**과제**

1. **실행시 발생하는 Blueprint Error Messeage 를 고쳐주세요. 각 에러 메시지에 대한 발생 이유와 고친 결과를 문서로 정리해주세요.**

> **가장 많이 표시되는 에러 목록은 다음과 같습니다.**

1. CallFunc\_GetPlayerControllerBP\_Controller 프로퍼티를 읽으려던 중 None 에 접근했습니다 >> GetPlayerController 노드에서 None 이 반환된 결과.
2. CallFunc\_GetGameModeBP\_MyGame 프로퍼티를 읽으려던 중 None 에 접근했습니다 >> GetGameMode 노드에서 None 이 반환된 결과.

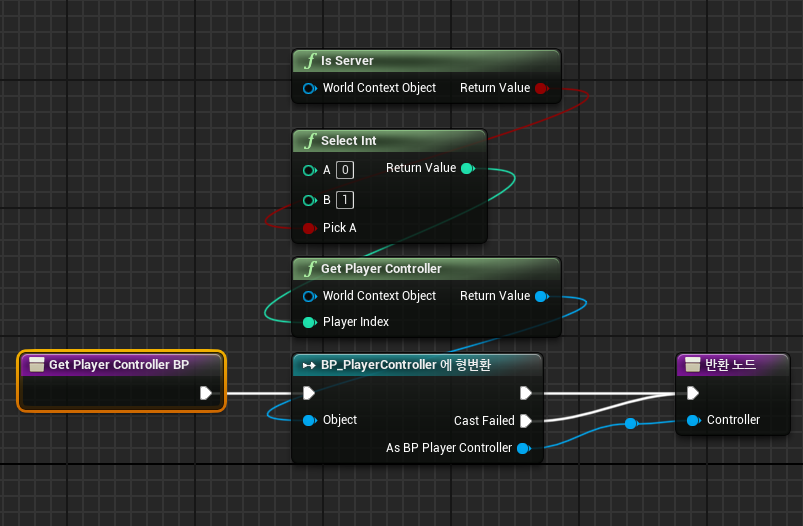
첫 번째 에러가 발생하는 이유는 GetPlayerController 노드의 동작 방식에 있습니다. GetPlayerController 는 로컬에서 실행되는 컨트롤러를 반환합니다. 싱글 플레이어 게임은 플레이어도 한 명, 컨트롤러도 한 개라서 문제가 되지 않지만, 멀티 플레이 환경에서는 서버와 각 클라이언트가 고유의 플레이어 컨트롤러를 소유하게 됩니다.

두 번째 에러인 GetGameMode 호출 시 None 이 반환되는 문제는 GetGameMode 는 서버에서만 사용할 수 있는 함수이기 때문입니다. GameMode 는 복제가 되는 속성도 아니며, 절대 복제되어서는 안되기 때문에 서버에서만 존재합니다.

따라서 클라이언트에서 GetGameMode 를 호출하게 되면 None 을 반환하는 것은 당연합니다.

첫 번째 에러 메세지를 해결하기 위해 제가 생각한 해결 방법은 다음과 같습니다.

1. IsServer 노드, 혹은 HasAuthority를 사용하여 현재 실행되는 환경이 서버인지 클라이언트인지 구분한다.
2. 서버에서 실행할 때에는 플레이어 인덱스 0 이 서버의 로컬 플레이어이기 때문에 인덱스를 정확하게 지정하여 컨트롤러를 가져온다.
3. 클라이언트에서는 클라이언트의 개수와 몇 번째 클라이언트인지에 따라 인덱스를 다르게 하여 가져온다.



위의 저의 계획대로 Get Player Controller BP 함수를 위와 같이 수정해보았습니다.

하지만 에러 메세지는 그대로였습니다.

생각한 다른 방법은 RPC를 사용하여 Get Player Controller 가 필요한 상황을 최대한 줄이고, 서버에서 최대한 안전하게 처리할 수 있도록 고치는 것입니다.

이 방법을 시도해보고 싶었으나 시간에 맞추지 못했습니다.

두 번째 에러인 GetGameMode 함수 호출에 None 이 반환되는 결과는 다음과 같이 고칠 구상을 했습니다.

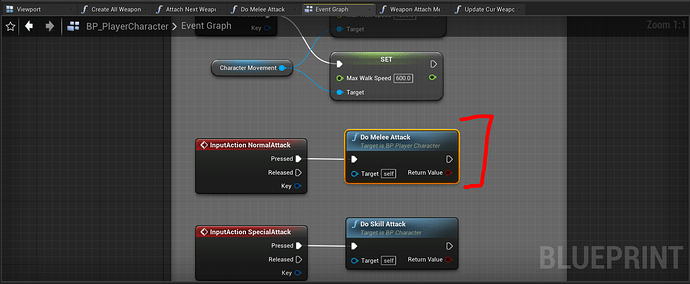
1. 클라이언트에서는 GameMode 를 호출하지 않게하는 로직을 추가한다
2. 서버에서는 GetGameMode 를 사용하고, 클라이언트에서는 필요한 경우, GetGameState 노드 호출로 대체한다.
3. GameState 는 서버와 클라이언트 간에 동기화되는 클래스이기 때문에 필요한 데이터를 GameState에 저장하여 GameMode 의 존재를 대체한다.
4. 위의 방법이 잘 안되는 경우 RPC를 통해 클라이언트가 필요한 모든 GameMode 관련 정보를 서버에 요청한다.

위와 같은 계획을 구상하였으나, 역시 멀티 플레이에 대한 이해도 부족으로 시간 내에 문제를 해결하지 못했습니다.

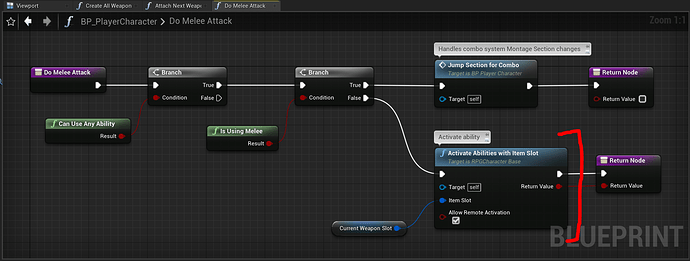
1. **GA\_PlayerAxeMelee 어빌리티가 마우스 클릭때부터 적을 때려서 데미지를 입히는 과정을 문서로 기술해 주세요. 확인 과정에서 왜 Client에서는 좌클릭시 공격을 안하고, 우클릭시 제자리에서 구르기만 하고 있는지 이유를 설명해주세요.**

**공격 과정 정리**

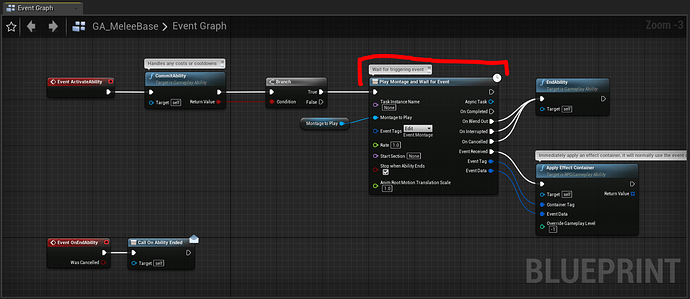
1. BP\_PlayerCharacter 클래스(BP)에서 NormalAttack 이벤트 발생으로 Do Melee Attack 함수가 호출됩니다.



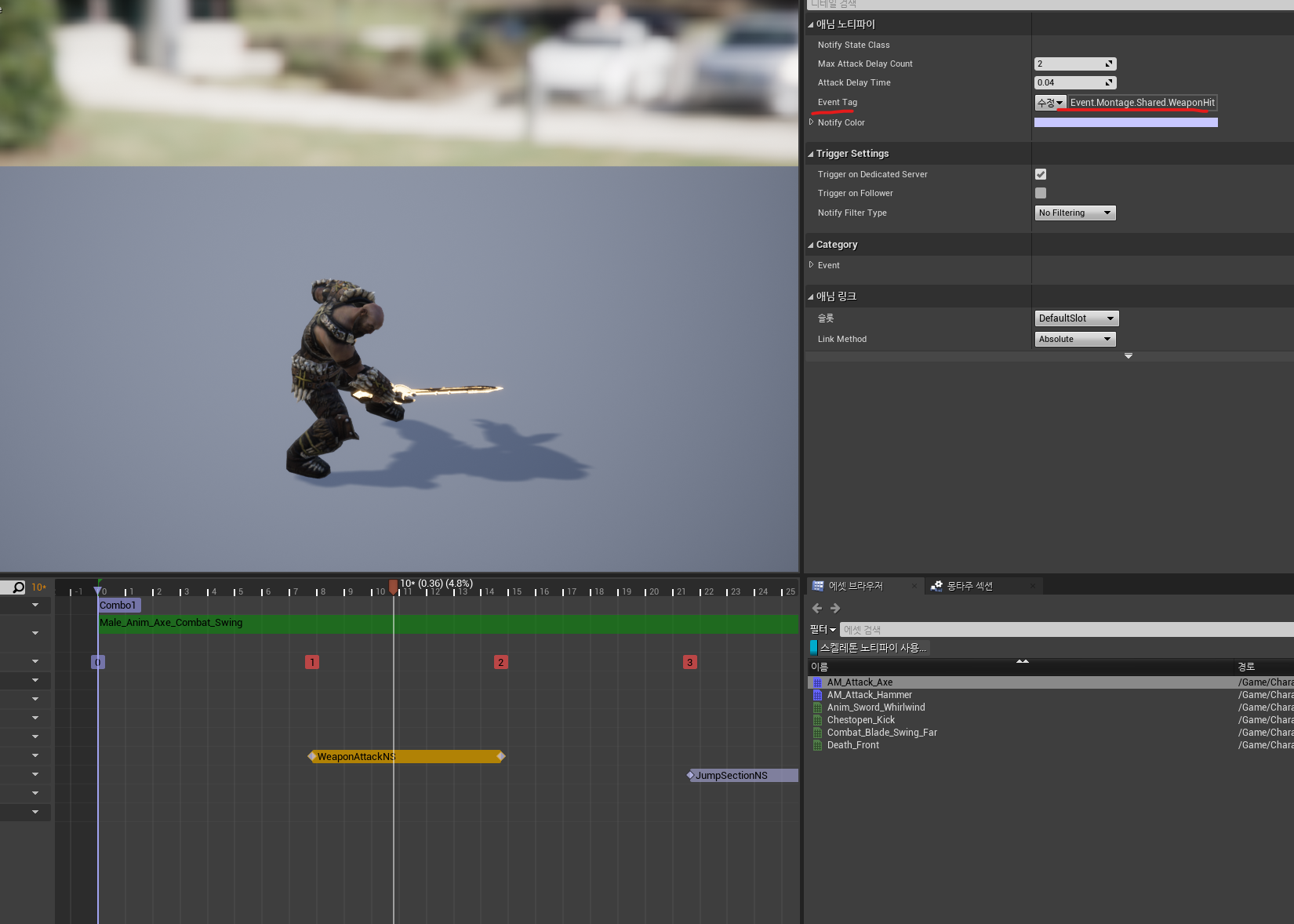
1. Do Melee Attack 함수는 ARPGCharacterBase 클래스(C++)의 Activate Abilities with Item Slot 함수를 호출합니다.



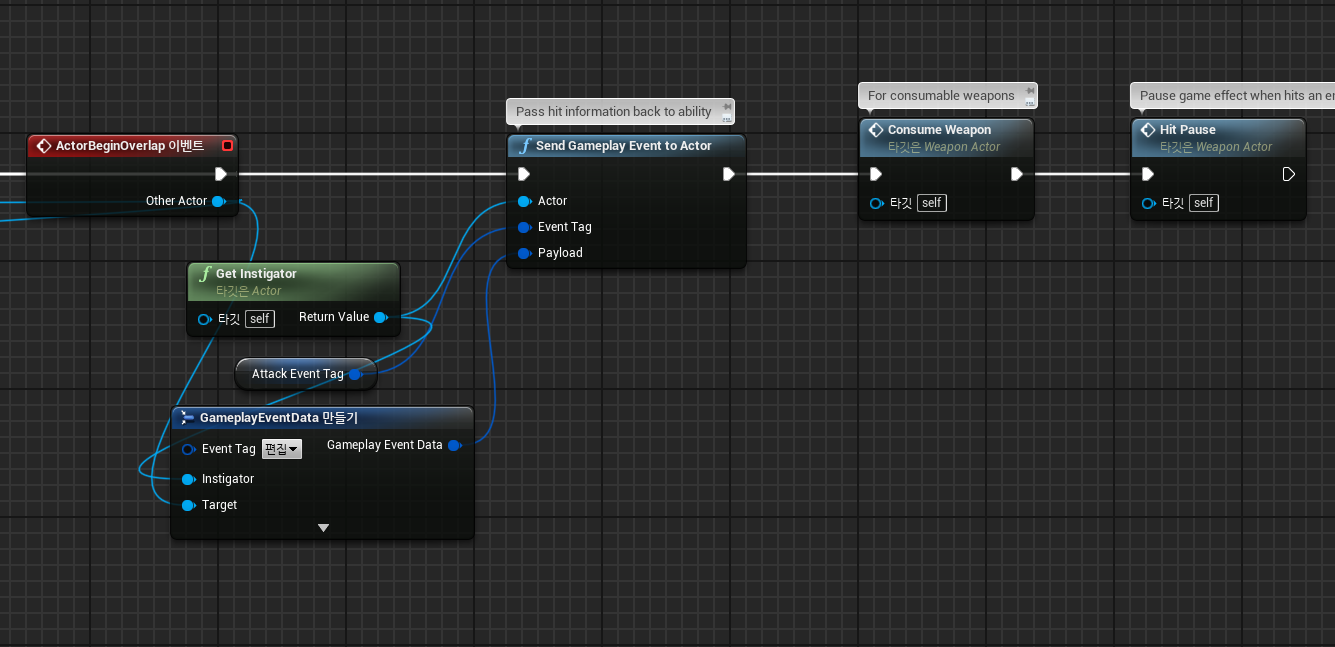
1. 해당 함수에서 아이템 슬롯의 무기(Axe, Hammer, Sword) 에 따라 GA 를 실행합니다. (AbilitySystemComponent->TryActivateAbility)
2. 어빌리티가 시작됨에 따라 GA\_MeleeBase (BP) 의 ActivateAbility가 호출됩니다. 이 때 CommitAbility 에서는 어빌리티의 Cost 및 Cooldown 를 적용합니다.



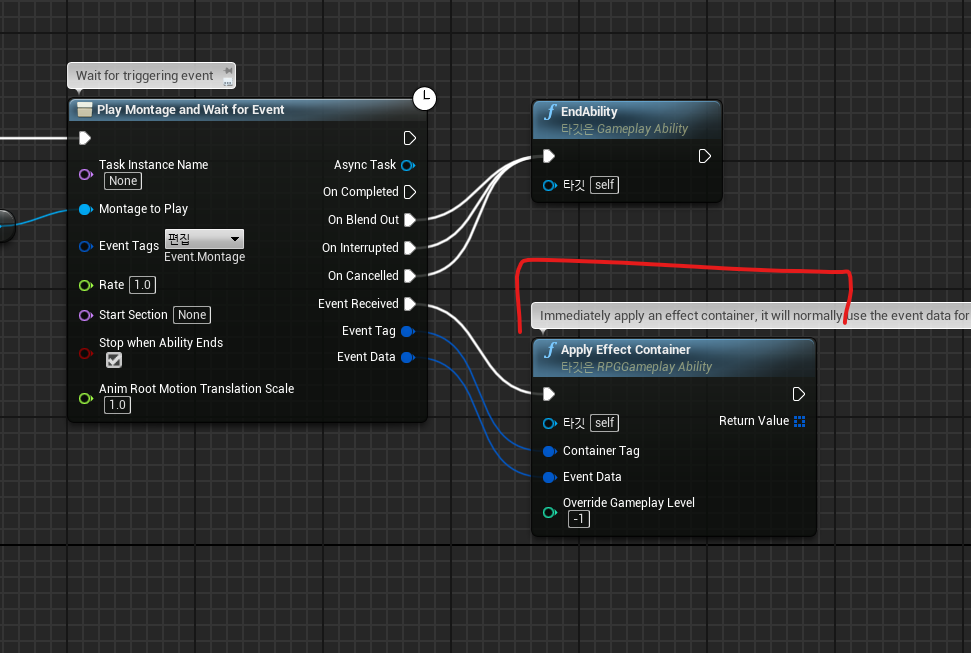
1. GA\_MeleeBase (BP) 의 PlayMontageandWaitForEvent 는 URPGAbilityTask\_PlayMontageAndWaitForEvent (C++) 에 해당하는 AbilityTask 노드입니다.
2. 위의 Task 에서는 일단(Activate) 몽타주를 재생하고, ASC 에서 GE가 방출될 때까지 대기합니다.
3. 몽타주(AM\_Attack\_Axe) 재생 중에 WeaponAttackNS 라는 이름의 Anim NotifyState가 실행됩니다. 해당 구간에서는 weapon에 충돌 판정이 생기고, WeaponActor(BP) 클래스의 Attack Event Tag 변수의 값은 Event.Montage.Shared.WeaponHit 로 설정됩니다.



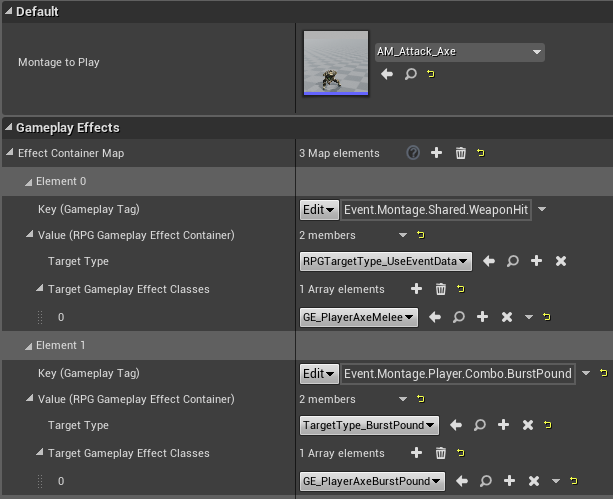
1. 위의 Notify State의 영향으로 충돌 판정이 생긴 상황에서, WeaponActor(BP)의 인스턴스가 오버랩되는 대상이 있다면, Actor BeginOverlap 이벤트가 발생합니다.
2. Actor BeginOverlap 이벤트가 발생하면, Make GameplayEventData 노드를 통해 페이로드(EventData)를 생성하고, Send Gameplay Event to Actor 노드를 통해 Target Actor 와 Instigator 에게 발생한 이벤트(Event.Montage.Shared.WeaponHit)를 전송합니다.



1. GE가 방출되었으니 GA\_MeleeBase(BP)에서 대기 중이던 PlayMontageandWaitForEvent Task 는 방출된 GE 와 전달된 EventTags 가 일치하는지 확인하고 태그 및 페이로드(EventData)와 함께 EventReceived 실행 핀을 가동하고 ApplyEffectContainer 함수를 호출합니다.



1. GA\_MeleeBase(BP)의 ApplyEffectContainer 노드는 URPGGameplayAbility(C++) 클래스의 ApplyEffectContainer 함수에 해당하며, 인자로 전달된 Tag 와 일치하는 FRPGGameplayEffectContainer 를 EffectContainerMap에서 찾고, EventData와 EffectSpecs 목록이 포함된 FRPGGameplayEffectContainerSpec 를 만듭니다. 또한, 이를 적용하여 타겟의 Attribute 에 영향을 줍니다 (데미지를 가함).
2. 위에서 언급한 EffectContainerMap는 URPGGameplayAbility 클래스의 TMap 으로, Tag 와 FRPGGameplayEffectContainer 구조체를 pair로 저장합니다. FRPGGameplayEffectContainer 구조체에는 타깃 정보와 적용할 GE 목록이 포함되어 있습니다. GA\_PlayerAxeMelee 를 보면 발동 조건의 Tag 를 Key 값으로 하고 그에 따른 이펙트와 타겟을 설정한 FRPGGameplayEffectContainer 구조체의 형태를 볼 수 있습니다.



1. 정리하면 다음과 같습니다.

공격 몽타주 실행 -> 게임플레이 이벤트 emit -> RPGTargetType\_UseEventData 타겟팅 클래스를 통해 GE\_PlayerAxeMelee 이펙트 실행 -> URPGDamageExecution (C++) 을 실행하여 수학 연산을 한다 -> Attribute 에 영향을 준다 (Stat 에 변동이 생긴다)

**클라이언트 동기화 문제 분석**

클라이언트에서 좌클릭 시 아무 행동도 나가지 않고, 우클릭 시에는 제자리에서 구르는 문제의 원인은 다음과 같다고 생각합니다.

1. 좌클릭 시 공격이 나가지 않는 문제

* 위의 공격 과정에서 시작 부분에 Activate Abilities with Item Slot 를 통해 손에 쥐고 있는 무기 아이템에 따라 다른 Ability 를 동작시키는 것을 알 수 있습니다.
* 손에 무기를 쥐는 것은 BP\_PlayerController 의 이벤트 On Possess 에 연결된 노드인 Create All weapon 에서 진행됩니다.
* 해당 함수에서는 GetPlayerController BP 가 사용되는데, 리슨 서버 환경에서는 클라이언트가 제대로 된 Player Controller 를 얻지 못하는 문제가 남아있습니다.
* 따라서 클라이언트에서는 생성된 무기가 없으니, 실제로 무기를 들고 있지 않으며 실행할 어빌리티도 없습니다.
* 이 외에도 서버가 아닌 클라이언트에서 GA 가 실행될 때 신경 써야하는 부분과 리플리케이션 옵션을 켜야하는 클래스인데 놓치고 키지 못한 경우가 있을 수 있어서 다양한 가능성을 염두해야할 것 같습니다.

2. 구르기 애니메이션이 제자리에서 멈추는 문제

* 이 문제는 일차원적으로는 클라이언트와 서버 간의 애니메이션 동기화 및 행동 권한의 문제로 생각됩니다.
* 클라이언트에서 애니메이션을 재생할 때, 그 동작이 서버와 동기화되지 않으면 서버에서 해당 애니메이션이 발생한 것으로 간주되지 않기 때문에 제자리에서만 애니메이션이 실행되고, 서버에는 아무 변화도 보이지 않는 현상이 발생할 수 있습니다.
* 멀티플레이 환경에서 이런 클라이언트의 캐릭터의 주요한 상태 변화(이동, 공격, 스킬) 등은 보통 서버가 결정하고, 클라이언트에게 전달하는 형태로 구조를 만듭니다.
* 따라서 이 문제를 해결하기 위해서는 서버의 권한을 늘리는 방향으로 해결하면 좋을 것 같습니다. 클라이언트에서 애니메이션을 직접 실행하는 것이 아니고, 서버에서 실행되도록 설정해야 합니다.
* RPC 를 사용하여 클라이언트에서 서버로 애니메이션 재생 요청을 보내고, 서버에서 실제로 애니메이션을 재생, 그 후에 그 결과를 모든 클라이언트에게 브로드캐스트하는 방식을 이용한다면 , 모든 클라이언트와 서버가 동일한 애니메이션을 볼 수 있을 것 같습니다.
* 또한, 구르기가 발동되는 것도 GA 가 발동되는 것의 결과이기 때문에, 멀티플레이 환경에서 신경써야하는 GAS 의 특성을 놓쳤을 가능성이 있습니다. 이 부분도 폭 넓게 고려해야 할 것 같습니다.

1. **GA\_PlayerAxeMelee 가 동작하는 모든 관련 기능, Gameplay Effect, Cue, Gameplay Tag들을 CPP로 재작업해주세요. Class 이름, Tag이름 등 네이밍은 기존 네이밍에 마지막에 \_CPP Postfix 붙여서 네이밍 해주세요. (멤버 변수는 상관없습니다.)**

>

재작업한 BP 는 다음과 같습니다 :

GA\_AbilityBase (BP) -> **GA\_AbililtyBase\_CPPPostfix (C++)**

GA\_MeleeBase (BP) -> **GA\_MeleeBase\_CPPPostfix (C++)**

GAS 에 대한 이해도가 부족하여 완성하진 못한 상태로, 실제로 구동되기 위해서는 추가적인 작업이 더 필요합니다. GE 컨테이너를 만드는 부분을 어떻게 처리해야할지 잘 모르겠습니다.

1. **3번과 같이 진행해도 됩니다. Client부분도 문제 없이 공격, 구르기, InGameUI뜨도록 고쳐주세요.**

>

InGameUI 는 고쳤습니다만, 공격과 구르기의 경우 위에서 언급한 문제들이 해결되지 못했기 때문에 고치지 못했습니다. 원인은 짐작이 가지만 멀티플레이와 GAS 둘 다 이해하고 고치기에는 시간이 부족했습니다.

1. **추가: 과제 진행중 아래 링크에 대해서 공부하신 내용이 있다면 대해 정리해주세요**  
   <https://github.com/tranek/GASDocumentation>

**게임플레이 어빌리티 시스템의 주요 개념**

* 액터가 소유하고 발동할 수 있는 어빌리티 및 액터 간의 인터랙션 기능을 제공하는 프레임워크
* 유연성과 확장성, 모듈러 시스템, 네트워크 지원, 데이터 기반 설계 등이 장점

**게임플레이 어빌리티 시스템의 핵심 요소**

| 어빌리티 시스템 컴포넌트 (ASC) | GAS 프레임워크를 관리하고 처리하는 중앙 처리 장치의 역할 |
| --- | --- |
| 게임플레이 태그 (Tag) | 객체의 상태를 분류하고 설명하는 이름. |
| 게임플레이 어빌리티 (GA) | 모든 스킬, 액션들이 발동해서 시작 |
| 게임플레이 이펙트(GE) | GA 실행에 대한 영향을 계산하고 시스템에 알림 |
| Attribute | 캐릭터의 스탯을 비롯한 수치 데이터. GE 에 의해 영향을 받는다. |

**게임플레이 어빌리티 시스템의 시작**

1. ASC 를 액터에 부착.

> 본 프로젝트에서는 ARPGCharacterBase 생성자 단계에서 부착했다.

AbilitySystemComponent = CreateDefaultSubobject <URPGAbilitySystem Component>(TEXT("AbilitySystemComponent"));

1. GA 를 정의.

> 본 프로젝트에서는 RPGGameplayAbilitiy(C++) 를 상속받은 GA가 Blueprint 형태로 선언 및 정의됨.

1. ASC 의 GiveAbility 함수에 발동할 GA의 타입을 전달하여 어빌리티를 등록

> 본 프로젝트에서는 AddSlottedGameplayAbilites() 함수에서 실행. Tag 가 있는 모든 GA를 ASC 에 등록하는 과정이 있음.

1. ASC 의 TryActivateAbility 함수에 발동할 GA의 타입을 전달 > 어빌리티 발동
2. EndAbility : 스스로 어빌리티를 마무리