Day 5. 카타나 공격/회피

5-1. 카타나 공격 구현

```cpp
void AKatanaCharacter::Attack_Basic(float damage)
{
    FHitResult HitResult;
    float AttackRange = 100.f;
    float AttackRadius = 50.f;

    UE_LOG(LogTemp, Log, TEXT("%f"), damage);
    FVector Center = GetActorLocation();

    FVector Forward = Center + GetActorForwardVector() * AttackRange;
    FCollisionQueryParams Params(NAME_None ,false, this);

    bool Result = GetWorld()->SweepSingleByChannel(
        OUT HitResult,
        Center,
        Forward,
        FQuat::Identity,
        ECollisionChannel::ECC_GameTraceChannel3,
        FCollisionShape::MakeSphere(AttackRadius),
        Params);

    float HalfHeight = AttackRange * 0.5f + AttackRadius;
    FQuat Rotation = FRotationMatrix::MakeFromZ(GetActorForwardVector()).ToQuat();
    FColor DrawColor;

    if (Result && HitResult.GetActor())
    {
        DrawColor = FColor::Green;

        UGameplayStatics::SpawnEmitterAtLocation(GetWorld(), HitEffect, HitResult.ImpactPoint);

        AActor* HitActor = HitResult.GetActor();
        UGameplayStatics::ApplyDamage(HitActor, damage, GetController(), HitActor, NULL);
    }
    else {
        DrawColor = FColor::Red;
    }

    DrawDebugCapsule(GetWorld(), Center, HalfHeight, AttackRadius, Rotation, DrawColor, false, 3.f);
}
```

첫 번째 캐릭터인 카타나를 사용하는 캐릭터의 공격을 구현했습니다. 검을 사용하는 캐릭터의 공격을 구현할 때 여러 가지의 방식이 존재했는데 그 중에서 SweepSingleByChannel를 사용하는 방식을 채택했다. SweepSingleByChannel을 사용하면 내가 어떤 형태의 충돌체를 사용할 것인지 지정해야 했고 충돌체의 종류에는 Box, Sphere, Capsule이 존재했습니다다. Box는 단순하고 가벼운 충돌처리를 검사하는데 적합하지만 정확성이 다소 떨어졌고 Sphere는 충돌을 정확하게 검사할 수 있지만 연산 비용이 높은편 이였으며 Capsule은 Box와 Sphere의 장단점에 중간에 위치 했습니다. 충돌의 연산 비용에 대해서는 아직 자세히 알지는 못하지만 마비노기영웅전과 같은 게임처럼 정밀한 히트박스를 구현해보고 싶어서 Sphere를 택하여 SweepSingleByChannel를 구현했습니다. 반지름과 길이, 높이 등은 아직 SweepSingleByChannel의 매개변수들에 익숙하지 못해서 하나씩 값들을 바꿔보면서 실행하여 제일 적당하다고 생각되는 값을 넣었습니다. 충돌을 확인해 보기 위해 SweepSingleByChannel에 넣었던 값들과 똑같은 값으로 DrawDebugCapsule을 그려 확인했습니다.

5-2. 회피

캐릭터마다 회피기를 구현하기 위해 회피 기능을를 만들어 두었습니다. 이동을 구현할 때와 똑같이 입력 키를 Bind해서 묶고 애니메이션을 실행하는 방식으로 만들었는데 이동할 Distance 값이나 8방향에 대한 움직임, 동시 키 입력등 생각보다 감안해야할 요소들이 많았습니다. 그리고 캐릭터를 조작할 때 공격/스킬/점프/달리기등의 연계를 Should와 is ~ing 등의 변수로 다룰 계획이라 다른 기능들이 구현되어야 설계를 할 수 있을 것 같아서 기본적인 구현까지만 완성해 두었습니다.