Doom-style 2.5D FPS

201704044 백문하

목차

프로젝트 소개 - 1

최초의 FPS 게임으로 일컬어지는 "Wolfenstein 3D"와 그 개발사의 히트작 "Doom"은 맵 구성, 오브젝트 표현에 있어서 Full 3D가 아닌 2D를 사용했다.

프로젝트 소개 - 2

맵은 2차원적 구조를 가지고 있다.
한 블럭은 천장, 바닥, 벽 등으로 단순하게 구성되어있다.
한 블럭에서는 위아래가 있을 수 있지만
복층 구조는 구현하지 않는다.

오브젝트의 경우 항상 카메라를 향해 회전하여,
거리만으로 입체감을 구현한다.

3D FPS의 기본적인 조작 방식을 구현했다.
마우스를 이용한 시야 회전, 시야 방향에 따른 이동 등.

마지막으로 객체의 충돌을 검출하고 처리하는 것을 구현했다.

사용 기술 설명 - 구조

main.cpp : GLUT 루프 및 콜백 함수, 초기화 등.
Manager : 각종 Actor와 Map을 업데이트하고 그리는 클래스.
Map : 맵을 불러오고 렌더링하는 역할.

게임은 나의 입력이 없어도, 다시 그릴 상황이 아니어도
매 루프마다 업데이트하고 다시 그린다.
때문에 main.cpp에 매 루프마다 실행되는 idle 콜백 함수를 만들었다.
이 함수는 delta time을 측정하고, Manager를 통해 모든 객체를 업데이트한 뒤,
glutPostRedisplay();로 즉시 다시 그리게 했다.

사용 기술 설명 - 구조

게임 객체들은 모두 Actor를 상속하는 모놀리식 클래스 구조로 개발했다.

Actor : 게임 객체. 위치와 각도, 크기 등의 정보를 가진 기본 클래스.

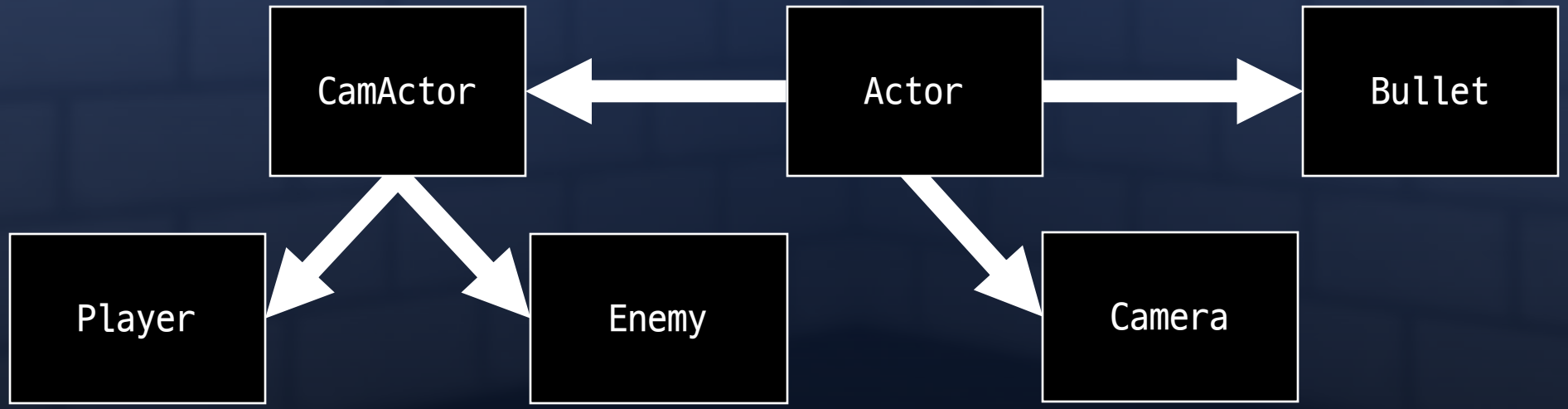
CamActor : 투사체를 발사할 능력이 있는 Actor 클래스. Camera를 가진다.

Player : 사용자의 입력으로 움직이고 맵에 대해 충돌체크를 하는 CamActor 클래스.

Enemy : 고정된 자리에서 지속적으로 총알을 발사하는 CamActor 클래스.

Camera : Player와 Enemy의 시선을 표현하기 위한 Actor 클래스.

Bullet : 플레이어와 적이 발사하는 총알 Actor 클래스.



사용 기술 설명 - 맵

맵은 미리 만든 숫자 텍스트 파일을 불러와 fscanf를 통해 파싱한다.
그 뒤 2차원 배열에 숫자를 그대로 넣는다.
각 숫자는 각 블럭의 번호를 의미한다.

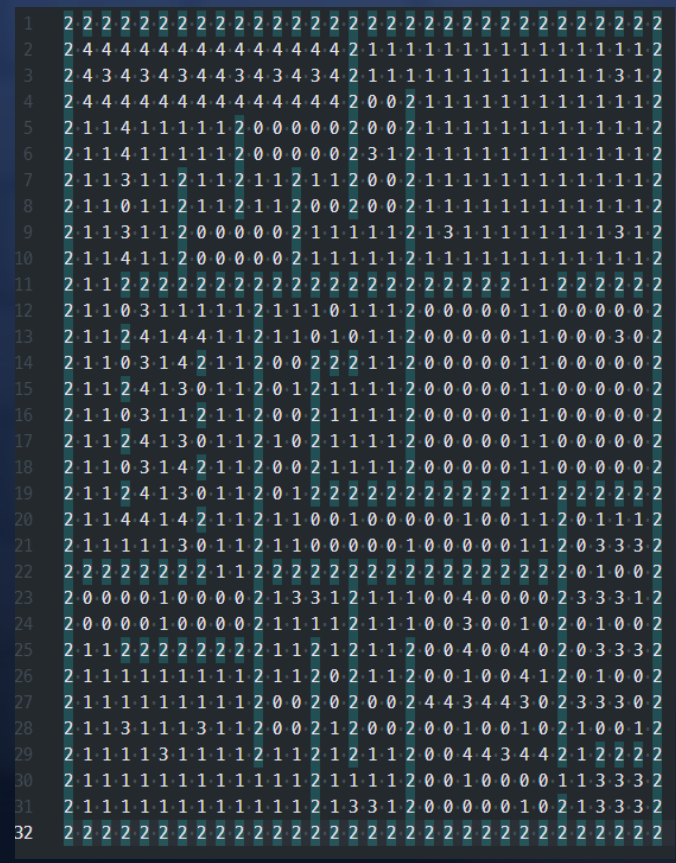
0 - bt_underground :
빠지면 죽는 함정 지형으로, 점프로 넘어가기 가능.

1 - bt_ground : 걸어다닐 수 있는 지형.

2 - bt_wall : 아무것도 통과할 수 없는 벽.

3 - bt_enemy :
bt_half와 같지만 로드 시 적이 위에 생성됨.

4 - bt_half : 플레이어가 건너갈 수 없지만
위로 총알이 통과할 수 있는 담장.

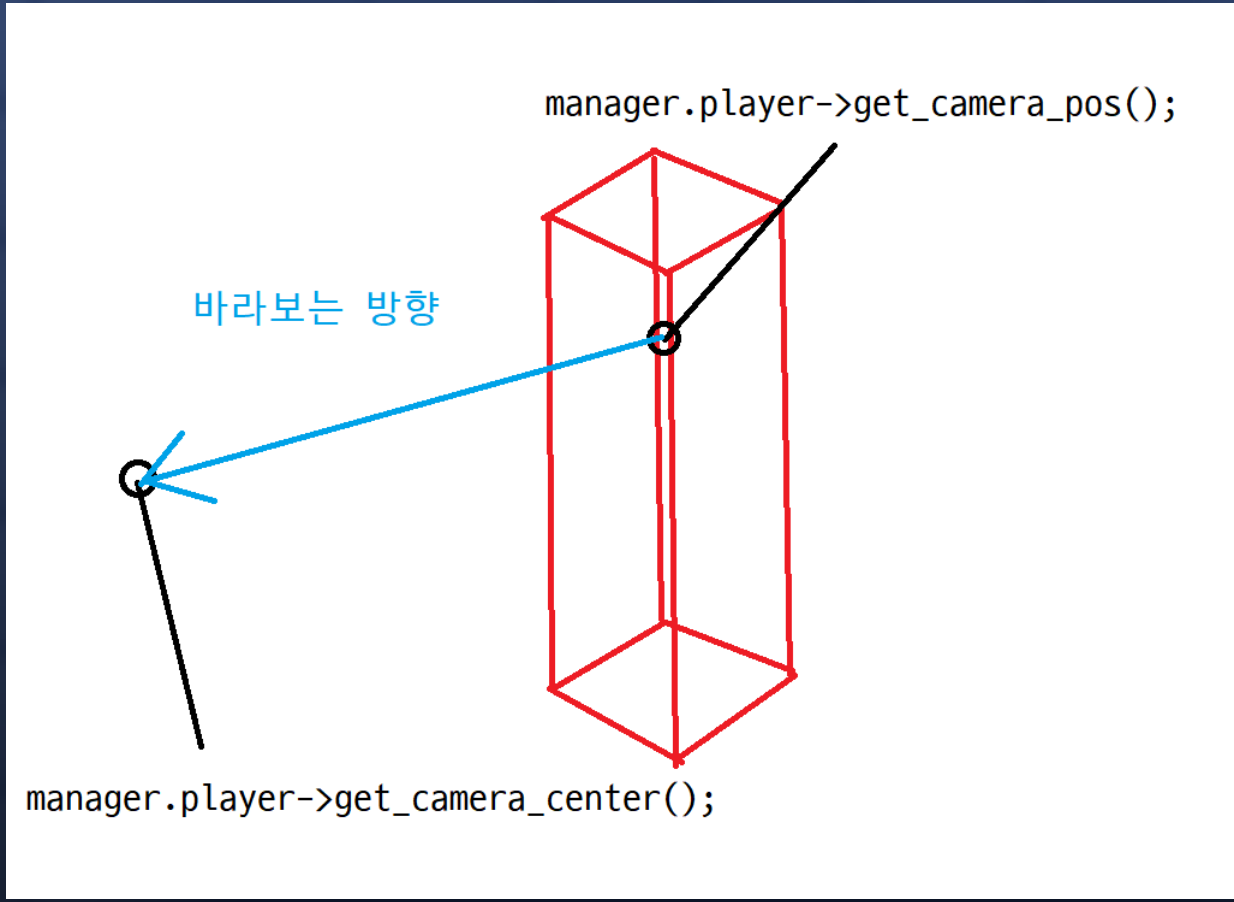


사용 기술 설명 - 맵

2차원 배열을 반복문으로 돌면서 해당하는 블럭을 그린다.
공통되는 부분은 따로 함수로 만들어 그리게 된다.
bt_wall을 제외하면 모두 천장을 그리게 되고,
bt_underground를 제외하면 모두 지하 벽면을 그린다.

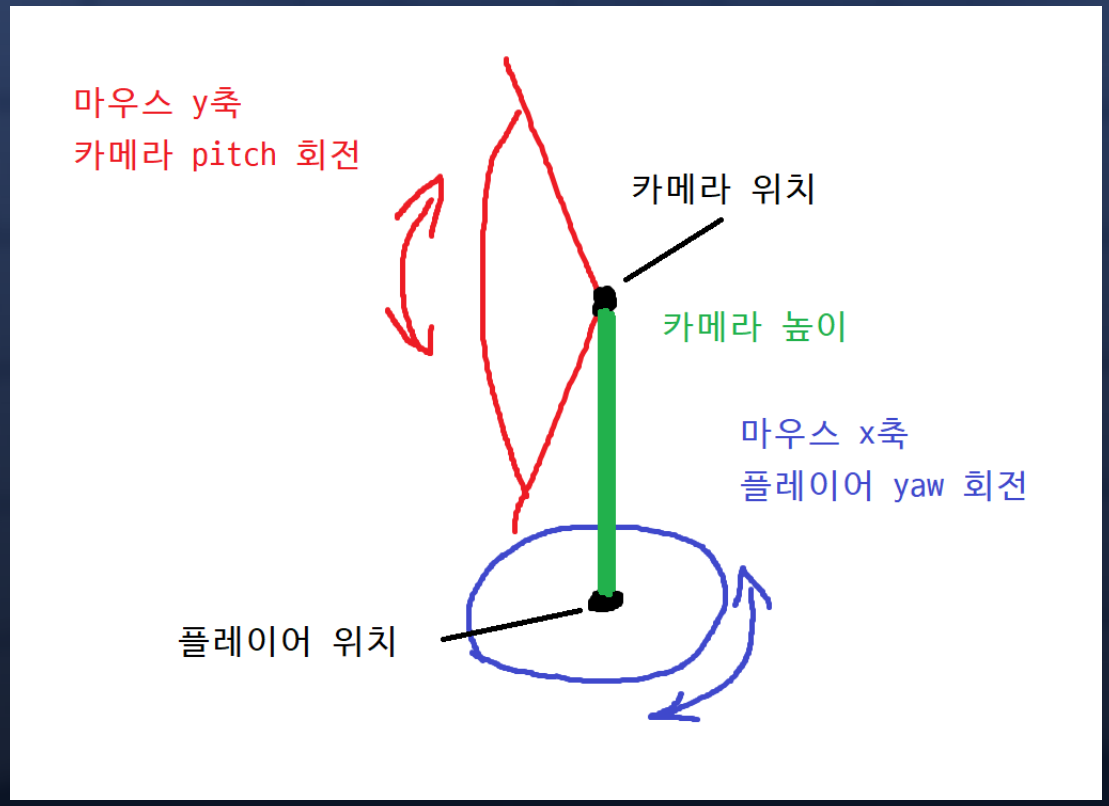
사용 기술 설명 - 1인칭 카메라

플레이어 카메라의 위치와 바라보는 방향 값으로 gluLookAt를 호출했다.
카메라 위치로부터 카메라 각도로 일정 거리 떨어진 곳에 LookAt 한다.



사용 기술 설명 - 1인칭 카메라

passive motion 함수에서, 마우스가 이동하면 이동 값을 기록하고
다시 화면 중앙으로 마우스 커서를 이동시킨다.
매 루프마다 호출되는 idle 함수에서, 기록한 마우스 이동값을 플레이어에 전달,
플레이어와 카메라의 각도를 회전시킨다.



사용 기술 설명 - 충돌 처리와 해결

게임 내에서 이동하는 오브젝트는 플레이어와 총알 두 가지다.

맵이 블럭으로 되어있기 때문에, 해당 오브젝트의 평면 위치와 근접한
블럭 사각형들과 충돌 체크를 하면 된다.

여기서 총알은 크기가 없는 점이기 때문에, 충돌 처리가 단순하다.
벽이나 플레이어, 적 모두 직육면체로 구현했기 때문이다.
또한 총알은 닿으면 즉시 파괴되기에 추가 처리가 필요없다.
단, bt_half에 대해서는 빈 공간을 통과하기에
높이를 기준으로 추가 처리가 필요했다.

그러나 플레이어는 평면 크기를 가지고 있고,
벽에 부딪히면 나아가지 못하기 때문에
더 많은 충돌 처리가 필요하다.

사용 기술 설명 - 충돌 처리와 해결

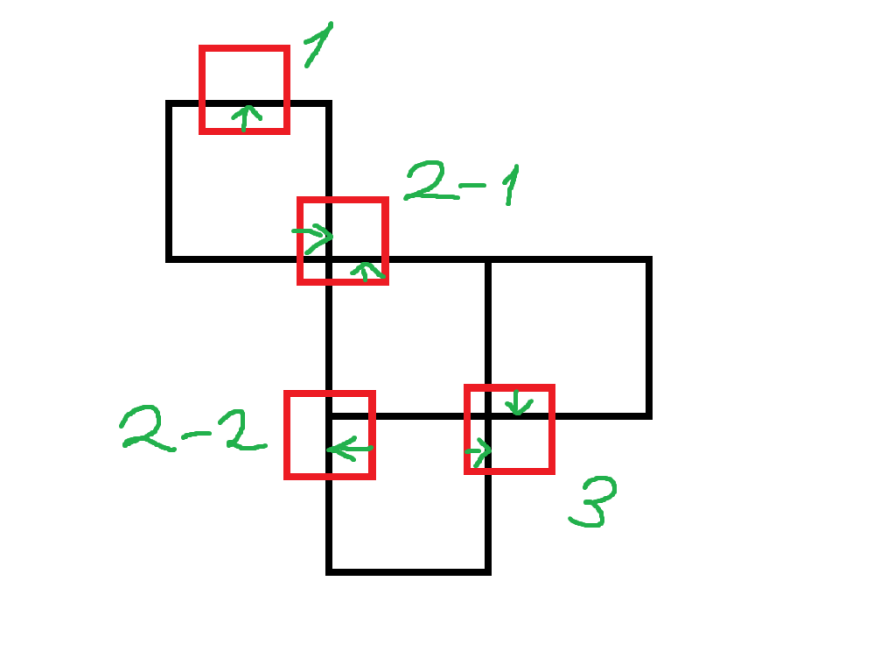
플레이어의 경우 크기가 블럭보다 작기 때문에 4가지의 충돌이 발생한다.

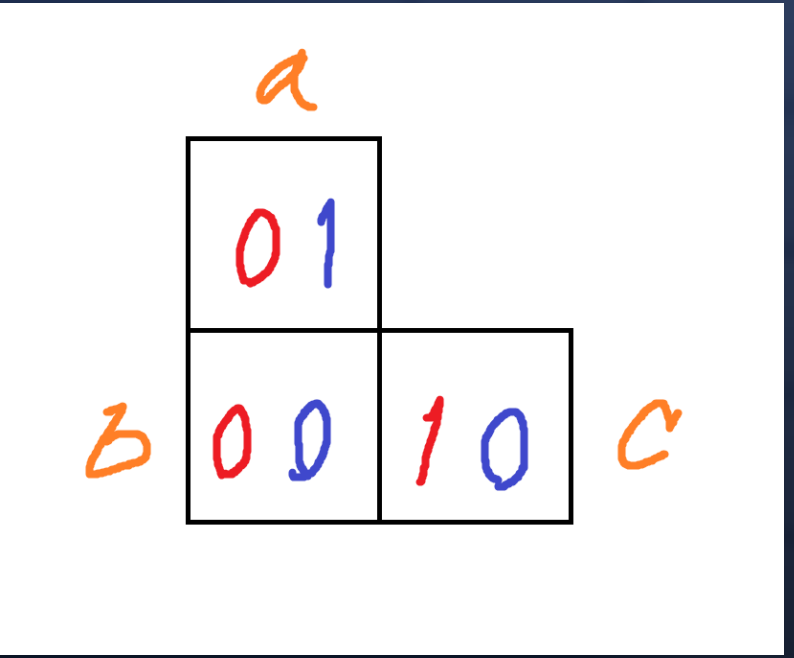
1 : 면적이 넓게 들어간 축 하나만 위치 보정.

2-1 : 여유 공간이 더 많은 쪽으로 양쪽 축에 대해서 보정.

2-2 : 가장 넓은 블럭만, 한쪽 축에 대해서만 보정.

3 : 안쪽 블럭(b)을 찾아서 보정. 결과적으로 바깥 두 블럭의 각 축에 대해서.



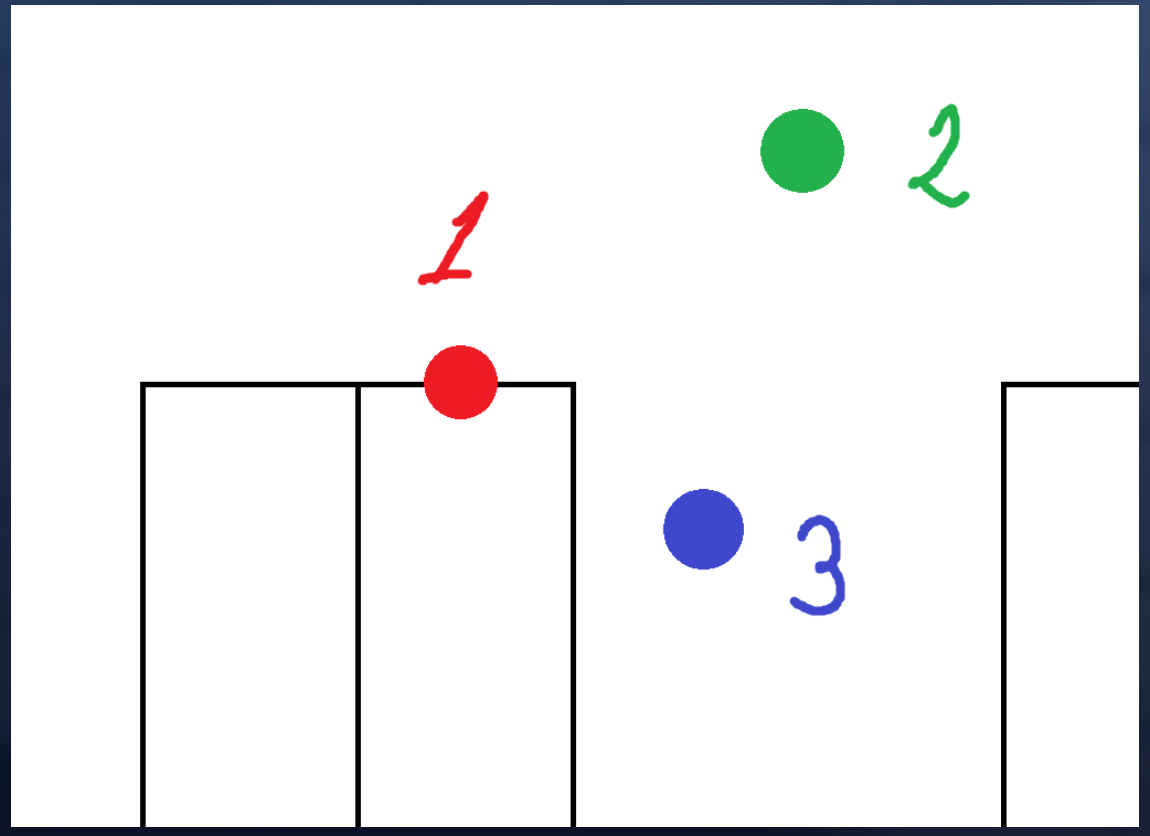


사용 기술 설명 - 충돌 처리와 해결

플레이어의 위치 기준점은 발 중앙이고, 위치는 중력의 영향을 받아 낙하한다.
bt_undergroound에 떨어져 바닥 높이보다 낮아진다면 다시 땅으로 올라올 수 없다.
bt_ground가 벽처럼 취급되는 것이다.

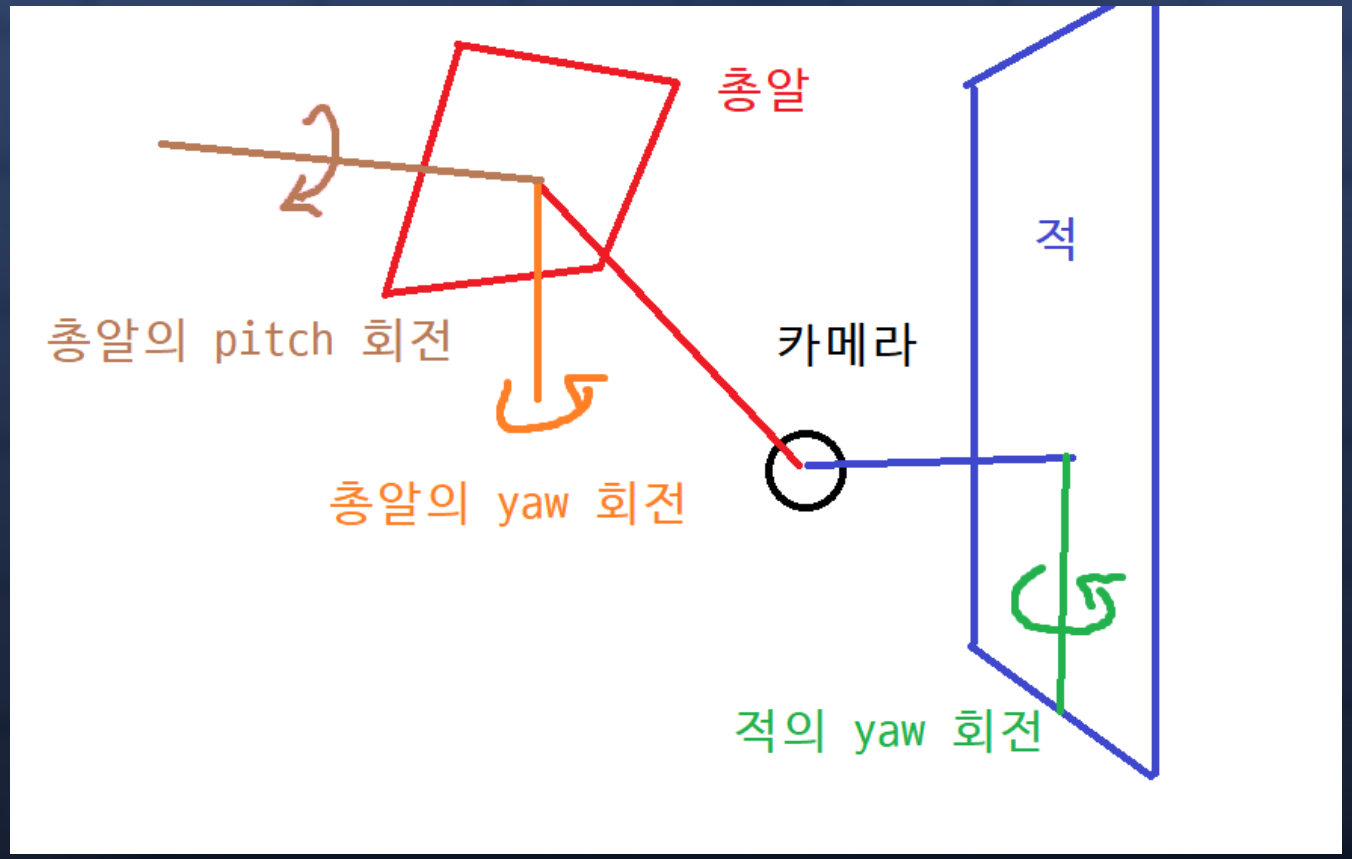
1은 bt_ground 위에 있을 때.
2는 bt_underground 위에 있지만,
공중에 떠 있는 상태.
3은 bt_underground 위에 있지만,
바닥 높이보다 낮은 상태.

3의 상태임이 확인되면
bt_ground도 벽으로 취급한다.



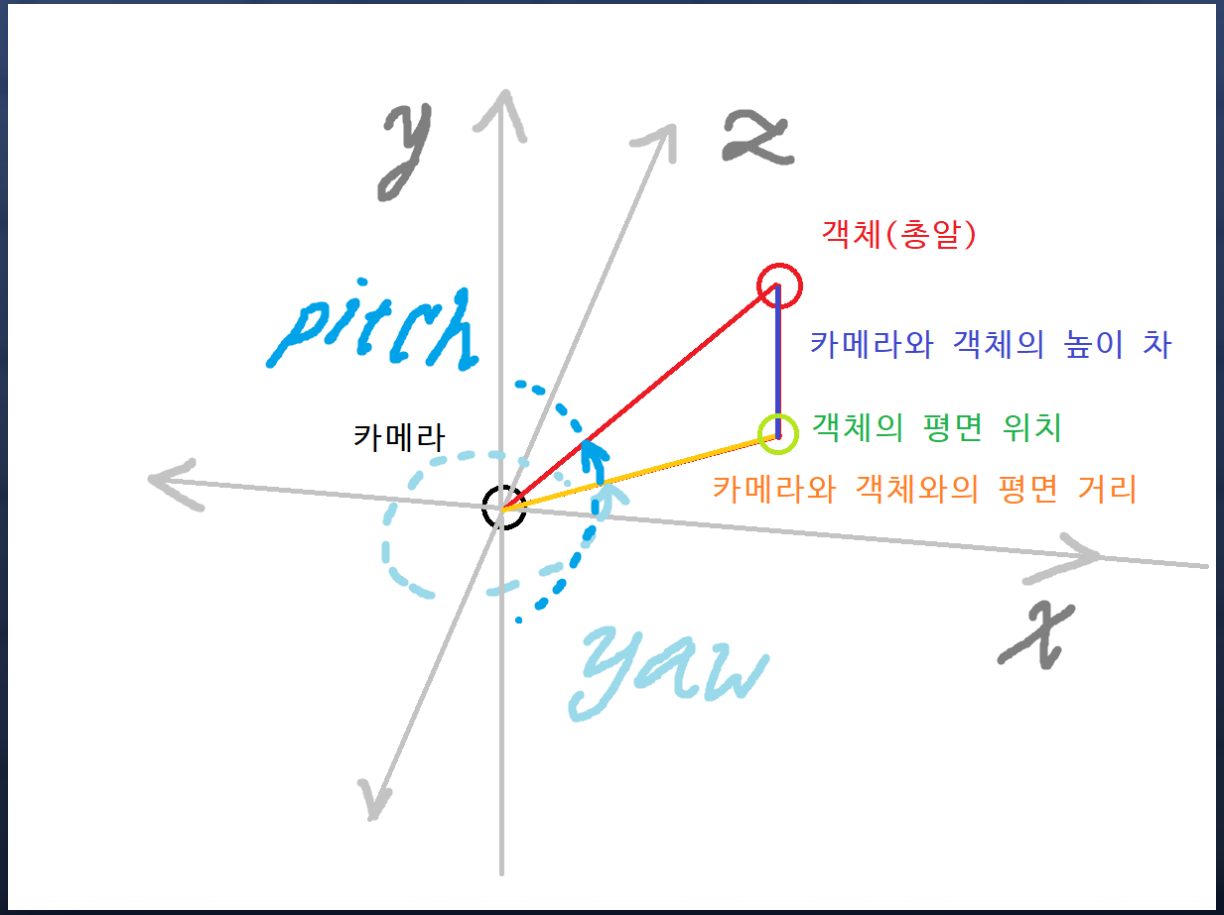
사용 기술 설명 - 카메라를 바라보는 객체

총알과 적 등의 객체는 2차원 스프라이트를 3차원에서 회전시켜
카메라를 바라보게 구현한다.



사용 기술 설명 - 카메라를 바라보는 객체

카메라와 객체의 상대적인 위치를 알면 atan을 통해 각도를 구할 수 있다.
이 각도로 객체를 회전시키면 카메라만을 바라본다.



사용 기술 설명 - HUD

플레이어의 크로스헤어와 체력바는 화면의 맨 앞에 평면으로 표시되어야 한다.
따라서, glDisable(GL_DEPTH_TEST); glDepthMask(GL_FALSE);을 통해
화가 알고리즘을 사용한다. 또한, gluOrtho2D로 재설정한다.
HUD를 그리고 난 뒤에는 미리 push해둔 매트릭스를 pop으로 되찾아간다.
gluOrtho2D 에서 화면의 크기를 기준으로 삼았기 때문에,
상대적 위치 좌표도 사용할 수 있게끔 별도의 함수도 제작했다.

사용 기술 설명 - 반투명 객체

등장하는 적들과 총알은 알파 채널을 가진 반투명 텍스처를 사용한다.
우선, glEnable(GL_BLEND); glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
을 통해서 알파 블렌딩을 하면 텍스처를 RGBA로 불러왔을 때 정상적으로 그린다.

그런데, 투명한 객체는 아무렇게나 그리면 뒤에 있는 객체가 그려지지 않는다.
따라서, 매번 그릴 때마다 액터들이 들어있는 리스트를 정렬한다.

std::list의 sort는 원소 타입을 위한 비교 함수가 필요하다.
때문에 두 객체와 플레이어간 거리를 비교하는
함수를 만들어 사용했다.

사용 기술 설명 - 기타

레트로한 분위기를 위해서 최근접 이웃 필터링(GL_NEAREST)을 사용했다.

총알과 적은 시인성을 위해서 glDisable(GL_LIGHTING); 했다.

glColor3f를 이용해서, 적은 체력이 부족할수록 빨간색으로 물들어간다.

페이스 컬링을 활성화했다 glEnable(GL_CULL_FACE); glFrontFace(GL_CCW);

FPS이기 때문에 커서를 지웠다. glutSetCursor(GLUT_CURSOR_NONE);

delta time을 계산하기 위해서 표준 라이브러리 std::chrono를 사용했다.
std::chrono::high_resolution_clock::now(); 로 현재 시간을 구했다.
게임 내 모든 액터의 업데이트는 이 delta time을 따른다.

보완할 사항

게임 재시작, 도착지점 트리거, 메뉴 등을 계획했으나 일정 문제로 실패했다.

적들도 둠 처럼 인공지능을 부여하고 각도마다 다른 스프라이트를 사용하거나,
애니메이션을 구현할 예정이었지만 하지 못했다.

클래스 구조가 엉성하고, 모든 요소가 public으로써 구조상 문제가 있다.
특히 충돌 체크 코드는 재사용되지 않게 짜여졌다.

감사합니다