1. 애니메이션 동기화 버그 해결

저번 작업에서 서버에서 들어오는 애니메이션 종류 패킷 중 Jump, Dash, Attack이 동기화 되지 않고 애니메이션이 실행되지 않는 문제가 있다.

구현했던 내용을 보면,

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 애니메이션 타입 패킷을 서버에서 클라이언트로 보내주면 클라이언트는 이를 토글 형식으로 이용하여 Jump, Dash, Attack을 진행하게 한다.

Jump, Dash, Attack 모두 비슷한 방식이라 Jump로 오류를 찾아보기로 했다.

패킷도 정상적으로 들어오는 것을 확인했고, 서버 없이 클라이언트를 실행했을 때 Jump() 자체에 문제가 없다는 것을 확인했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

결정적인 부분을 찾아낸 것으론 클라이언트 본인은 서버로부터 애니메이션을 잘 받아와서 진행하지만, 다른 플레이어만 애니메이션이 작동하지 않는다는 것이다.

그래서 다른 플레이어만 작동하지 않는 이유에 대해 찾아보려 했다.

텍스트이(가) 표시된 사진

자동 생성된 설명



며칠에 걸쳐 찾아봤지만 결국엔 사소한 문제였다.

서버에서 클라이언트로 애니메이션 타입을 보내줄 때 p\_Client를 SendOverlapped하여 한 클라이언트에 반복하여 들어가는 것이 문제였고, 이를 바꿔서 다른 클라이언트에게 전송되게 바꿔주었다.

1. 몬스터 타겟팅 동기화

이전 작업까지 몬스터의 타겟팅은 항상 먼저 들어오는 플레이어로 고정했었다.

하지만 이제는 몬스터가 주변에 있는 플레이어들을 탐색하고, 그 중 제일 가까울 플레이어를 타겟팅으로 설정해야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선, 단일 플레이어를 탐색하는 함수를 여러 플레이어를 탐색할 수 있도록 바꿔줘야 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

현재는 모든 클라이언트를 탐색하며 거리를 재고, 그 거리가 가장 짧은 플레이어의 주소를 해당 몬스터의 TargetPlayer으로 둔다.

이렇게 한다면 몬스터의 Track(추적) 대상은 제일 가까운 플레이어가 된다.

테스트 결과, Track(추적)의 타겟은 제대로 적용되어 가까운 플레이어를 따라가지만, 다른 플레이어를 보고 공격하는 버그가 있었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이전에 테스트를 위해 처음 접속하는 플레이어의 id를 SC\_AGGRO\_PLAYER\_PACKET에 담아 보내던 것이 문제였다. 이를 몬스터 타겟 플레이어의 id로 바꿔서 보내주어 해결했다.

1. 맵의 OBB을 파일로 저장

클라이언트에서 사용하는 맵을 서버에서 충돌처리 할 수 있도록 맵의 OBB만 파일로 저장하여 서버에서 불러오려고 했다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Ofstream을 이용하여 맵 오브젝트의 OBB를 Center, Extents, Orientation 순서로 저장하는 함수다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명 파일에 제대로 저장된 모습이다.



서버에서도 OBB끼리 충돌처리를 하기 위해 OBB 컨테이너를 따로 만들었다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

서버에서 ifstream으로 OBB를 불러오고, 그 포인터를 BoundingGameObjeccts에 추가한다.

이 작업으로 서버는 맵의 OBB를 모두 불러오게 된다.

단, 주의해야 할 점이 미리 사전작업으로 클라이언트에서 맵 OBB데이터를 저장해야 한다는 것이다.

몬스터, 플레이어, 원거리 공격 등 동적인 OBB객체는 서버에서 생성하고 작업해도 클라이언트로 제대로 정보만 보내준다면 아무 문제가 없다.

이제 받아온 OBB로 충돌체크를 해보려 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

플레이어를 생성할 때 바운딩 박스를 생성해주고,

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

플레이어가 맵 오브젝트와 충돌하는지 텍스트 출력으로 확인해본다.

다행히 충돌체크가 정상적으로 작동했고 이제 PlayerMovementComponent등 충돌 처리가 필요한 Component에 충돌 처리하는 방법을 생각해야 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

위의 충돌체크를 PlayerMovementComponent로 옮겨주고, 플레이어 다음 움직임이 물체와 충돌한다면, 플레이어의 Velocity를 변경시켜 미끄러지듯이 움직이도록 처리하려 한다.