오승철

MonumentValleyCloneCoding Project

1. 프로젝트 개요

* 본 프로젝트는 monument valley라는 게임의 level1을 clone coding한 것입니다.
* Monument valley는 퍼즐 게임이며 원근감을 제거하여 나타나는 착시를 이용한 게임입니다.
* opengl의 렌더링 파이프라인을 보다 더 깊게 이해하고 익숙해지기 위해서 또한 C++의 class 활용 연습을 위해 프로젝트를 진행하였습니다.
* 깃 주소 : <https://github.com/secul5972/MonumentValleyCloneCoding>
* 영상 노션 : <https://ripe-newsboy-b56.notion.site/MonumentValley-010f811ab10e48cf933eb1dc01e168ed>

1. 프로젝트 작성 환경

* visual studio 2022 17.3.4, glfw 3, opengl 3.3버전을 사용하여 작성되었습니다.
* Source tree 프로그램을 이용해서 github에서 버전 관리를 하였습니다.

1. 구현 사항

※주의 : 그림 아래에 그림에 대한 설명이 있습니다.

1. 코드 확장을 고려해서 백터, 공간도형연산들을 일반적인 경우를 가정해서 구현하였습니다.
2. 원근감 제거

- 원근감 제거를 위해 원근투영이 아닌 직교투영을 사용하였습니다.

- orthogonal projection matrix는 아핀 변환이므로 역행렬을 구하여 마우스 클릭 좌표를 WC 또는 MC의 좌표로 변환하는 것이 가능해 집니다.

- 이를 이용해서 마우스로 특정 영역 안을 클릭하는 것을 인식할 수 있게 됩니다.

1. Object

- 화면에 표시될 입체 도형을 의미합니다.

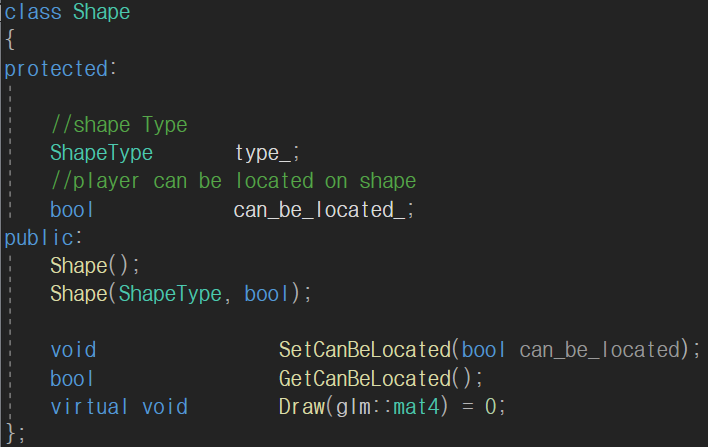
- Object는 ActerCanGoObject와 Ornament class중 하나를 상속합니다.

- ActerCanGoObject는 acter가 움직일 수 있는 object 말합니다.

- Ornament는 나머지 장식용 도형을 의미합니다.

- ActerCanGoObject는 Shape, Rotatable, CanBeLocated class를 상속합니다.

* Shape

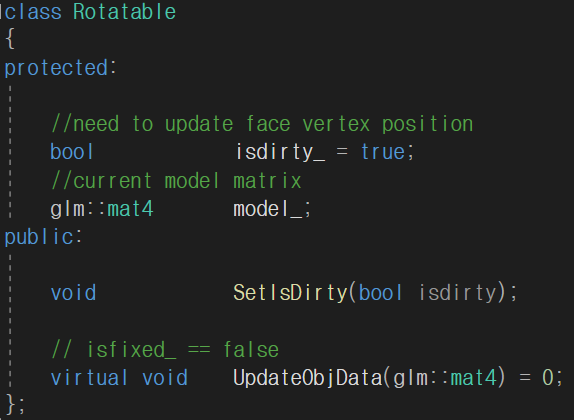


- 도형의 기본정보를 저장합니다.

- 도형의 type과 도형 위에 acter가 위치하는 것이 가능한지 나타내는 변수가 있으며 그에 대한 set, get함수를 가집니다.

- draw는 도형을 그리는 함수입니다. 순수가상함수를 이용해서 모든 도형에서 재정의하여 사용할 수 있도록 하였습니다.

* Rotatable



- acter의 이동을 구현하기 위해서 도형이 회전하였을 때 회전에 대한 정보를 저장하기 위한 class 입니다.

- 기본적으로 물체가 움직이면 model matrix에 아핀변환을 가하는 것으로 구현하며 물체의 초기 vertex 정보는 변하지 않습니다.

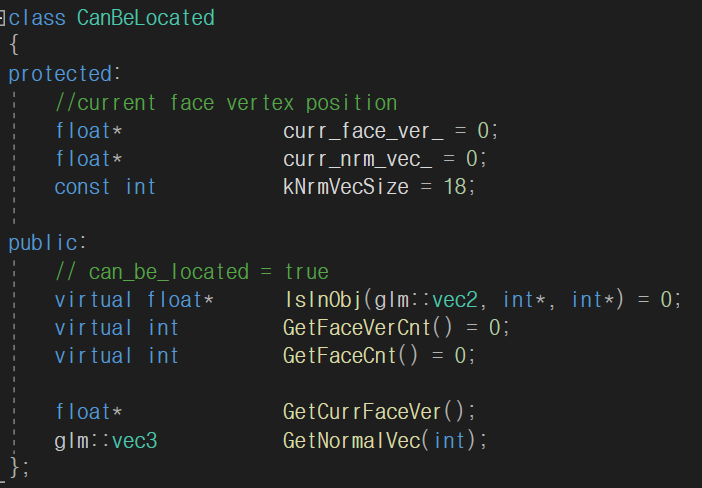
- 하지만 acter가 움직일 때는 물체가 움직인 후의 좌표가 필요하므로 이를 계산해줘야 합니다.

- isdirty변수는 도형이 움직여서 도형의 vertex들을 갱신해줘야 함을 의미하며 model은 항상 도형에 적용되어야할 model matrix를 의미합니다.

- SetIsDirty를 이용해서 isdirty를 on하며, UpdateObjData를 이용해서 Object의 필요한 정보를 저장합니다.

- 이때 모든 vertex를 갱신하는 것이 아니라 도형의 면과 법선 백터 정보만 갱신합니다.

* CanBeLocated

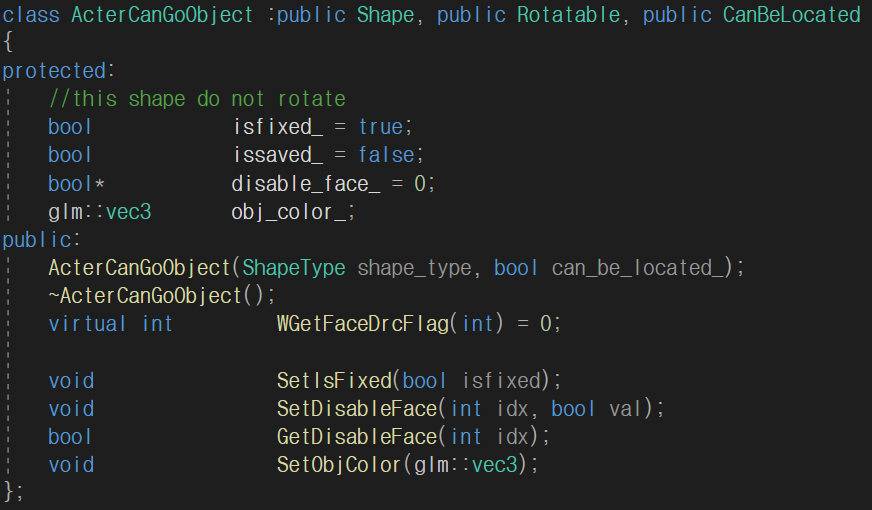


- 현재 렌더링된 상태의 면과 법선백터의 vertex를 이곳에서 저장합니다.

- curr\_face\_vertex에서 원하는 vertex를 찾는 연산을 위해 face의 vertex 개수와 face의 개수, curr\_face\_vertex를 return 하는 함수와 normal vector를 return해주는 함수를 가집니다.

- IsInObj는 마우스 클릭 좌표를 받아서 해당 좌표가 어떤 면 위에 존재하는지 찾는 함수 입니다. 해당하는 면이 존재하면 면의 첫 vertex의 포인터를 return하고 해당 면의 번호와 면 위에서 이동 가능한 방향을 포인터 파라메터를 이용해 전달합니다.

* ActerCanGoObject



- 도형들을 배열로 관리하기 위한 class 입니다.

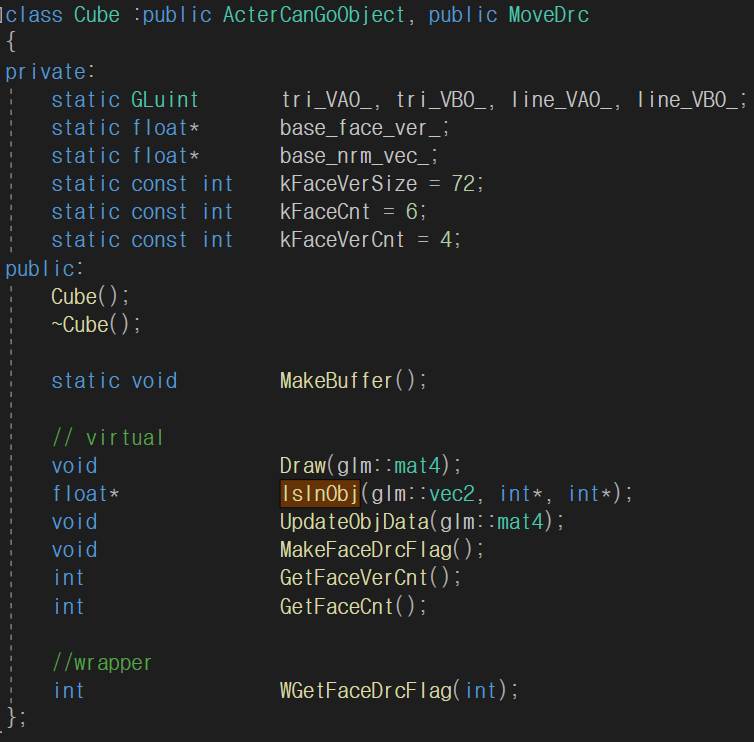
- Isfixed는 도형이 고정되어 있는지 저장합니다.

- 고정된 도형의 움직이지 않으므로 렌더링된 면과 법선백터의 정보를 한번만 저장하면 되는데 이때 issaved를 사용합니다.

- disable\_face는 도형과 도형이 겹쳐 있을 경우 이동이 불가능한 면이 생기는데 록 이 면들을 비활성화 시켜서 acter의 이동 경로를 구할 때 사용하지 못하도록 합니다.

- obj\_color은 도형을 색을 저장합니다.

* ActerCanGoObject를 상속하는 도형 예시



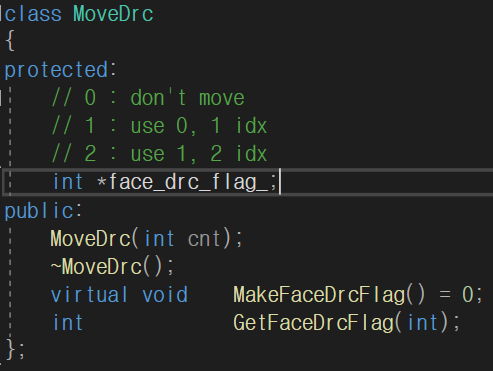
- VAO와 VBO를 삼각형과 도형의 테두리를 그릴 선으로 구분하여 저장하며 MakeBuffer에서 수행합니다.

- curr\_face\_ver를 구하기 위한 base\_face\_ver와 curr\_nrm\_vec를 구하기위한 base\_nrm\_vec입니다.

- face에 대한 정보를 가진 상수들도 저장합니다.

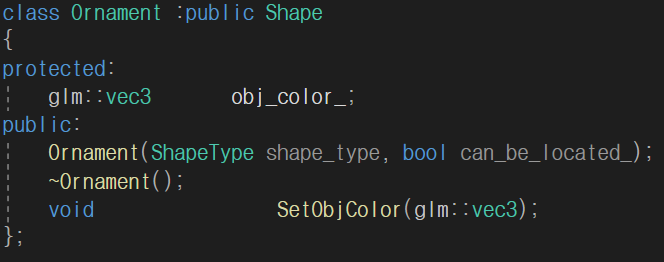
- 설명한 값들은 Cube라는 class에 단 하나씩만 필요하므로 static으로 지정했습니다.

* MoveDrc



- 면에서 acter가 이동할 direction vector를 만들기 위한 정보입니다.

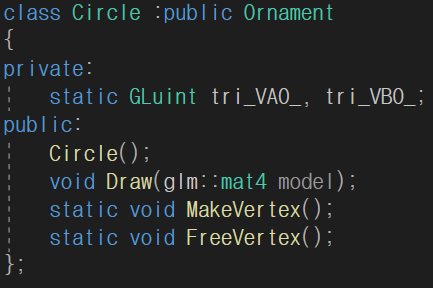
* Ornament



- Ornament는 Shape만 상속합니다.

- 특이한 동작이 없으므로 object의 색깔만 저장합니다.

* Ornament를 상속하는 class 예시



- 일반적으로 기본적인 draw함수만 존재하며 circle과 sphere 같은 경우 다른 도형의 기본 vertex가 되므로 vertex를 만들어 주는 함수가 추가되어있습니다.

1. 마우스 드래그 인식과 인식 영역 설정

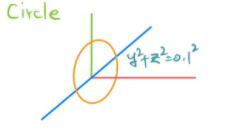
<https://ripe-newsboy-b56.notion.site/MonumentValley-010f811ab10e48cf933eb1dc01e168ed>

1번 영상

- 영상과 같이 정해진 영역을 드래그하면 물체가 움직입니다.

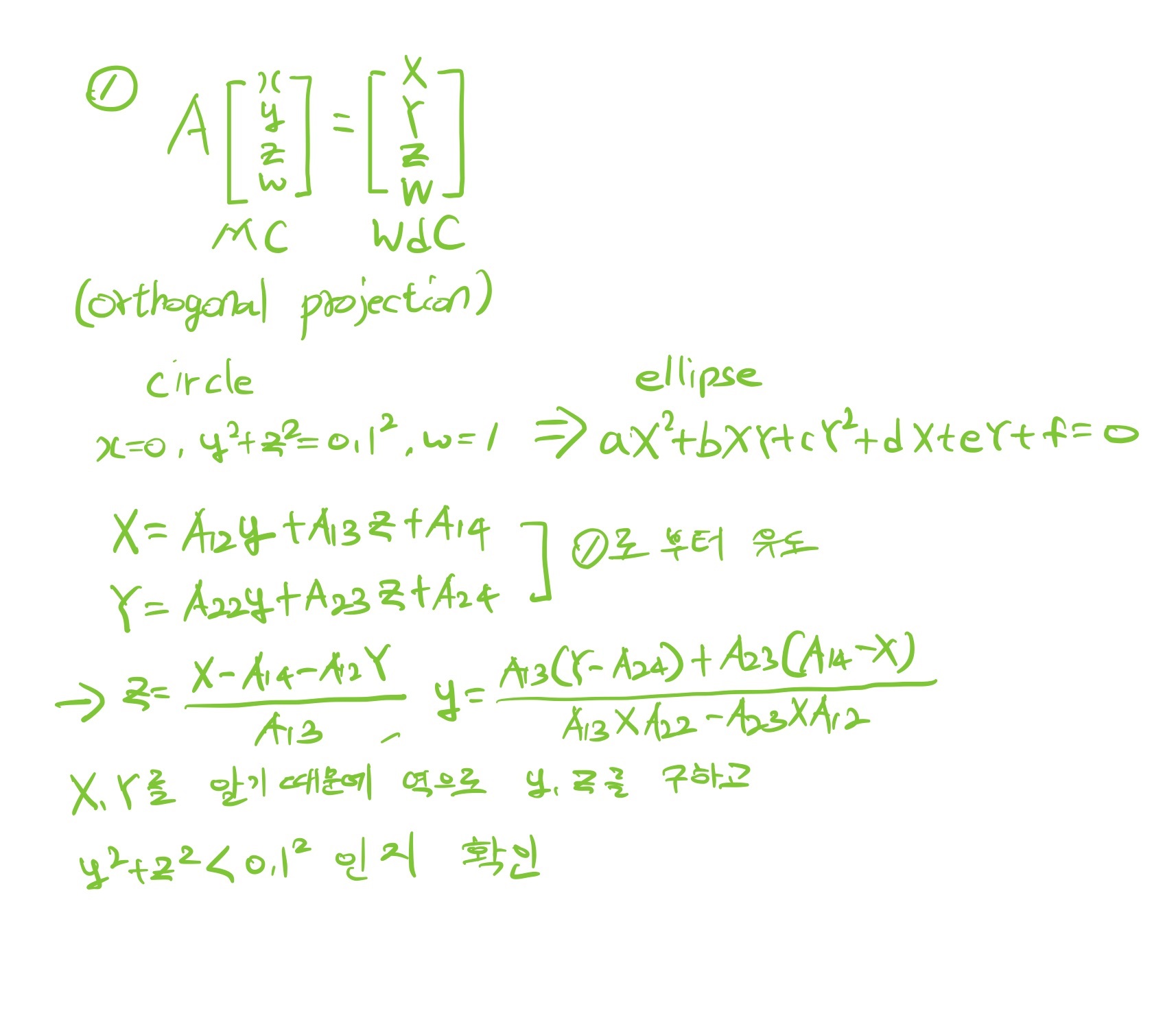
- 회전할 도형 위에 acter가 있지 않은지 확인 후에 드래그한 시작점과 도착점을 이용해서 회전각을 구합니다.

- 클릭을 인식할 영역은 MC에서 y,z평면의 원이며 WdC에서는 타원으로 변환됩니다.

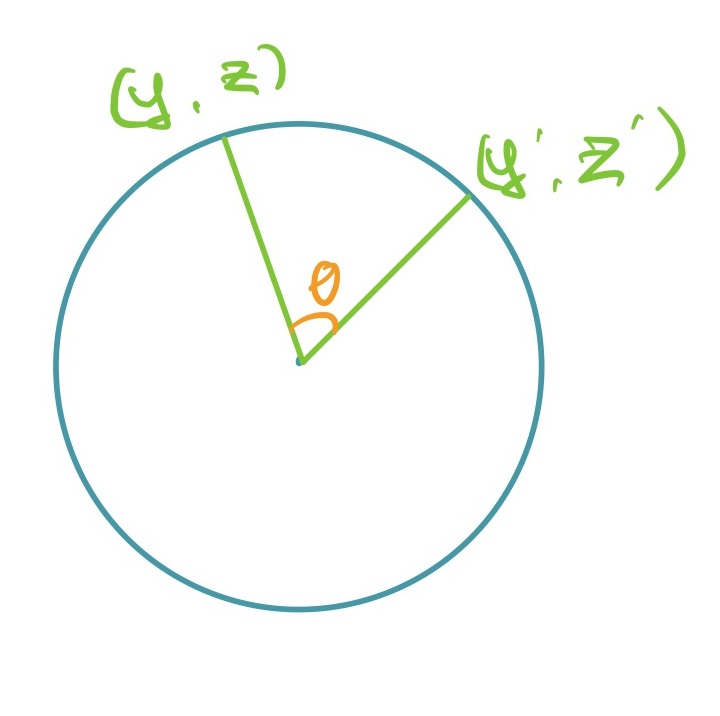


- 직교투영은 matirx가 아핀변환인 것과 원의 x값이 0인 것을 이용해서

다음과 같이 마우스 클릭 위치(WdC)를 원의 MC 좌표계로 변환하는 식을 유도했습니다.



- 위와 같이 y, z좌표를 구하고 좌표가 원 내부에 존재하면 이전 프레임에 구했던 y, z좌표와 내적을 해서 각도를 구합니다.

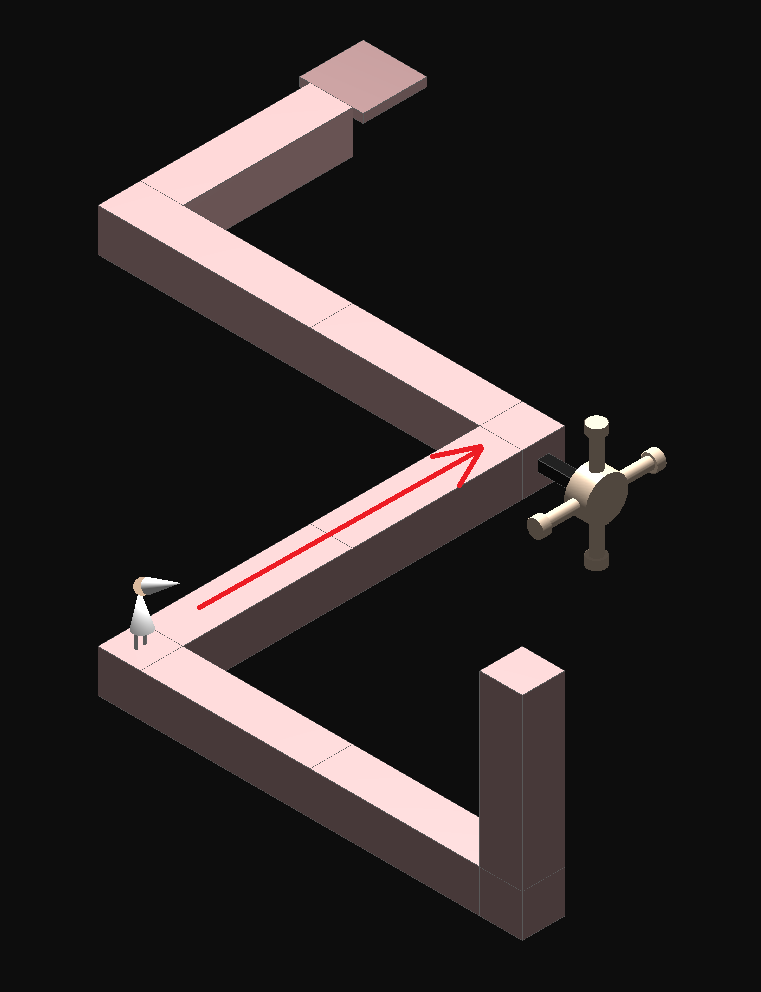


- 구한 각도는 rotate\_obj\_angle에 더해지며 rotate\_obj\_angle은 회전하는 물체의 각도를 의미합니다.

- acos으로 각을 구할 때 nan이 발생할 수 있으므로 예외 처리 해줍니다.

- (0, y, z)와 (0, y’, z’)의 외적을 구해서 각도의 방향(부호)를 정해줍니다.

- rotate\_obj\_angle의 각도가 270도 일 때는 그림과 같이 goal이 존재하는 지형으로 이동 가능 해야합니다.



- 이를 위해서 움직인 도형 위쪽으로 연결된 도형들은 225 < 각도< 315에서 (-1.8f, -1.8f, -1.8f)만큼 이동하며 시각적으로는(빛 효과 제외하고) 동일하지만 acter가 이동이 가능하게 됩니다.

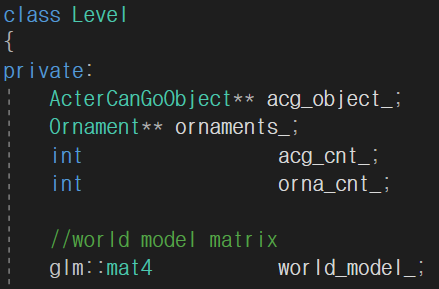
- 또한 90도 단위가 아닌 각들에서 드래그를 중단하게 되면 자동으로 0, 90, 180, 270도중 가까운 각으로 서서히 움직입니다.

- 마지막으로 도형이 움직였으므로 해당 도형의 면과 법선 백터 정보를 갱신해 줍니다.

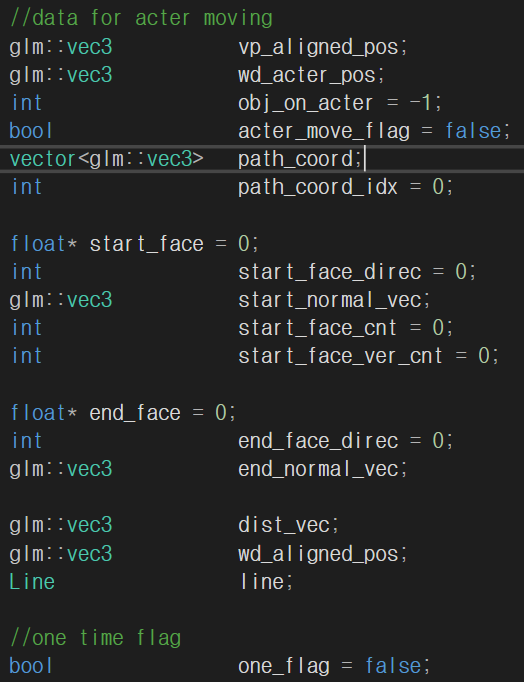
1. Level

- Level class는 게임에서 한 단계를 구성합니다.

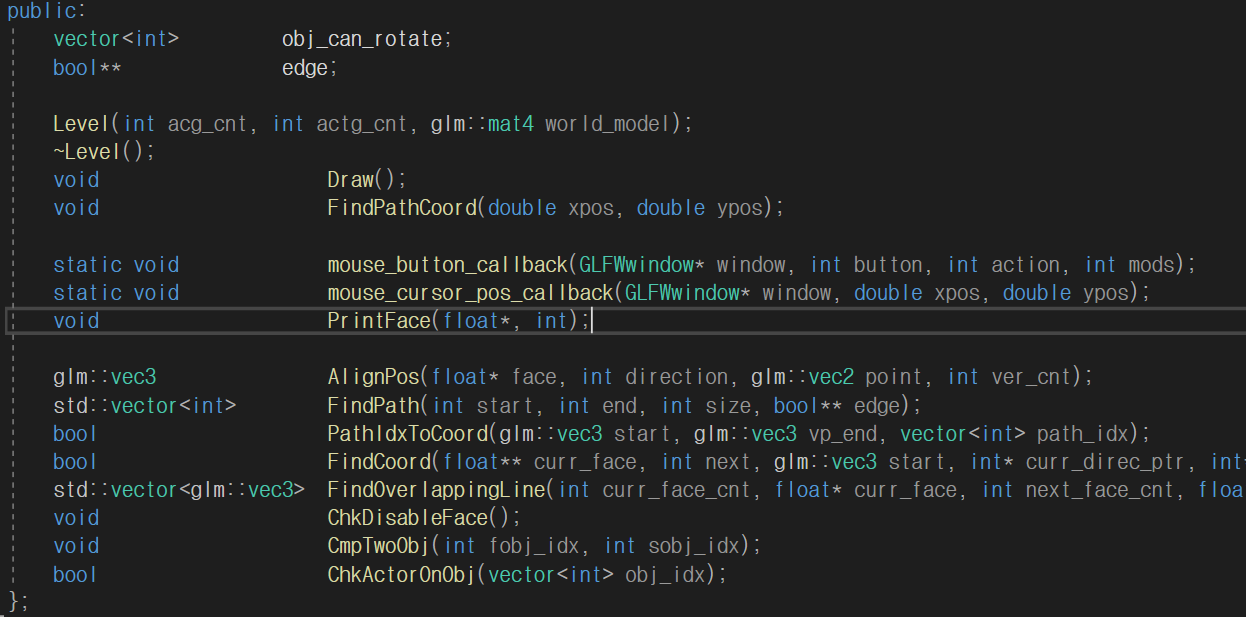
- object, acter 움직임을 관리하며 이를 구현 시 필요한 변수들을 포함하고 있습니다.



- 도형의 포인터와 개수, Level의 world 상 위치를 저장합니다.



- acter의 움직임에 필요한 정보와 이동경로를 그릴 line, 한 번만 실행을 위한 flag를 저장합니다.



- obj\_can\_rotate는 acter가 회전가능한 도형 위에 있지 않은지 확인하기 위한 변수이며 edge는 acter가 이동가능한 경로를 나타내는 인접배열입니다.

- 함수는 acter 움직임에서 설명하겠습니다.

1. Acter 움직임

<https://ripe-newsboy-b56.notion.site/MonumentValley-010f811ab10e48cf933eb1dc01e168ed>

2번 영상들

- 마우스로 클릭을 하면 acter가 움직이고 있는지 확인 후 FindPathCoord 함수로 이동합니다.

* FindPathCoord

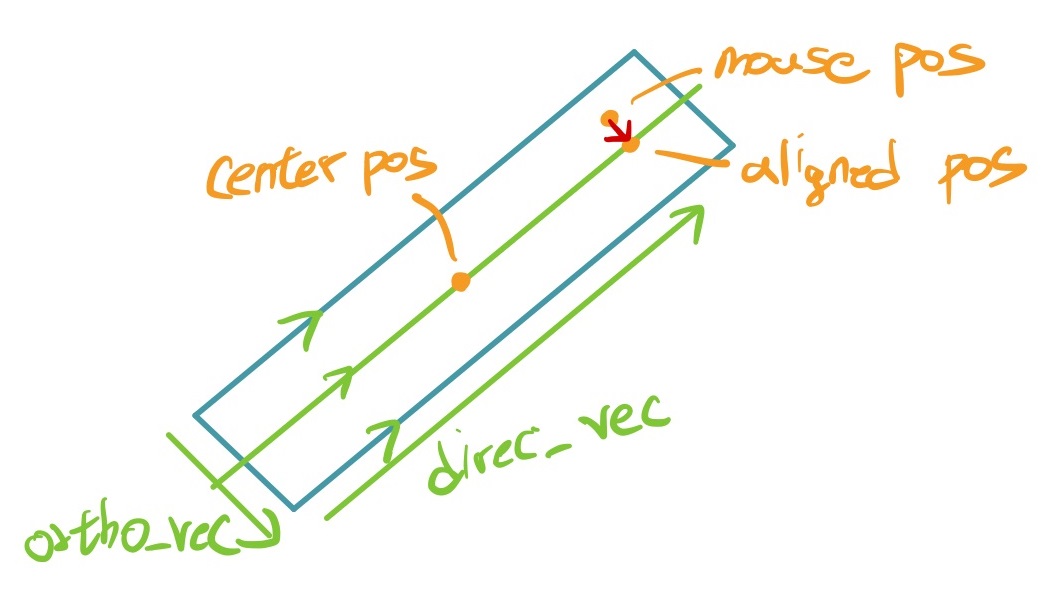
- FindPathCoord에서는 acter와 마우스 클릭 위치에 해당하는 face(면)과 관련된 정보를 찾아냅니다.

- 이때 각각의 object에 재 정의 된 IsInObj를 이용하여 검사합니다.

- 찾았다면 마우스 클릭한 좌표를 AlignPos 함수를 이용해서 정렬합니다.

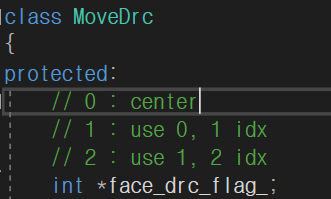
- 이 연산은 마우스 클릭 좌표계 즉 WdC에서 이루어집니다.

* AlignPos



- AlignPos는 위 그림과 같이 center와 direc\_vec를 지나는 직선 위로 마우스 좌표를 이동시키는 것이 목적입니다.

- direc\_vec를 구하기 위해 direction변수가 존재하는 데 이는 face의 여러 점들 중에서 어떤 두 점을 이용해서 direc\_vec을 구할지 알려줍니다.



- 위와 같이 0이면 aligned\_pos로 면의 중심점을 사용하고 1이면 direc\_vec로 0번과 1번 vertex의 차이를, 2이면 1번과 2번 vertex의 차이를 이용합니다.

- otrho\_vec는 direc\_vec에서 0,1번을 사용했으면 1,2번을 사용하는 식으로 구합니다.

- center와 direc\_vec를 이용한 직선과 mouse\_pos와 ortho\_vec를 지나는 직선의 교점이 aligned\_pos입니다.

- 이때 z값을 고려하면 연산이 복잡해지고 x, y는 0~1200, z는 0 ~ 1의 값을 가지므로 제곱 등의 연산을 거쳤을 때 오차가 심해지므로 x와 y값만 사용해서 직선을 구하고 교점을 구합니다.

- direc\_vec와 ortho\_vec의 좌표 값이 0인 경우 다양한 예외가 발생하는데 코드로는 작성했으나 따로 설명하지 않겠습니다.

- 면의 법선 백터를 구한 후, 방금 구한 aligned\_pos의 x,y 값을 이용해서 z값을 구합니다.

* FindPath

- 다음으로 acter가 이동할 경로를 구합니다.

- 미리 작성된 edge 인접배열을 bfs로 탐색하여 경로를 구하고 경로를 return합니다.

- 시작점과 끝점이 같은 경우(같은 object의 face내에서 움직이는 경우) 시작점을 vector에 담아 return합니다.

- 경로가 존재하지 않으면 FindPathCoord함수를 종료합니다.

* PathIdxToCoord

- 다음에는 object idx로 된 경로를 구체적인 WC의 좌표로 바꾸어 줍니다.

- FindCoord 를 반복 이용해서 curr\_idx와 next\_idx사이의 이동 좌표를 구합니다.

- 마지막으로 aligned\_pos까지의 좌표경로를 구합니다.

* FindCoord

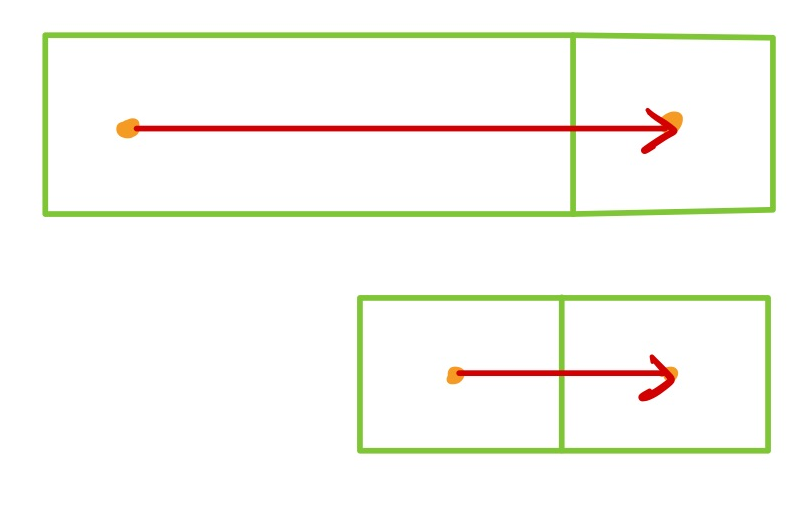
- curr 정보는 현재 actor가 있다고 가정한 면의 정보이고 next는 이동할 면의 정보입니다.

- curr\_face는 현재 actor가 있는 면이며, next\_face는 이동할 면의 vertex입니다.

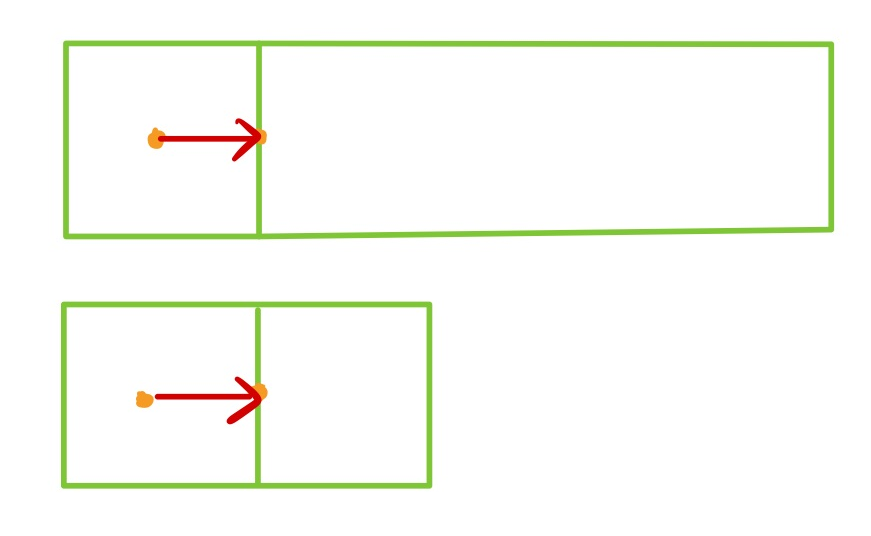
- 현재 위치한 면의 법선 백터 curr\_normal와 같은 방향백터를 갖는 next의 면을 찾습니다. 찾지 못하면 여기서 멈춥니다.

- 찾았다면 현재 면과 다음 면 사이의 direction에 따라 경우를 나누어 좌표를 구합니다.

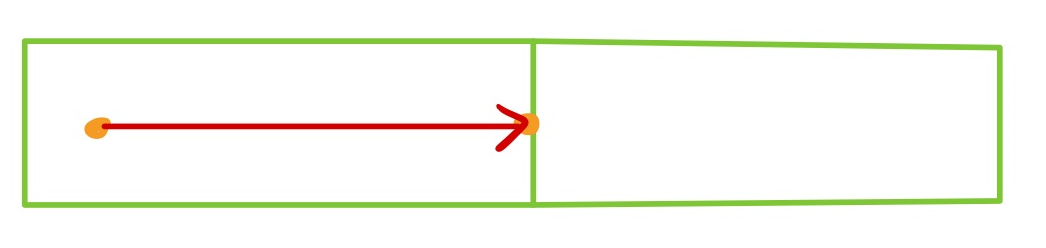
- direc가 0이면 정사각형, 1, 2이면 직사각형을 의미합니다.



- next\_direc가 0이면 curr\_direc에 무관하게 next\_face의 중점을 저장합니다.

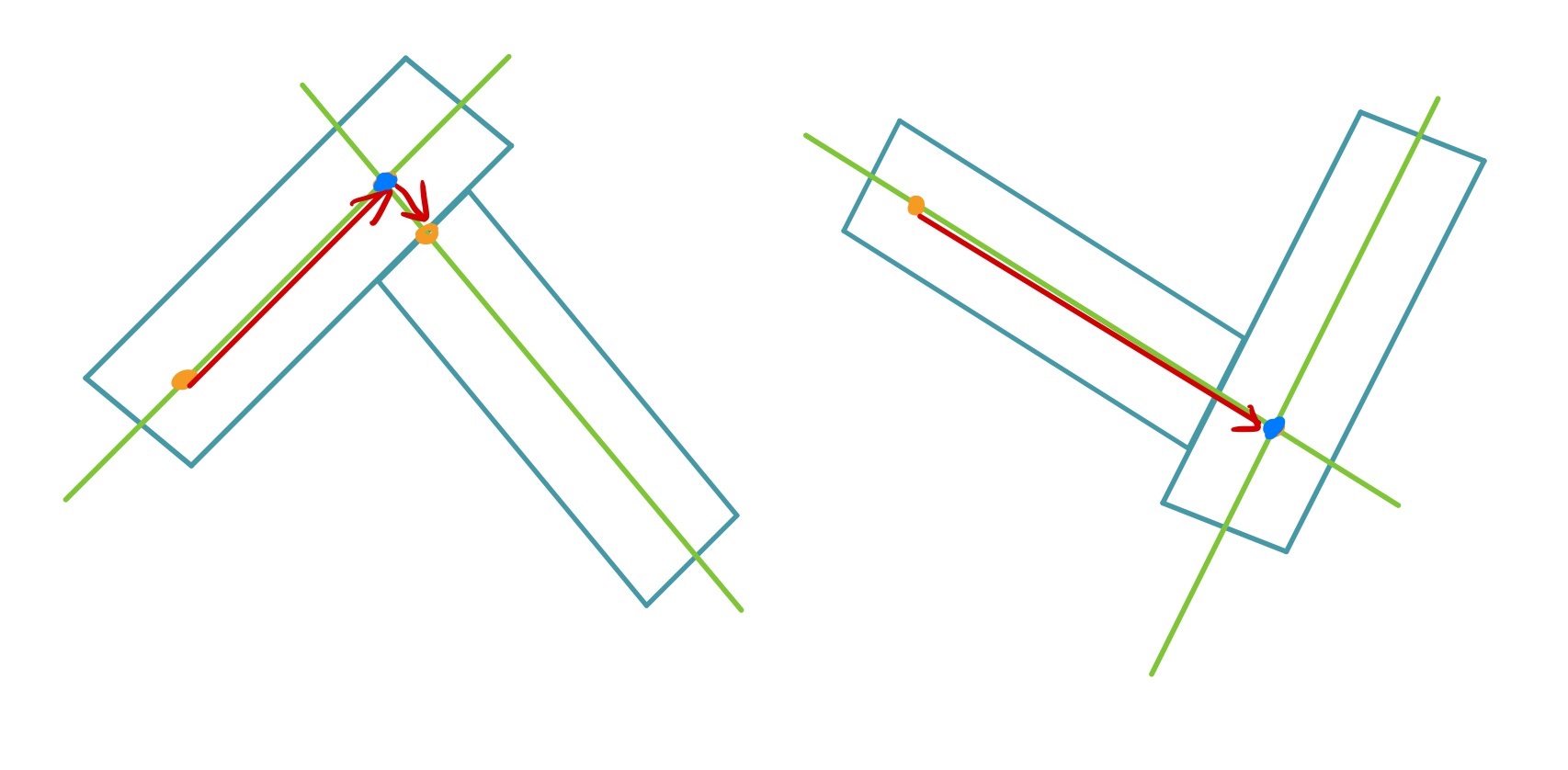


- curr\_direc가 0이면 next\_direc에 무관하게 curr\_face와 next\_face사이의 겹치는 변을 FindOverlappingLine함수로 찾고 중점을 저장합니다.



- 둘다 0이 아니라면 curr\_direc\_vec, next\_direc\_vec를 구합니다.

- 두 백터(x, y성분만)이 평행하면 curr\_direc가 0일 때와 동일하게 겹치는 변의 중점을 저장합니다.



- 평행하지 않다면 curr\_direc\_vec 방향으로 curr\_center\_pos를 지나는 직선과next\_direc\_vec 방향으로 next\_center\_pos를 지나는 직선 사이의 교점을 구합니다. 두 그림에서 파랑색 점이 교점에 해당합니다.

- direc\_vec의 성분이 0이어서 생기는 예외는 코드에는 포함되어 있으나 설명은 생략합니다.

- 구한 교점으로 curr\_face와 next\_face중 어디에 교점이 속하는 지 찾습니다.

- curr\_face에 포함되는 것이 첫 그림 next\_face에 속하는 것이 두번째 그림입니다.

- curr\_face에 포함 된다면 이동시 그림과 같이 교점을 지난 후 두 면 사이에 겹치는 선분의 중점 또한 지나야 합니다.

- next\_face에 포함 된다면 교점으로 만 이동하면 됩니다.

- 교점의 z좌표는 평면의 방정식을 이용해서 구한 후 필요한 좌표들을 저장합니다.

- 마지막으로 curr 정보들을 next정보들로 바꿔 다음 loop에서 사용가능하도록 합니다.

* FindOverlappingLine

- 두 면에서 겹치는 선분을 찾고 그중 작은 선분의 양 끝점을 return하는 함수 입니다.

- 두 면의 모든 선분에 대해 비교하며 긴 선분과 짧은 선분을 찾고 선분의 방향 백터가 평행한 지 찾습니다.

- 평행 하다면 긴 선분의 내부에 짧은 선분이 포함되는 지 검사합니다.

- 포함된다면 짧은 선분의 양 끝점을 return합니다.

- 이때 선분의 방향 백터의 성분이 0인 경우 예외를 조심해야합니다.

1. Actor의 이동 제한

- 위 사항까지 구현하게 되면 도형이 겹쳐 있어 이동 불가능해야하지만 경로가 구해지는 경우가 발생합니다.

- 따라서 도형이 인접하는 경우에 겹치는 면을 ChkDisableFace 함수를 이용해서 비활성화 시켜줍니다.

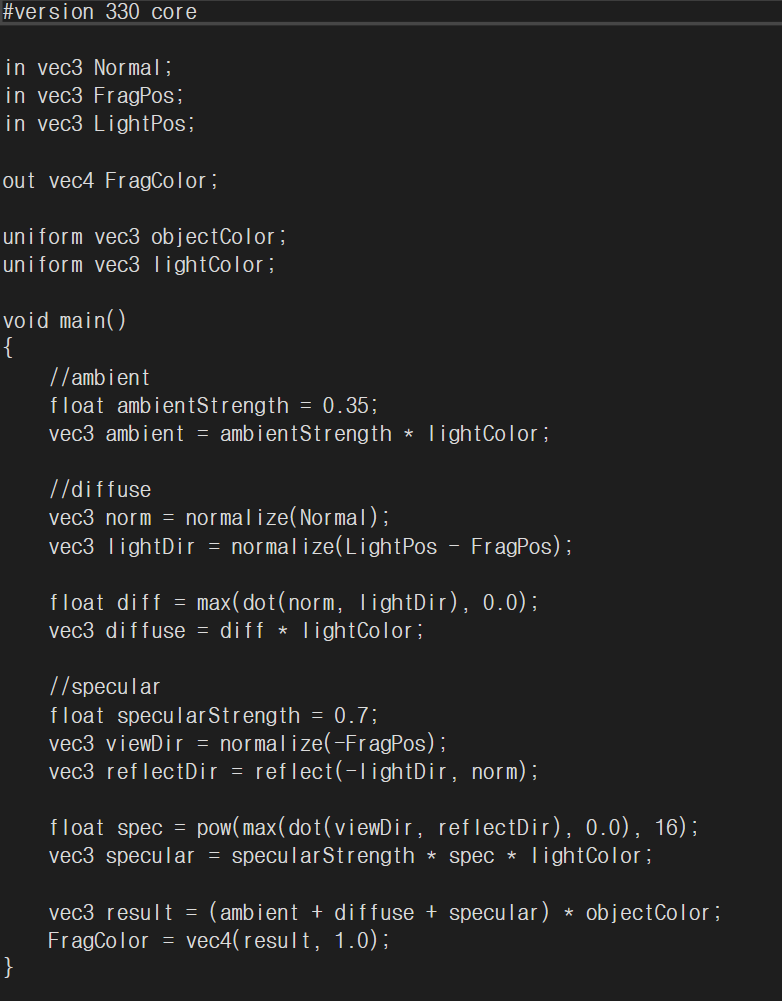
- ChkDisbleFace함수는 edge[i][j]가 1인 두 i, j를 찾아 CmpTwoObj함수를 호출합니다.

- 호출된 CmpTwoObj함수는 두 object의 면들 중 법선 백터가 반대 방향인 두 면을 찾아 겹치는지 비교 후 같으면 각각의 object에서 해당 면을 비활성화 합니다.

- 두 변이 겹치는지 비교할 때는 vertex를 각각 비교하는 것이 아니라 중심점을 구해서 두 중심점사이의 거리가 무시할 만큼 작은지 구하는 방법을 이용했습니다.

1. 빛 효과

- 기본적인 phone’s lighting model을 사용했습니다.



- fragment shader

1. 추가 구현 예정 사항

* 캐릭터가 움직이는 방향으로 회전
* 착시를 위해 물체를 움직이면 light 효과가 부자연스러운 현상 고침
* 특정 부분에서 느려지는 문제

- 레버를 돌리면 물체가 (-1.8, -1.8, -1.8)만큼 움직이는 문제때문에 발생

* level 파일 만들기

- 파일에서 level에 대한 정보를 읽어서 level을 생성하고 level class를 재사용가능하게 만들기

* 추가 level 만들기
* 시작화면 만들기

- 화면에 클릭 가능한 버튼 만들기

* 추가 object 만들기
* level제작 모드 만들기