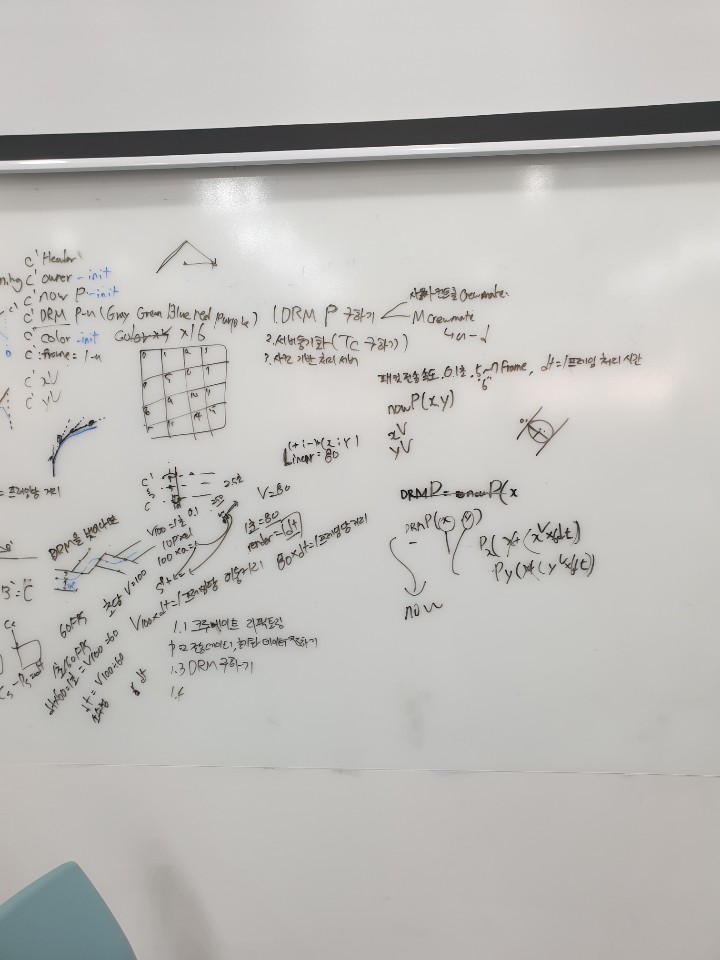
Dead Reckoning(추측 항법)



캐릭터 이동에 부자연스러운 문제점이 생김

* 실제로 비행기 경로 예상에 사용한다는 추측 항법 알고리즘을 간단하게 만들어서 구현해볼 예정

Lib Gdx에는 2D 물리 엔진이 있음. 그래서 스프라이트 객체에 속도를 줄 수 있음.

그리고 캐릭터가 이동 할 때 방향키에 따라 방향도 존재함

그래서 지금 위치로부터 몇 초 뒤에 내 캐릭터가 위치할 지점을 예측하는 것이 가능함

ex) 캐릭터(2D 엔진이 들어간 스프라이트) 속도가 초당 100만큼 움직인다고 할 때 0.1초당 움직이는 거리는 10이고 1dt(델타 타임)당 움직이는 거리는 100\*dt이다.

게임의 프레임은 60프레임이니까 1초당 60프레임이 재생된다. 그러면 1초에 100\*dt\*60만큼 움직인다.

델타 타임은 이전 프레임 시작시점과 현재 프레임 시작 시점의 차이를 나노 타임으로 찍어 놓고 차이를 이용해서 구할 수 있다.

패킷의 전송 속도는 초당 10번이라고 할 때 0.1초당 한번씩 패킷이 전송된다고 생각할 수 있다.

그럼 0.1초당 패킷이 재생되기 때문에 우리는 0.1초뒤에 새로운 패킷을 받기 전까지 내 캐릭터가 이 지점까지는 움직일 것이다 라는 것을 예상해서 다른 클라이언트 측에다가 보내줄 수 있다. 1초는 60프레임 = 0.1초는 6프레임

6프레임 이동 예상 거리 = 100 \* dt \* 6

그럼 현재 위치 + 6프레임 이동 예상 거리를 구해서 DRM을 구할 수 있다.

여기서 2가지 방식을 생각할 수 있는데

첫번째는 현재 실제 위치(True Path)와 DRM을 보내고 실제 위치로 바로 오차를 수정후에 다시 DRM으로 이동하게 하는 방식(살짝 부자연 스러울 수 있다고 예상 됨)

두번째는 실제 위치는 고려하지 않고 DRM으로만 이동하게 만드는 것(약간의 오차를 감수하더라도 살짝 부자연 스러움을 지울 수 있음)

세번째도 있는데 어려움

세번째는 현재 클라이언트가 DRM을 그리고 범위를 지정한 다음 실제 경로가 DRM을 벗어날 경우 실제 위치와 DRM을 갱신해서 보내는 방법(실제 위치가 DRM을 벗어 낫는지 확인 할려면 직선이 교차하는지 방정식을 만들어야 함)

그리고 이를 적용시키기 위해서 크루메이트 인터페이스를 기본적으로 하나 정의해 놓고

실제로 유저가 컨트롤하는 2D 엔진이 들어간 크루메이트 클래스를 하나 만들고

멀티에 접속해 있는 다른 유저의 캐릭터를 보여주기 위한 크루메이트 클래스를 만들어야 함

둘 다 정보를 계속 업데이트 시켜야 하고 프레임을 가져와야 한다는 점에서 메소드는 같겠지만 실제로 움직이는 방식은 서로 다르기 때문에 크루메이트 기본 인터페이스에 정의 후 각각 구현하는 방식으로 가야함