Advanced OpenGL / Face Culling

-실제로 큐브 하나를 가지고 최대로 볼 수 있는 면의 수를 생각해보자

-최대로 볼 수 있는 면의 수는 3개를 넘지 않을 것임

-그럼 렌더링 할 때, 나머지 면, fragment들을 버릴 수 있다면 50% 이상의 자원을 절약할 수 있을 것임

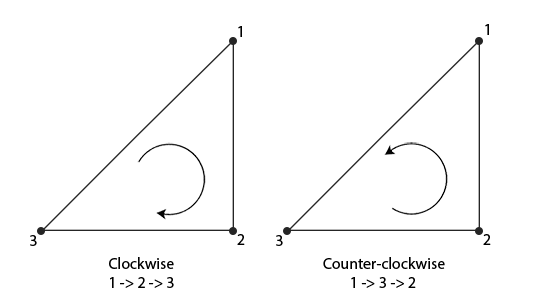
-이에 해결해야 하는 문제는 Object의 어떤 부분이 사용자의 관점에서 보이는가, 보이지 않는가 판단을 해야함

-이러한 작업들을 Face Culling이라고 함

-OpenGL이 뷰어쪽으로 향하는 모든 면을 렌더링, 나머지 면을 폐기하면서 Fragment 호출을 절약함

-이는 정점 데이터의 와인딩 순서 Windinf order을 분석해서 처리가 이루어짐

Winding Order

-삼각형의 vertex를 정의할 때, 시계 방향 또는 시계 반대 방향으로 Winding order를 정의함

|  |
| --- |
| float vertices[] = {  vertices[0], // vertex 1 //시계방향  vertices[1], // vertex 2  vertices[2], // vertex 3  vertices[0], // vertex 1 //반시계방향  vertices[2], // vertex 3  vertices[1] // vertex 2  }; |

-OpenGL이 삼각형이 정면을 바라보는 삼각형인지, 뒤를 향하는 삼각형인지 결정하기 위해 이 정보를 이용함

-기본적으로 반시계방향 vertex로 정의된 삼각형은 정면 삼각형으로 처리됨

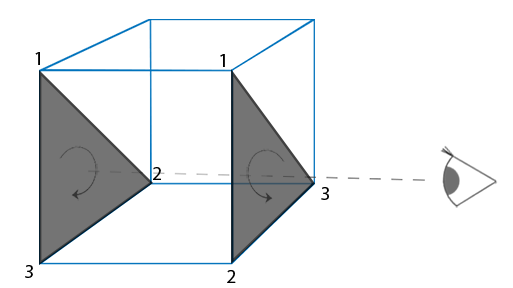
-정점의 순서를 정의할 때, 해당 삼각형이 사용자를 향한 것처럼 시각화함

-따라서 사용자가 지정하는 삼각형은 시계 반대 방향이어야 함

-실제로 rasterization단계에서 와인딩 순서가 계산되어짐

-사용자에게 직면하고 있는 삼각형 vertex들은 모두 지정한 순서대로 렌더링됨

-그러나 반대 방향에 있는 면의 삼각형 vertex들은 반대 방향으로 렌더링되어짐

-이 원리를 이용해서 Face Culling을 할 수 있음

Face Culling

-반 시계 방향 와인딩 순서를 염두한 정점 정의

|  |
| --- |
| float cubeVertices[] = {  // 뒷면  -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, // Bottom-left  0.5f, 0.5f, -0.5f, 1.0f, 1.0f, // top-right  0.5f, -0.5f, -0.5f, 1.0f, 0.0f, // bottom-right  0.5f, 0.5f, -0.5f, 1.0f, 1.0f, // top-right  -0.5f, -0.5f, -0.5f, 0.0f, 0.0f, // bottom-left  -0.5f, 0.5f, -0.5f, 0.0f, 1.0f, // top-left  // 정면  -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, // bottom-left  0.5f, -0.5f, 0.5f, 1.0f, 0.0f, // bottom-right  0.5f, 0.5f, 0.5f, 1.0f, 1.0f, // top-right  0.5f, 0.5f, 0.5f, 1.0f, 1.0f, // top-right  -0.5f, 0.5f, 0.5f, 0.0f, 1.0f, // top-left  -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, // bottom-left  // 왼쪽 면  -0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // top-right  -0.5f, 0.5f, -0.5f, 1.0f, 1.0f, // top-left  -0.5f, -0.5f, -0.5f, 0.0f, 1.0f, // bottom-left  -0.5f, -0.5f, -0.5f, 0.0f, 1.0f, // bottom-left  -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, // bottom-right  -0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // top-right  // 오른쪽 면  0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // top-left  0.5f, -0.5f, -0.5f, 0.0f, 1.0f, // bottom-right  0.5f, 0.5f, -0.5f, 1.0f, 1.0f, // top-right  0.5f, -0.5f, -0.5f, 0.0f, 1.0f, // bottom-right  0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // top-left  0.5f, -0.5f, 0.5f, 0.0f, 0.0f, // bottom-left  // 아랫면  -0.5f, -0.5f, -0.5f, 0.0f, 1.0f, // top-right  0.5f, -0.5f, -0.5f, 1.0f, 1.0f, // top-left  0.5f, -0.5f, 0.5f, 1.0f, 0.0f, // bottom-left  0.5f, -0.5f, 0.5f, 1.0f, 0.0f, // bottom-left  -0.5f, -0.5f, 0.5f, 0.0f, 0.0f, // bottom-right  -0.5f, -0.5f, -0.5f, 0.0f, 1.0f, // top-right  // 윗면  -0.5f, 0.5f, -0.5f, 0.0f, 1.0f, // top-left  0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // bottom-right  0.5f, 0.5f, -0.5f, 1.0f, 1.0f, // top-right  0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // bottom-right  -0.5f, 0.5f, -0.5f, 0.0f, 1.0f, // top-left  -0.5f, 0.5f, 0.5f, 0.0f, 0.0f // bottom-left  }; |

-현재 정점들은 반시계 방향으로 되어있음

-Face Culling 활성화

|  |
| --- |
| glEnable(GL\_CULL\_FACE); |

-앞면이 아닌 다른 면이 버려짐(기본 값)

-앞에 그린 Blending한 풀을 그릴려면 다시 해제해줘야 함

-버리고 싶은 면의 유형을 바꿀 수 있음

|  |
| --- |
| glCullFace(GL\_FRONT); |

-GL\_FRONT : 앞면만 제거

-GL\_BACK : 뒷면만 제거

-GL\_FRONT\_AND\_BACK : 앞 뒤 모두 제거

-정면이 어떤 방향인지 알려줄 수 있음

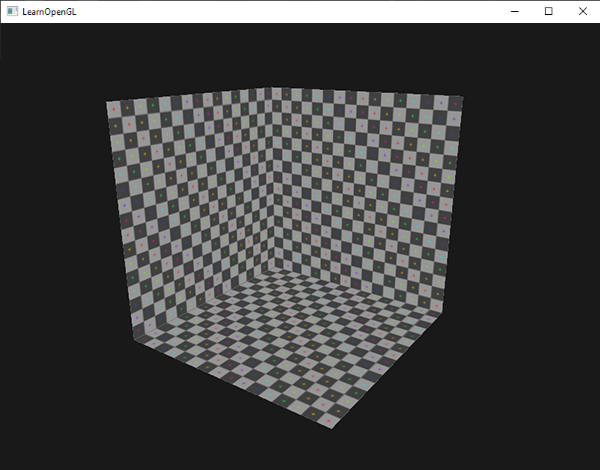
|  |
| --- |
| glFrontFace(GL\_CCW); //GL\_CW가 시계 방향, 기본값은 CCW |

-현재 반시계 방향이 정면임을 알려줌

|  |
| --- |
| glEnable(GL\_CULL\_FACE);  glCullFace(GL\_BACK);  glFrontFace(GL\_CW); |

-Culling을 활성화 하고, 뒷면을 제거, 정면은 시계방향으로 설정

-따라서 뒷면인 반시계 방향이 제거가 되는데, 현재 큐브의 정점들은 반시계로 되어 있기에 뒷면만 렌더링이 됨



-이를 정면만 보이게 렌더링 할려면

1. 정점의 정의에서 시계방향을 정의하기

2. 코드를 변경하기

로 정면만 보이게 할 수 있을 것 같다.

보이지 않는 면까지 모두 렌더링을 하는 것 보다는 확실히 보이는 면만 렌더링 하는 방법이 장면을 최적화 하여 렉을 줄일 수 있는 수단이 될 것같다.