□ Event

; 컨텍스트 클래스를 Pool 에서 생성하여 순서 있는 처리를 합니다.

; 주로 캐릭터 애니메이션 호출, 사운드 출력, 이펙트 호출 등 한 곳에서 종합적으로 처리하고자 할 때 사용합니다(로직 통합/분리 목적).

; 어떤 작업들이 실행되고 있는지 또는 실행될 예정인지 확인하기 용이합니다.

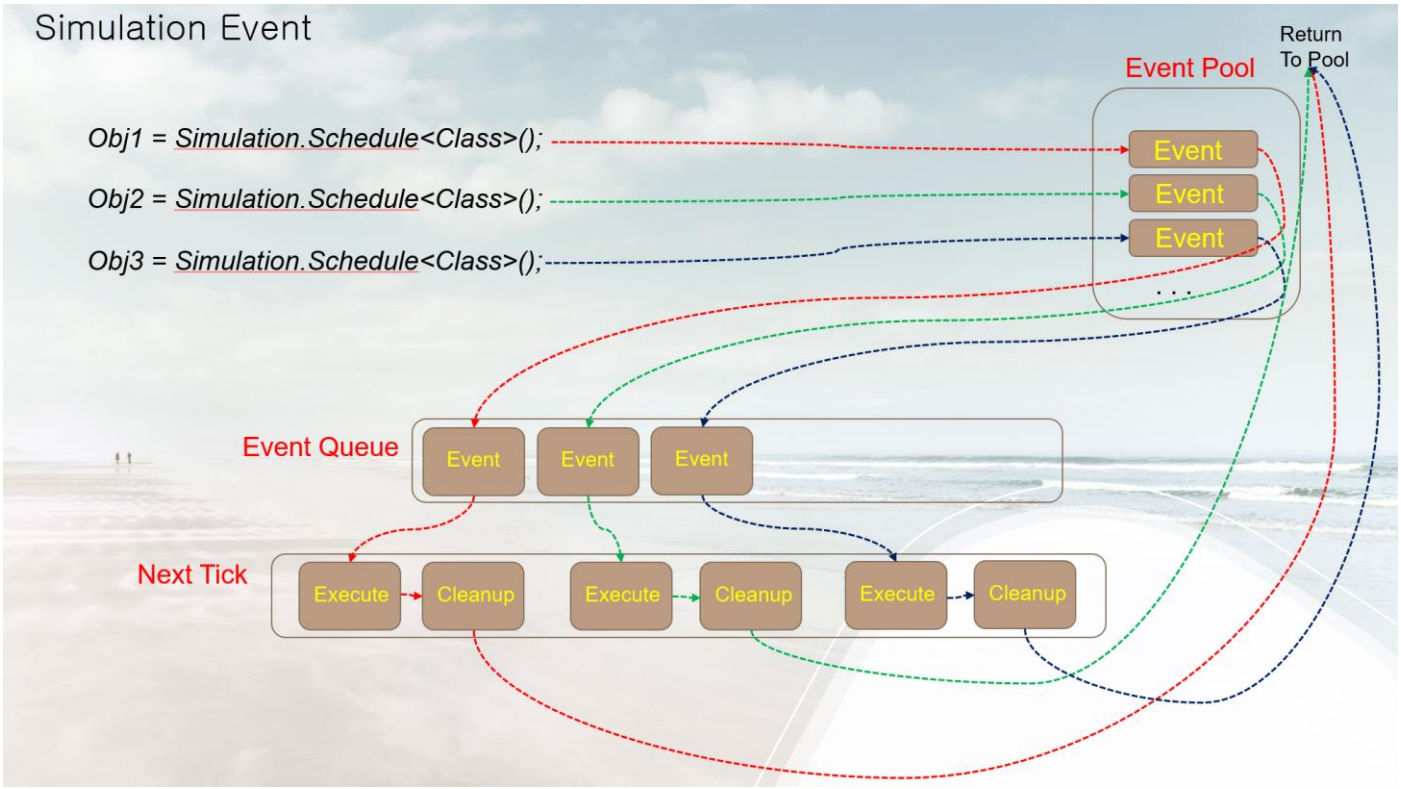
; Schedule

. **다음 프레임에 실행**됩니다.

. 다음 프레임에 실행되면(Execute()) 바로 Cleanup() 메소드가 호출된 후 반환됩니다. 별도로Cleanup 호출할 필요 없음.

. 보통 한 프레임에 한 번만 호출될 내용들 위주로 사용합니다.

. 호출하는 프레임에서 즉시 사용되어야 한다면 Simulation.Query 구문을 사용합니다.



; Query

. Simulation.Query<ClassName>() 로 클래스를 얻고 Simulation.Execute() 호출하여 바로 실행합니다.

. 현재 프레임에서 실행된다는 점 외 Schedule 과 동일합니다.

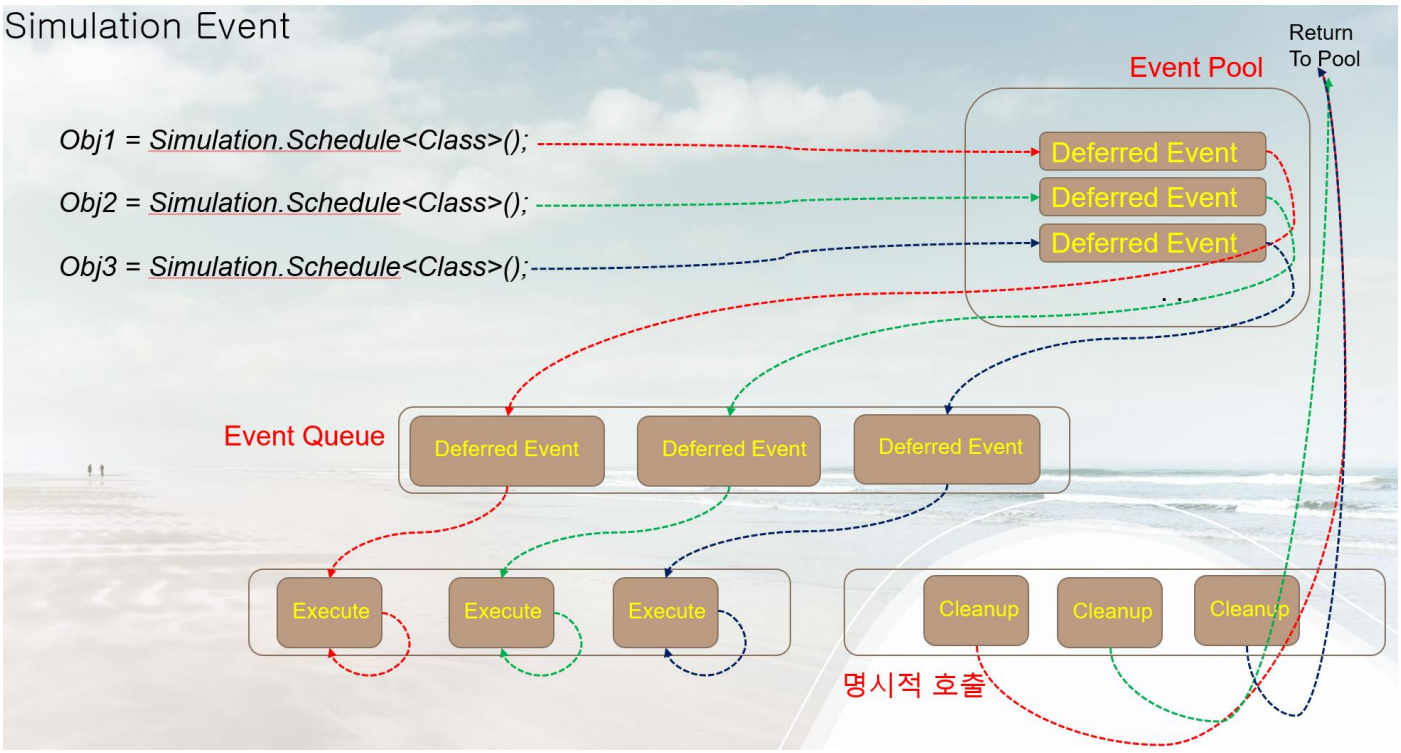
. 네트워크 플레이어 스킬과 같은 경우 Delay 를 최소화하기 위해 Query 를 사용합니다.

□ Deferred Event

; Event 와 달리 Execute() 호출 후 바로 Cleanup() 이 호출되지 않고 클래스가 반환되지도 않습니다.

; 반환을 원하는 시점에 명시적으로 Cleanup() 을 호출합니다.

; 주로 특정한 시간 동안 실행되어야 할 때 사용합니다.



□ 소멸 조건 처리

; Event 의 경우 Execute() 호출 후 바로 Cleanup() 및 반환 처리가 이루어지기 때문에 특별히 처리할 필요는 없습니다.

; Deferred Event 는 명시적으로 Cleanup() 을 호출 해야하기 때문에 각 구현에 맞게 소멸/반환 조건 처리를 해줍니다.

; 플레이어 스킬의 경우 보통 스테이지 종료 또는 플레이어 디스폰 또는 해당 애니메이션 종료 등의 콜백을 등록하고 호출 받을 시 Cleanup 처리를 합니다.

; Cleanup 처리를 하지 않게 되면 이벤트 큐에 계속 등록되어 실행되고 있으므로 프레임 저하를 유발할 것임에 유의합니다.

```
참조 5개
public override void Execute()
{
    if (playerCharacter == null)
    {
        Cleanup();
        return;
    }

    GlobalEvent.SkillLogging_Start?.Invoke(loggingSkillType);

    GlobalEvent.StageCleanupEvent += OnStageCleanup;

    playerCharacter.PreDeathDelegate += OnPlayerDead;

    playerCharacter.PreDespawnDelegate += OnPreDespawn;

    bool bOk = Attack();

    if (bOk)
```

```
참조 2개
private void OnStageCleanup()
{
    GameplayStatics.StopSkakeMainCamera();

    Cleanup();
}

참조 2개
private void OnPlayerDead(IDeathable inDeathable)
{
    GameplayStatics.StopSkakeMainCamera();

    Cleanup();
}

참조 2개
private void OnPreDespawn(ISpawnable inSpawnable)
{
    Cleanup();
}
```

; Simulation.cs 의 static public void Execute(Simulation.Event evt) 의 runningEvents 에 브레이크 포인트를 걸어서 불필요한 Deferred Event 가 실행되고 있는지 확인해 봅니다. 보통 웨이브나 스테이지 변경 시 한 번씩 찍어서 살펴봅니다.

```
    참조 90개
    static public void Execute(Simulation.Event evt)
    {
        lock (cs)
        {
            runningEvents.Add(evt);

            var tick = evt.tick;

            evt.ExecuteEvent();

            if (evt.tick > tick)
            {
                //event was rescheduled, so do not return it to the pool.
            }
            else
            {
                // Debug.Log($"<color=green>{ev.tick} {ev.GetType().Name}</color>");
                evt.Cleanup();

                try
                {
                    eventPools[evt.GetType()].Push(evt);
                }
                catch (KeyNotFoundException)
                {
                    //This really should never happen inside a production build.
                    Debug.LogError($"No Pool for: {evt.GetType()}");
                }
            }
        }
    }
}
```