Advanced OpenGL / Depth Testing

-Fragment shader가 Fragment를 처리하고 난 뒤, 조각을 파기할 수 있는 스텐실 테스트가 실행됨, 그 후 깊이 테스트로 넘어감

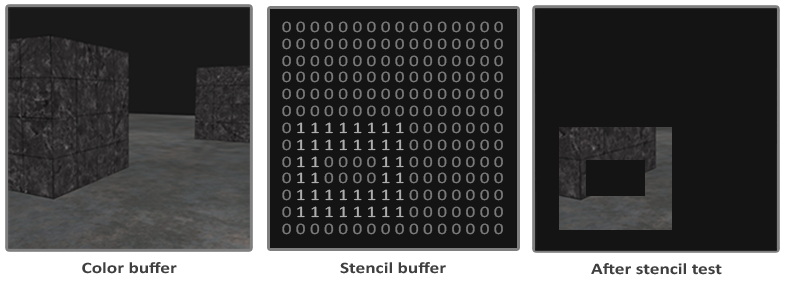
-렌더링 중에 스텐실 버퍼는 업데이트가 됨

-스텐실 값 당 8피트를 포함, pixel/fragment 당 총 256개의 스텐실 값을 가짐

-스텐실 값을 원하는 값으로 설정하면 특정 조각에 대해 조각을 버리거나 보관할 수 있음

-GLFW는 스텐실 버퍼를 자동으로 수행하지만, 다른 윈도우 라이브러리는 스텐실 라이브러리를 만들지 않음

-스텐실 버퍼의 예제

-스텐실 버퍼 연산을 통해 렌더링하는 곳에서 특정 값으로 설정이 가능

-렌더링 하는 동안, 스텐실 버퍼의 내용을 변경하면 다음 렌더링 때 버퍼를 읽어 특정 조각을 삭제하거나 전달할 수 있음

-스텐실 버퍼의 개요

-스텐실 버퍼 쓰기를 활성화함

-스텐실 버퍼의 내용을 업데이트, 객체를 렌더링함

-스텐실 버퍼 쓰기를 비활성화 함

-다른 객체를 렌더링해 스텐실 버퍼의 내용을 기반으로 특정 조각을 버림

-스텐실 테스트 활성화 코드

|  |
| --- |
| glEnable(GL\_STENCIL\_TEST); |

-렌더링에 glClear

|  |
| --- |
| glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT |  GL\_STENCIL\_BUFFER\_BIT); |

-스텐실 마스크는 비트연산으로도 비트 마스크를 버퍼에 기록하도록 할 수 있

|  |
| --- |
| glStencilMask(0xFF); // 그리기  glStencilMask(0x00); // 그리지 않기 |

Stencil functions

-스텐실 테스트를 구성하는데 사용할 수 있는 함수

-glStencilFunc(GLenum func, GLenum ref, GLenum mask)

-func : 기능을 설정/ GL\_NEVER, GL\_LESS, GL\_LEQUAL, GL\_GREATER, GL\_GEQUAL, GL\_EQUAL, GL\_NOTEQUAL, GL\_ALWAYS

-ref : 참조 값을 지정, 버퍼의 내용이 ref값과 비교됨

-mask :테스트가 값을 비교하기 전, AND된 마스크를 지정, 모두 1로 초기화 되어있음

|  |
| --- |
| glStencilFunc(GL\_EQUAL, 1, 0xFF) |

-Fragment의 스텐실 값이 참조 1 와 같을 때를 알려줌

-glStencilOp(GLenum sfail, GLenum dpfail, GLenum dppass)

-실제로 버퍼를 업데이트 하는 함수

-sfail : 테스트가 실패할 경우 수행되는 작업

-dpfail : 테스트가 통과했으나, 깊이 테스트가 실패할 경우 수행되는 작업

-dppass : 스텐실, 깊이 모두 통과할 경우 수행되는 작업

-옵션들

-GL\_KEEP : 저장된 스텐실 값이 유지

-GL\_ZERO : 스텐실 값이 0으로 설정

-GL\_REPLACE : glStencilFunc의 설정된 기준 값으로 대체

-GL\_INCR : 최대값보다 작을 경우, 1 증가

-GL\_INCR\_WRAP : INCR과 동일, 최대값보다 초과할 경우에는 0으로 랩핑

-GL\_DECR : 최솟값보다 클 경우, 1 감소

-GL\_DECR\_WARP : DECR과 동일, 0보다 미만일 경우 최댓값으로 랩핑

-GL\_INVERT : 현재 스텐실 버퍼값을 반전함

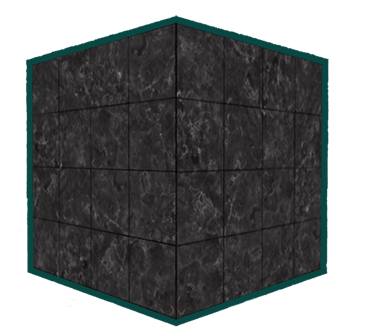
-기본 설정되어진 옵션은 GL\_KEEP로 설정되어 있음

-두 함수를 사용해 스텐실 버퍼를 업데이트 할 시기와, 방법을 정확하게 원하는 시간에 지정할 수 있음

-fragment를 삭제할 때, 테스트가 통과해야하는지도 지정할 수 있음

Object outlining 오브젝트 테두리 그리기

-스텐실 테스팅으로 구현할 수 있는 유용한 기능

-보이는 것과 같이 오브젝트를 선택했을 때, 주위에 작은 컬러 테두리를 만들 수 있음

-1. 스텐실 함수를 GL\_ALWAYS로 설정, 버퍼를 1로 업데이트

-2. 개체를 렌더링함

-3. 스텐실 쓰기, 깊이 테스트를 비활성화

-4. 객체의 크리를 약간 조정

-5. 단일 색상을 출력하는 다른 fragment shader을 사용함

-6. 조각의 스텐실 값이 1이 아닌 경우에만 오브젝트를 다시 그림

-7. 스텐실 쓰기, 깊이 테스트를 활성화

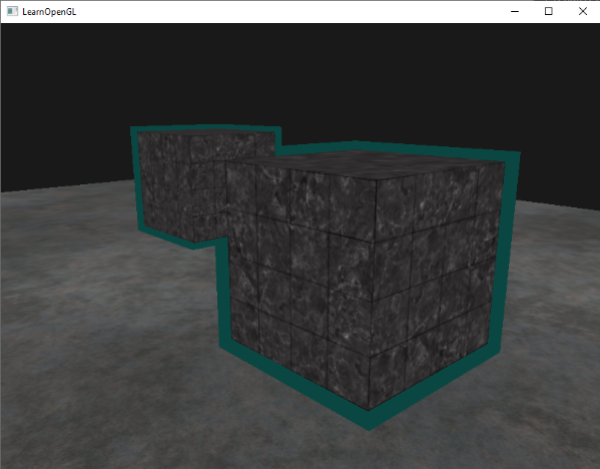
|  |
| --- |
| -오브젝트보다 살짝 큰, 같은 위치에 새로운 오브젝트를 만들며, 이를 다른 shader을 사용해 원하는 단일 색상을 표시하게 하고 이를 Stencil 테스트를 통해 겹친부분(스텐실 값이 1)이 아닌 부분을 그리는 방법인 것 같이 보임 |

-단일 색 표시하는 쉐이더

|  |
| --- |
| void main()  {  FragColor = vec4(0.04, 0.28, 0.26, 1.0);  } |

-렌더링 내부

|  |
| --- |
| glEnable(GL\_DEPTH\_TEST);  glStencilOp(GL\_KEEP, GL\_KEEP, GL\_REPLACE);    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT |  GL\_STENCIL\_BUFFER\_BIT);  glStencilMask(0x00); // 바닥을 그릴 때는 업데이트 하지 않아야 함  normalShader.use();  DrawFloor()    glStencilFunc(GL\_ALWAYS, 1, 0xFF);  glStencilMask(0xFF);  DrawTwoContainers();    glStencilFunc(GL\_NOTEQUAL, 1, 0xFF);  glStencilMask(0x00);  glDisable(GL\_DEPTH\_TEST);  shaderSingleColor.use();  DrawTwoScaledUpContainers();  glStencilMask(0xFF);  glStencilFunc(GL\_ALWAYS, 1, 0xFF);  glEnable(GL\_DEPTH\_TEST); |

-현재 테두리가 다 합쳐져 있는 것을 볼 수 있음

-이를 개체당 완전한 테두리를 원한다면 스텐실 버퍼를 지우고, 깊이 버퍼로 설정해야함

-깊이 버퍼로 앞에 있는 것을 표시하면 뒤에 것은 보이지 않고, 앞에 것이 보이게 됨

-또한 테두리가 딱딱한 것이 싫다면, Gaussian Blur같은 후처리 필터를 사용해 부드럽게 만들 수 도 있음