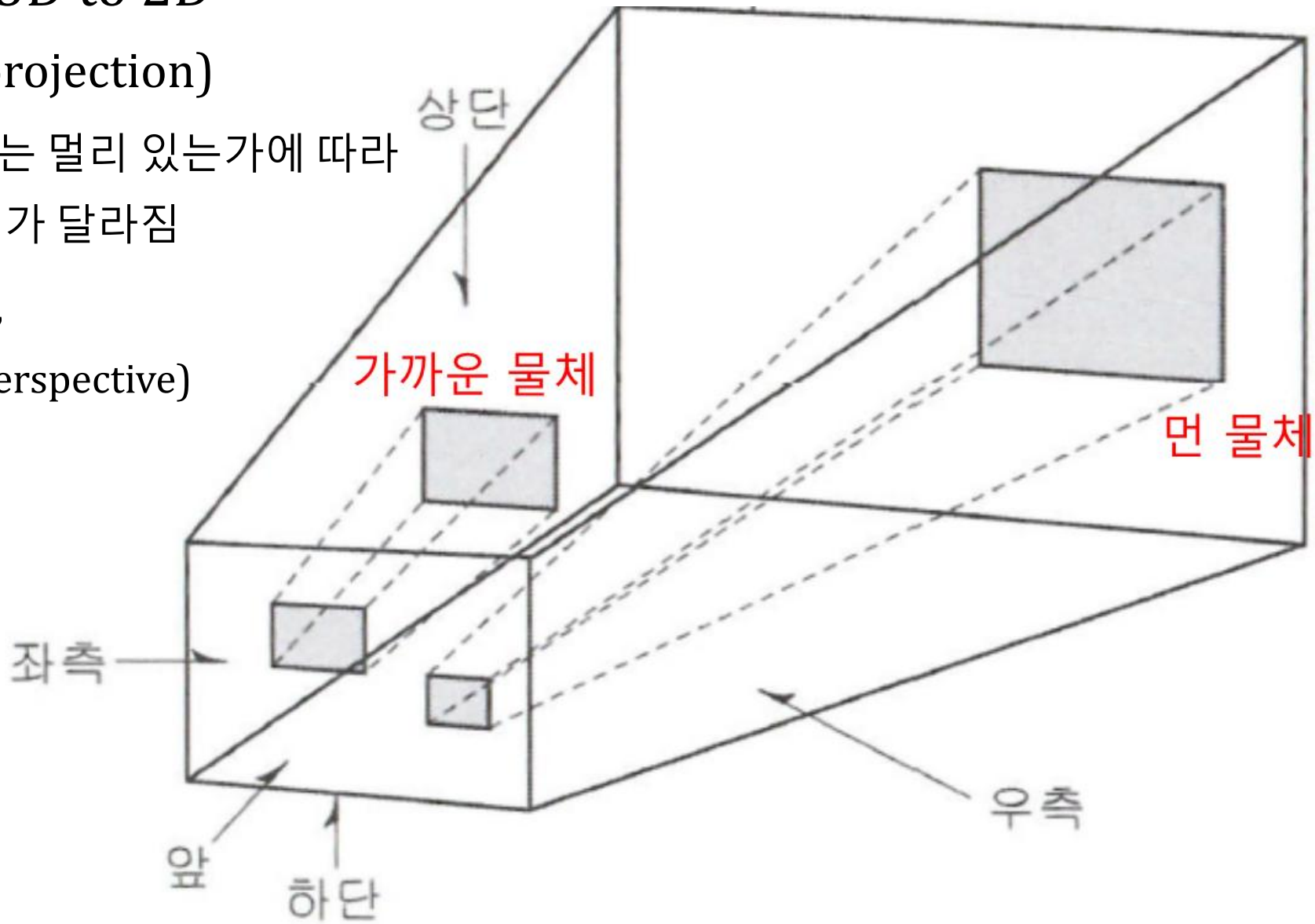
3D 프로그래밍의 기본 지식

- 투영(Projection) : from 3D to 2D
  - 직교 투영(orthographic projection)
    - ✓정사각형 또는 직사각형의 영 역 내에 모든 것 이 투영
    - ✓그 영역 을 벗어나는 것들은 그려지지 않음
    - ✓영역 내에 있는 모든 물체 들이
      거리에 관계없이 모두 **같은 크기 비율로 표현**



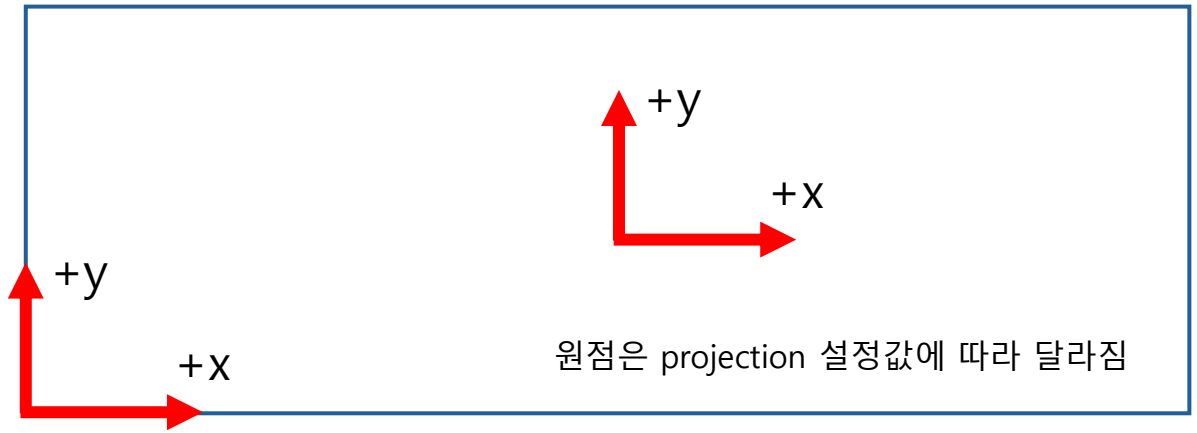
3D 프로그래밍의 기본 지식

- 투영(Projection) : from 3D to 2D
  - 원근 투영 (perspective projection)
    - ✓물체가 얼마나 가까이 또는 멀리 있는가에 따라 2D 화면에 나타나는 크기가 달라짐
    - ✓가까이 있는 물체는 크게, 먼 물체는 작게 나타남(Perspective)



출력 창 좌표 비교

- openGL



- openCV