C++: Line Trace / Raycast

- Raycast (레이캐스트) 또는 Trace (트레이스)를 통해 보이지 않는 광선을 "발사"하여 두 점 사이에 걸리는 지오메트리가 있는지 확인하고, 걸리는 지오메트리가 있으면 어떤 작업을 할 수 있도록 걸린 것을 반환
- 트레이스를 실행하여 오브젝트와 콜리전 검사를 한 뒤 히트한 오브젝트를 반환하거나,
- Trace Channel (트레이스 채널)로 트레이스를 실행하여 특정 (콜리전 세팅으로 설정 가능한) 트레이스 채널에 반응하도록 지정된 오브젝트의 경우에만 히트 정보를 반환하도록 할 수도 있다.
- 오브젝트 또는 트레이스 채널로 트레이스를 실행하는 것에 추가로, 트레이스를 실행시켜 Single (싱글) 히트 또는 Multi (멀티) 히트 감지가 가능
- 싱글 트레이스는 트레이스에서 단일 히트 결과만을, 멀티 트레이스는 다중 히트 결과를 반환
- 트레이스를 통해 사용되는 광선 유형을 직선, 박스, 캡슐, 구체 중에서 지정 가능

장애물 처리

충돌된 액터가 플레이어일 때만 공격 실시

```
Evoid ARangedMonster::HasObstacle()

{
FHitResult outHit;

FVector Start = this->GetActorLocation(); // 레이캐스트의 시작점
FVector End = Start + this->GetActorForwardVector() * 100000; // 레이캐스트의 종료점

FCollisionQueryParams CollisionParams;

CollisionParams.AddIgnoredActor( this ); // 현재 액터는 충돌 검사에서 무시

// 레이캐스트를 수행하고 충돌 정보를 outHit에 저장.
bool blsHit = GetWorld()->LineTraceSingleByChannel( outHit , Start , End , ECC_Visibility , CollisionParams );

if (blsHit)
{
    DrawDebugLine( GetWorld() , Start , End , FColor::Red , false , 0.1f , 0 , 5.f );
    UE_LOG( LogTemp , Warning , TEXT( "Obstacle detected!" ) );
}
else
{
    DrawDebugLine( GetWorld() , Start , End , FColor::Green , false , 0.1f , 0 , 5.f );
    UE_LOG( LogTemp , Warning , TEXT( "No Obstacle detected!" ) );
}
```

C++: Line Trace / Raycast

```
if (blsHit)
{

// 충돌한 액터를 처리

AActor* HitActor = outHit.GetActor();

if (HitActor)
{

// 충돌한 액터가 플레이어가 아니라면

AProjectDCharacter* player = Cast<AProjectDCharacter>( HitActor );

if (player)
{

RangedAttack();

}

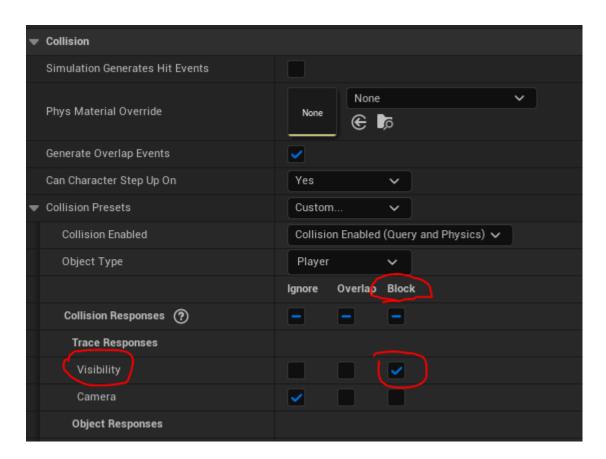
else
{

//PatrolState로 전환

MonsterFSM->state = EMonsterState::Move;

anim->animState = MonsterFSM->state;
}
}
```

Visibility→Block 처리!!



C++: Line Trace / Raycast 2