어빌리티 시스템은 정의되지 않은 어떤 추상화된 능력들을 구체화시키는 데 사용되는 프레임워크입니다. 캐릭터가 사용할 스킬이나 버프/디버프 등의 상태이상 등을 AbilityTask 클래스를 사용하여 수명주기를 조절하며 구현할 수 있습니다.

**어빌리티 태스크, 어빌리티 시스템, 상호작용 이벤트**

어빌리티 태스크는 NBattleAbilityTask 추상클래스에서 파생됩니다. 생성부터 소멸까지 모든 단계에서 외부 여러 델리게이트와 연동되어 호출 받고 구현됩니다. 게임 내에서 공격력 강화와 같은 능력에 대한 활성/비활성화는 Activate/Deactivate 단계에서 주로 구체화되어 수행됩니다. 핵심 기능을 요약하면 다음과 같습니다.

* 활성 및 쿨타임 주기 제어
* 자체 수명 주기 제어
* 생성 가능 조건 제어
* 생성 ~ 소멸 내 각 단계별 디테일 구현

어빌리티 태스크의 생성은 어빌리티 시스템에서 관리합니다. 어빌리티 태스크 내부의 일련의 이벤트들은 어빌리티 시스템에 의해 델리게이트로 외부에 제공됩니다. 이로써 어빌리티 태스크는 다른 어빌리티 태스크의 이벤트를 받아서 처리하는 식으로 상호작용할 수 있습니다. 주요 이벤트는 다음과 같습니다.

* 초기화 ~ 소거 (Initialize ~ Cleanup)
* 활성화 시작 ~ 종료 (ActiveBegin ~ ActiveEnd)
* 쿨타임 시작 ~ 종료 (CoolingdownBegin ~ CoolingdownEnd)

|  |
| --- |
|  |

**상태이상 어빌리티**

스턴(Stun), 빙결(Freeze), 공격력 증가, 방어력 감소 등 일시적인 캐릭터의 이상 상태를 어빌리티 태스크로 구현 가능합니다. 먼저 새로운 NBattleStatusEffectAbilityTask 클래스를 만들고 NBattleAbilityTask 를 오버라이드 하는 것으로부터 시작해서 아래 사항들을 차례대로 구현합니다.

* 소유 캐릭터 설정, 활성화 주기, 쿨타임 주기 등 변수 초기화
* 주요 함수 오버라이드 : OnRegister, OnInitialize, OnCleanup, OnRunningBegin, OnRunningEnd, OnActiveBegin, OnActiveEnd, OnCoolingdownBegin, OnCoolingdownEnd, OnCalculateNextActiveTurnCount, OnCalculateNextCoolingdownTurnCount
* 턴 관련 주요 이벤트 연동 및 호출 : OnStartTurn, OnEndTurn
* 오버라이드 함수에서 루아(Lua) 함수 호출, 호출된 루아 함수에서 효과 구현
* 호출된 루아 함수에서 효과 구현

C++ 의 오버라이드 함수들은 관련 루아 함수를 호출하고 고유 데이터를 넘기는 데에만 집중합니다. 루아를 사용하지 않는다면 물론 C++ 파생 클래스를 생성해서 관리하는 것도 가능합니다.

|  |
| --- |
| Void NBattleStatusEffectAbilityTask::**OnActiveBegin()**  {  **CallLuaFunction**(LuaFunctionName, … ); // 루아 함수 이름, 기타 고유 데이터  }  … |

호출된 루아 함수에서는 실제 효과를 구현합니다. 공격력 증가 효과를 원한다면 어빌리티 활성화 단계에서 추가적인 공격력을 부여할 것이고 비활성화 시에 부여한 공격력을 감소하면 될 것입니다.

|  |
| --- |
| function **LuaStatusEffect\_AtkUp\_OnActiveBegin**(InAbilityTask, InTaskUniqueId)  local CurrentAtk = BattleLuaFunctionLibrary.GetCurrAtk(InAbilityTask:GetOwningHero())  local **NewlyCalculatedAtk** = BattleLuaFunctionLibrary.CalculateStatValueByPer(…)  BattleLuaFunctionLibrary.**SetAtk**(InAbilityTask:GetOwningHero(), **NewlyCalculatedAtk**)  End  function **LuaStatusEffect\_AtkUp\_OnActiveEnd**(InAbilityTask, InTaskUniqueId)  local CurrentAtk = BattleLuaFunctionLibrary.GetCurrAtk(InAbilityTask:GetOwningHero())  local **RestoredAtk** =CurrentAtk - CachedNewlyCalculatedAtk  BattleLuaFunctionLibrary.**SetAtk**(InAbilityTask:GetOwningHero(),**RestoredAtk**)  end |