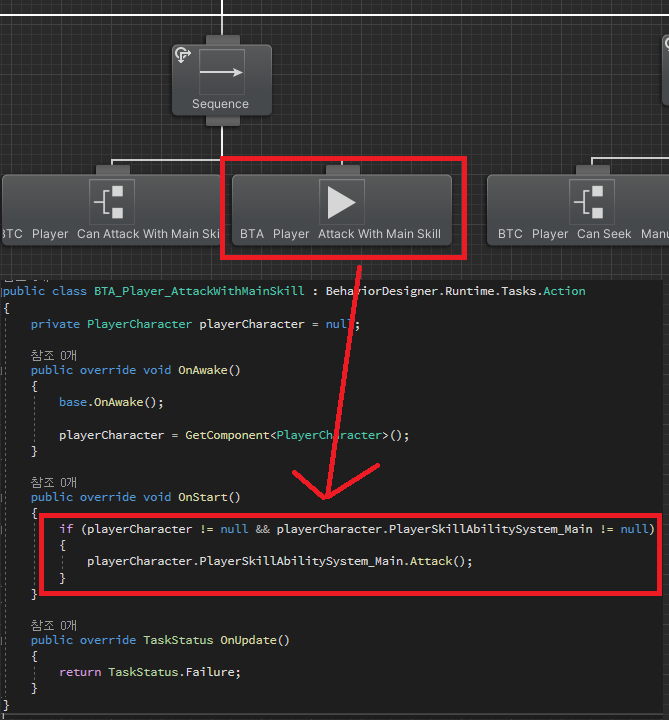
□ 시작 지점 : Behavior Tree



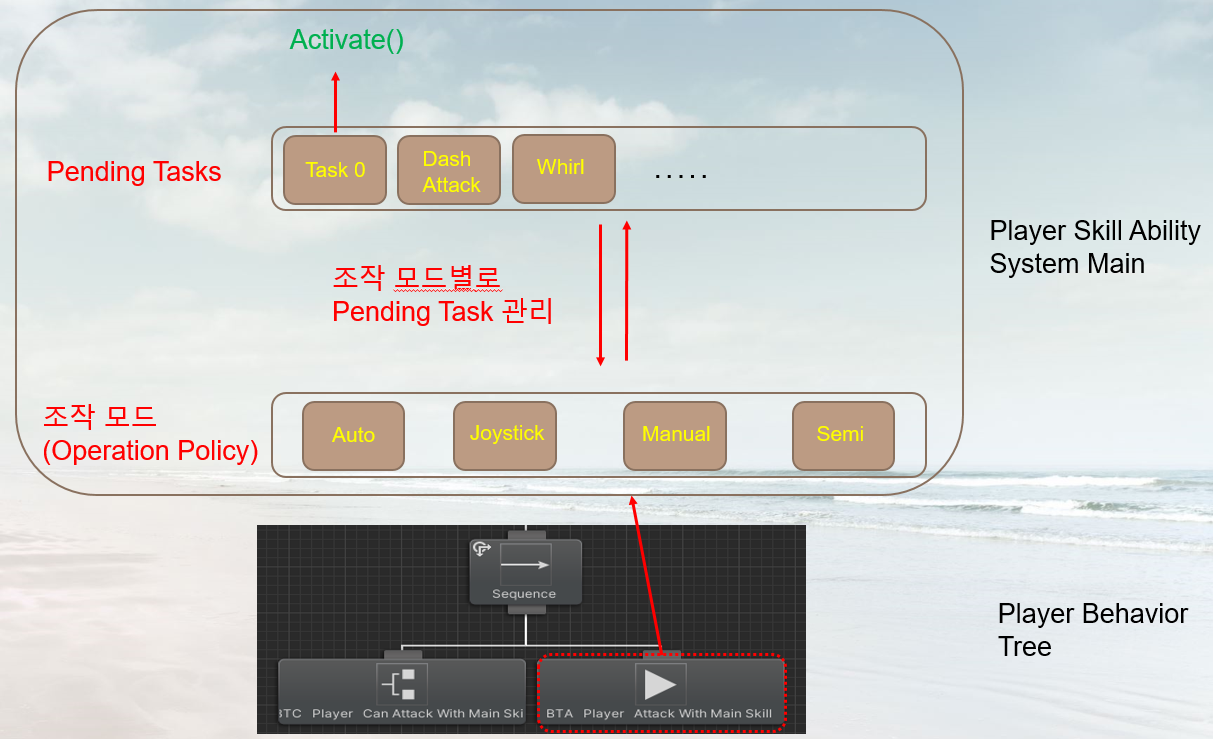
; BTA Player Attack With Main Skill 액션 노드 진입

; 플레이어 스킬 어빌리티 시스템의 Attack() 호출.

; Attack() 호출되면 최종적으로 조작 모드별로 관리되는 Pending Task 를 실행.

; 자세한 내용은 아래 내용 참조.

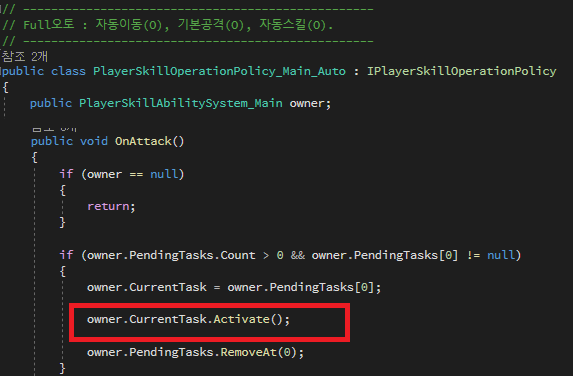
□ Player Skill Ability System Main



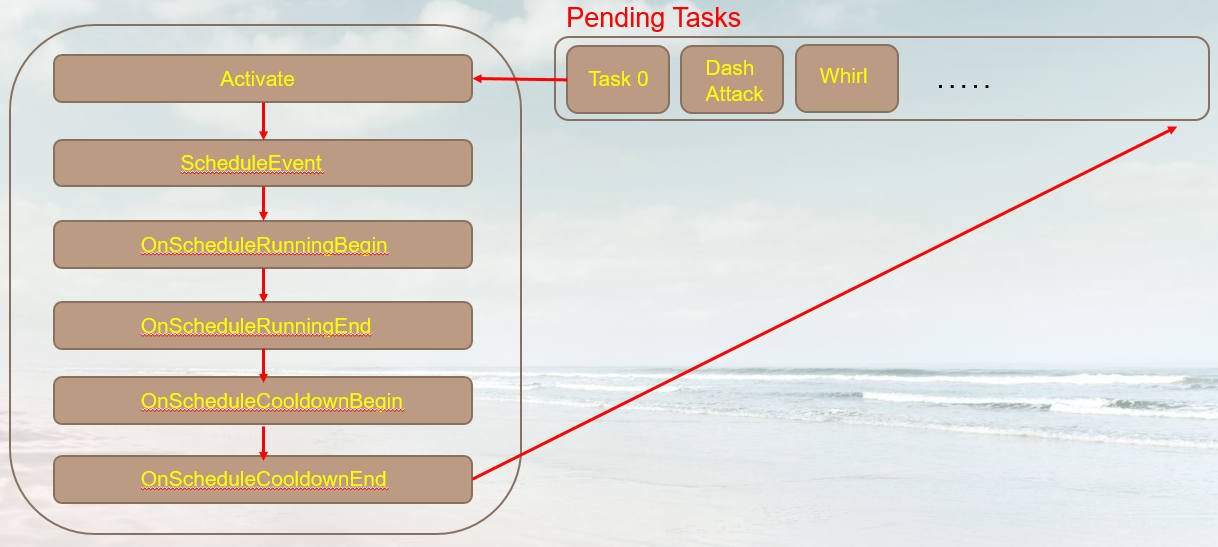
; Pending Tasks 는 조작 모드별로 관리됨.

. 보유 태스크의 순서 책정, Add / Delete 등의 처리.

; Pending Tasks 의 첫 번째 태스크를 실행(Activate) 하여 실질적인 Attack 처리.



□ Player Skill Task

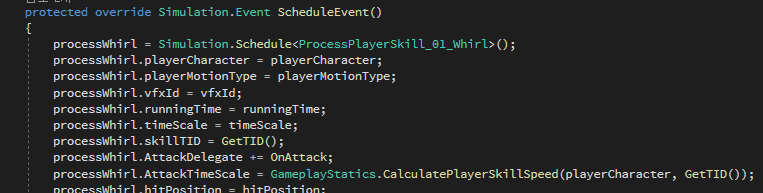


; Pending Tasks 의 첫 번째 Task0 을 Pop.

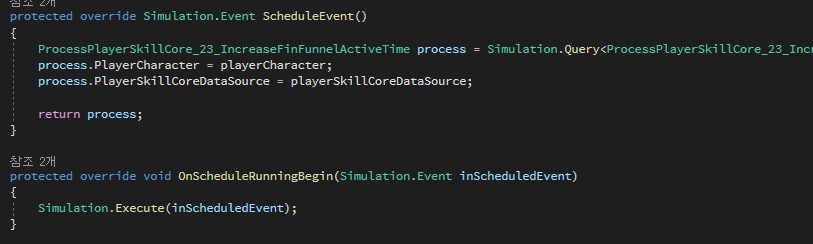
; Task0 을 Activate() 호출하여 라이프사이클 활성화.

; ScheduleEvent : Simulation.Event 기반의 컨텍스트 클래스 생성 & 호출(스킬 실 코드).

. 다음 프레임에 실행 : Simulation.Schedule<> 사용



. 즉시 사용 : Simulation.Query<> ~ Simulation.Execute()



**\* 주의 : Query 사용 시 ScheduleEvent() 안에서 Simulation.Execute() 사용하면 안됨. ScheduleEvent() 호출 후 Cleanup 델리게이트를 등록하게 되는데 이 과정이 끝나야 함. 그렇지 않으면 Cleanup 처리가 제대로 이뤄지지 않아 문제 발생함.**

; OnScheduleRunningBegin : Simulation.Event 실행 예정.

; OnScheduleRunningEnd : Simulation.Event 실행이 종료 됨.

; OnScheduleCooldownBegin : 쿨타임에 들어감.

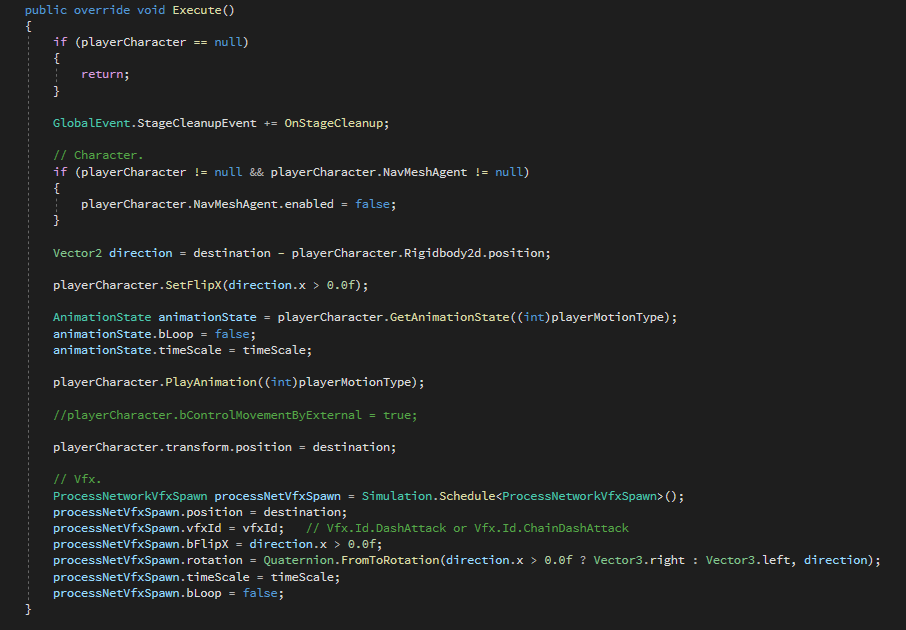
; OnScheduleCooldownEnd : 쿨타임 종료 되고 PendingTasks 에 규칙에 따라 재등록 됨.

□ Player Skill Processor

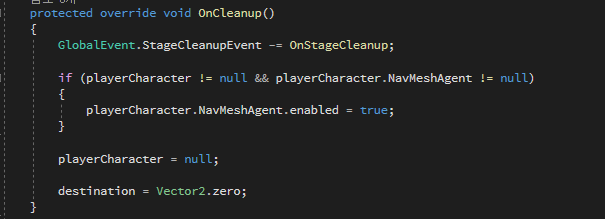
; Simulation.Event 또는 Simulation.DeferredEvent 기반의 스킬 처리 용 컨텍스트 클래스.

; Interface

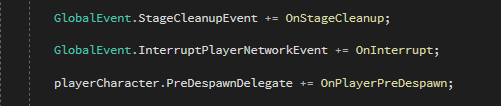
. Execute : 종료 조건 설정, 애니메이션/이펙트 출력, 충돌 검출, 데미지 계산 등 해당 스킬에 관련된 실 코드 작성.



. OnCleanup : 클래스 반환 전 초기화 작업.



; 종료 조건 : **해당 스킬을 무효화해야 하는 시점을 잘 잡아서 제대로 종료 처리 해야 함**. 보통 플레이어 Despawn, 스테이지 종료 시 또는 Death 상황.



; 종료 조건에 대한 설정과 종료 처리를 Owner 클래스가 하지 않고 여기서 하는 이유는 최대한 유연하게 처리하기 위함임. Owner 클래스에서는 호출만 하지 종료에 대해서는 전혀 모름(책임 전가 처리).