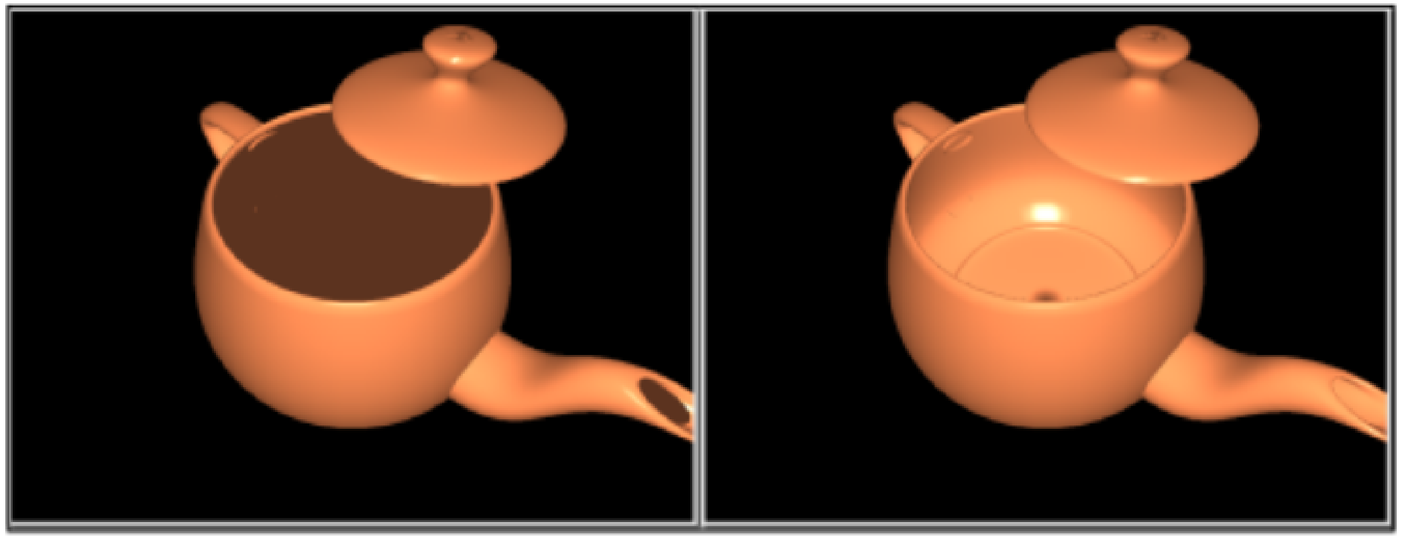
Implementing Two-Sided Shading

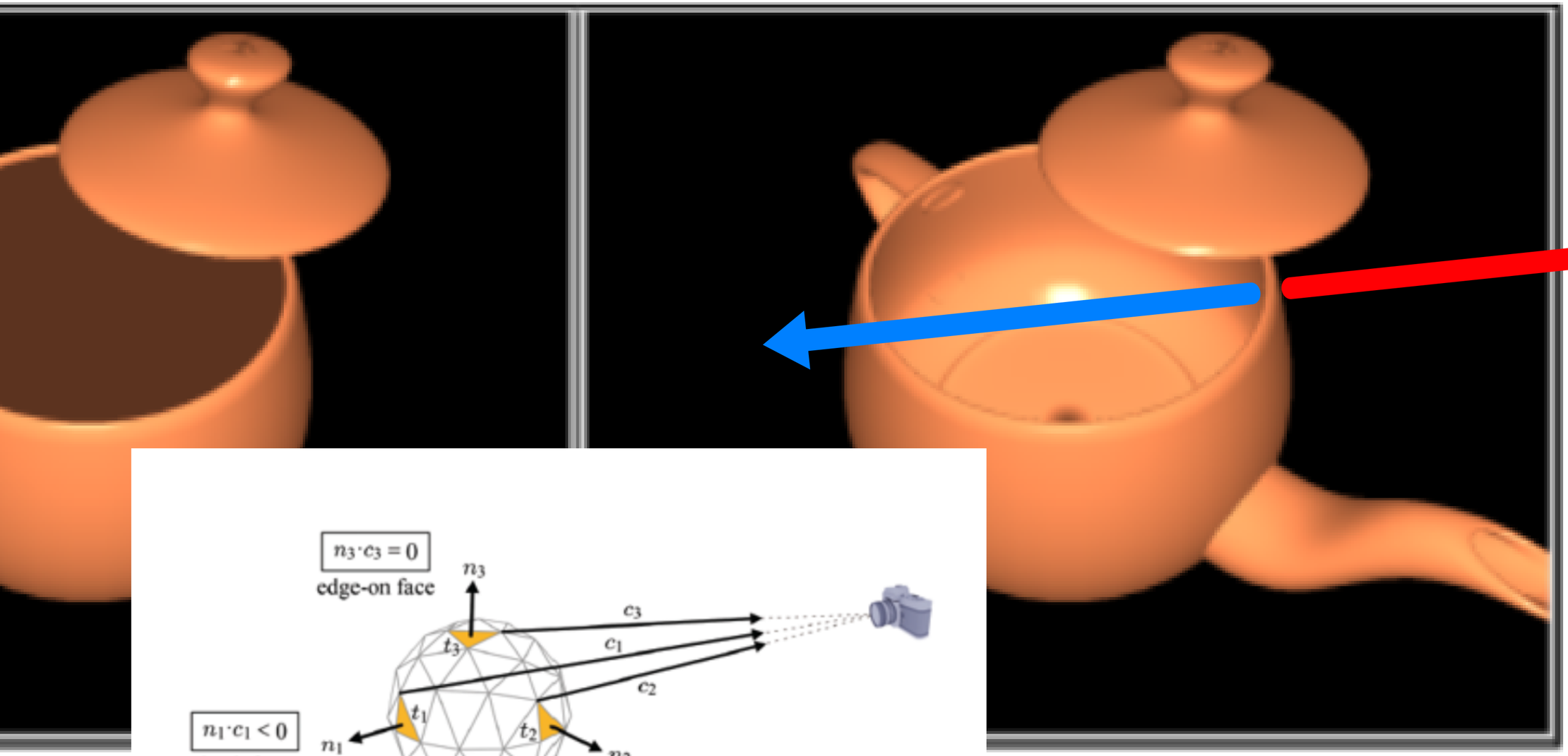
남희수

OpenGL은 기본적으로 래스터화 과정에서 보이는 면과 보이지 않는 면을 계산함.
기본값으로 은면과 후면을 다 제거하는 과정을 거쳐 셰이딩 연산 비용을 줄인다.



은면 : 다른 객체나 면들에 가려져 보이지 않는면들

색상



$n_3 \cdot c_3 = 0$
edge-on face
n_3
c_3
c_1
c_2
t_3
t_1
t_2
$n_1 \cdot c_1 < 0$
back face
n_1
n_2
$n_2 \cdot c_2 > 0$

```
vec3 tnorm = normalize( NormalMatrix * VertexNormal );
vec4 eyeCoords = ModelViewMatrix * vec4(VertexPosition,1.0);
FrontColor = phongModel( eyeCoords, tnorm );
BackColor = phongModel( eyeCoords, -tnorm );
```

후면 판별

gl_FrontFacing

3차원 기준

cx - 카메라 시선 벡터

nx - 삼각형의 법선 벡터

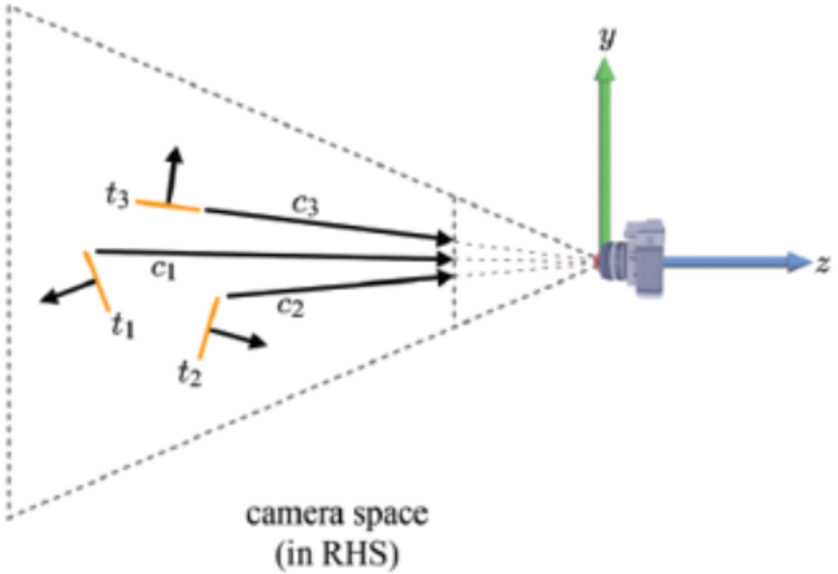
내적은 두 벡터 사이의 각도가 크냐 작냐를 판별하기 좋음 (90도를 기준으로)

cx dot nx 의 값이

음수일 때는 뒷면

0일땐 변만 보이는 삼각형 (edge 판별)

양수일 때는 앞면



2차원 기준

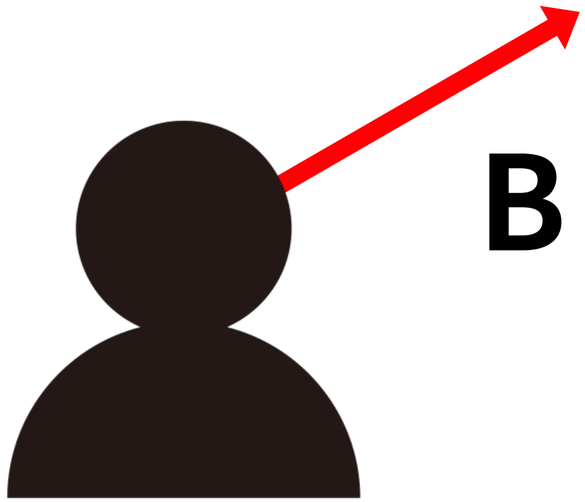
OpenGL 파이프라인에서 정점 데이터를 3차원 > 2차원 매핑이 완료됨.
카메라 좌표계로 변환 후 시점 변환을 한 상태이기 때문에 주어진 x,y 만으로 앞뒷면 판단이 가능하다.

$$\begin{vmatrix}(x_2 - x_1)(y_2 - y_1) \\ (x_3 - x_1)(y_3 - y_1)\end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$$

벡터나 점들이 이루고 있는 면적 및 부피와 방향을 판별할 수 있는 행렬식만으로 앞면,뒷면 판단
-> 양수일 때는 반시계방향, 음수일 때는 시계방향

A

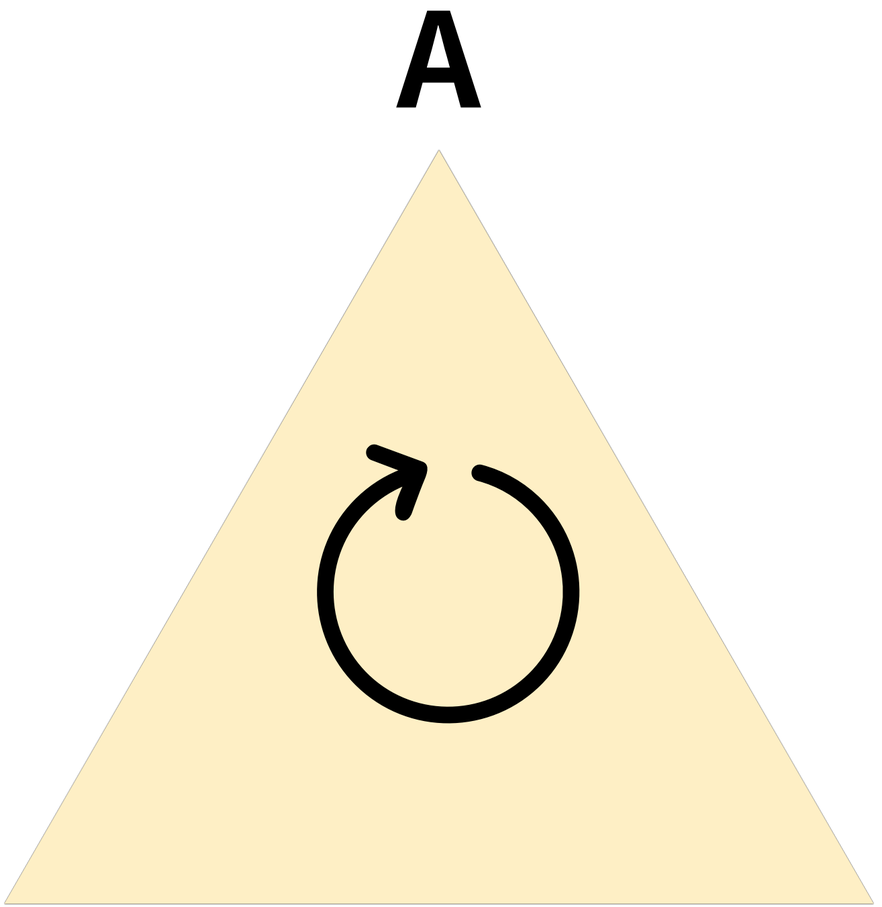
A - B - C

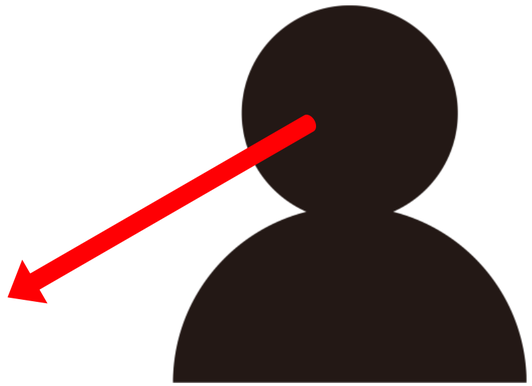


B

C

A - C - B





```
#version 400

in vec3 FrontColor;
in vec3 BackColor;
layout( location = 0 ) out vec4 FragColor;

void main() {
    if( gl_FrontFacing ) {
        FragColor = vec4(FrontColor, 1.0);
    } else {
        FragColor = vec4(BackColor, 1.0);
    }
}
```

glDisable(GL_CULL_FACE)

기본값 :
glEnable(GL_CULL_FACE) < 후면제거 모드
glCullFace(GL_BACK) < 어떤 면을 후면으로 볼 것인지

법선 벡터는 단순히 표면의 방향만 보여주는 역할을 하기 때문에
광원, 그림자, 가려짐과 같은 실제 물리적인 효과는 계산하지 않는다.