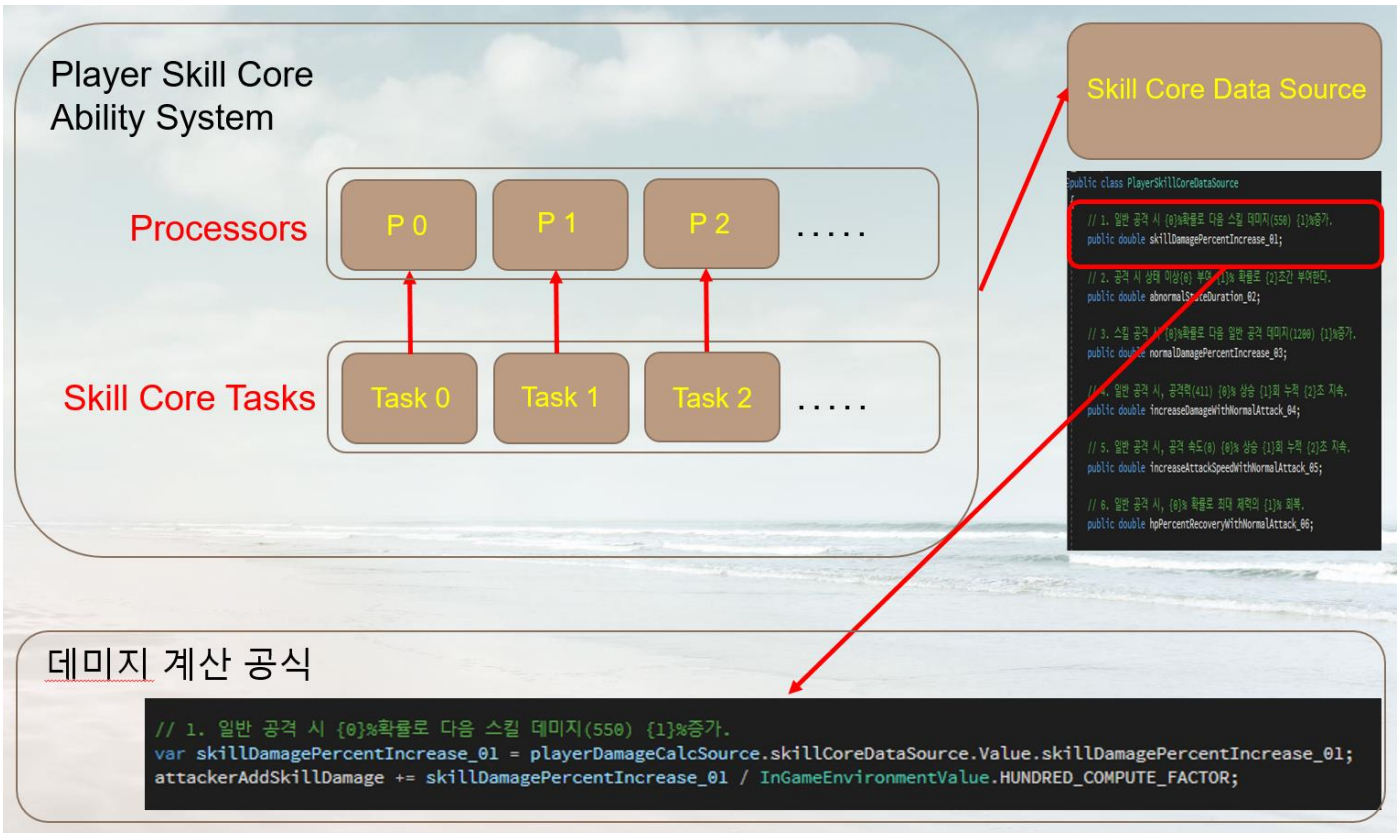
□ 스킬 코어 시스템 구조



; 스킬 코어 태스크를 활용한 스킬 코어 데이터 접근 및 변경이 핵심.

; 데미지 계산 등 외부에서 필요 시 데이터 접근하여 작업 수행.

□ 일반 공격 시 {0}%확률로 다음 스킬 데미지(550) {1}%증가

; 데이터 준비 : skillDamagePercentIncrease_01.

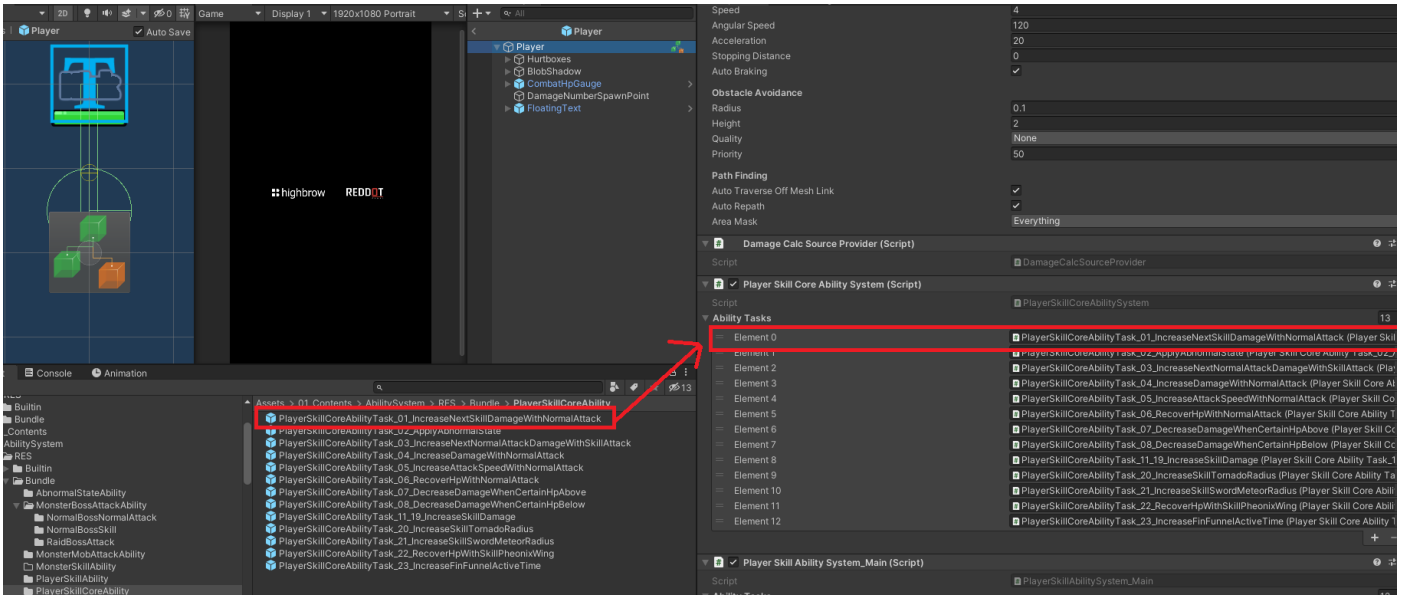
```
public class PlayerSkillCoreDataSource
{
    // 1. 일반 공격 시 {0}%확률로 다음 스킬 데미지(550) {1}%증가.
    public double skillDamagePercentIncrease_01;
```

; 태스크 클래스 작성.

```
public class PlayerSkillCoreAbilityTask_01_IncreaseNextSkillDamageWithNormalAttack : AbilityTask
    , IPlayerCharacterAccessable
    , IPlayerSkillCoreDataSourceAccessable
{
    #region Fields
    private PlayerCharacter playerCharacter;

    private PlayerSkillCoreDataSource playerSkillCoreDataSource;
    #endregion
```

; 태스크 프리팹 생성 및 연동.



; '일반 공격 시'에 대한 발동 조건 마련.

```
protected override void OnInitialize()
{
    cooldownTime = -1.0f;

    if (playerCharacter != null)
    {
        PlayerSkillAbilitySystem_Main playerSkillAbilitySystem_Main = playerCharacter.PlayerSkillAbilitySystem_Main;

        if (playerSkillAbilitySystem_Main != null)
        {
            playerSkillAbilitySystem_Main.abilityTaskRunningBeginDelegate += OnPlayerMainSkillRunningBegin;
            playerSkillAbilitySystem_Main.abilityTaskRunningEndDelegate += OnPlayerMainSkillRunningEnd;

            playerSkillAbilitySystem_Main.AttackDelegate += OnPlayerMainSkillAttack;
        }
    }
}
```

; 일반공격(Default Attack) 체크 -> Activate() -> ScheduleEvent().

```
참조 2개
private void OnPlayerMainSkillAttack(int inTID, int inCount)
{
    if (inTID == (int)PlayerSkillAbilityType.DefaultAttack)
    {
        Activate();
    }
}
```

; 프로세서 실행(다음 프레임).

```
protected override Simulation.Event ScheduleEvent()
{
    ProcessPlayerSkillCore_01_IncreaseNextSkillDamageWithNormalAttack process = Simulation.Schedule<ProcessPlayerSkillCore_01_IncreaseNextSkillDamageWithNor
    process.PlayerCharacter = playerCharacter;
    process.PlayerSkillCoreDataSource = playerSkillCoreDataSource;

    return process;
}
```

; **ProcessPlayerSkillCore_01_IncreaseNextSkillDamageWithNormalAttack**

. **'{0}%확률로 다음 스킬 데미지(550) {1}%증가'** 처리.

```
public override void Execute()
{
    if (playerCharacter == null)
    {
        return;
    }

    if (playerSkillCoreDataSource == null)
    {
        return;
    }

    IDamageCalcSource damageCalcSource = playerCharacter.DamageCalcSource;

    if (damageCalcSource != null && damageCalcSource is PlayerDamageCalcSource playerDamageCalcSource)
    {
        // 1. 일반 공격 시 {0}%확률로 다음 스킬 데미지(550) {1}%증가

        //dataSource.skillDamagePercentIncrease_01 = 0.0f;

        if (playerDamageCalcSource.GetUnitStat().modules.ContainsKey(Common.Const.eModuleType.next_skill_dmg_up_when_normal_atk))
        {
            var moduleList = playerDamageCalcSource.GetUnitStat().modules[Common.Const.eModuleType.next_skill_dmg_up_when_normal_atk];

            for (int index = 0; index < moduleList.Count; index++)
            {
                (double, double, double) values = moduleList[index];

                double probability = values.Item1;

                float randomizedValue = UnityEngine.Random.Range(0.0f, 100.0f);

                if (randomizedValue <= probability)
                {
                    double skillDamagePercentIncrease = values.Item2;

                    if (playerSkillCoreDataSource.skillDamagePercentIncrease_01 < skillDamagePercentIncrease)
                    {
                        playerSkillCoreDataSource.skillDamagePercentIncrease_01 = skillDamagePercentIncrease;
                    }
                }
            }
        }
    }
}
```

; 데이터 리셋 처리.

. 일반 공격 시에만 적용되는 사항이므로 다른 스킬 사용 시 데이터 리셋.

```
protected override void OnInitialize()
{
    cooldownTime = -1.0f;

    if (playerCharacter != null)
    {
        PlayerSkillAbilitySystem_Main playerSkillAbilitySystem_Main = playerCharacter.PlayerSkillAbilitySystem_Main;

        if (playerSkillAbilitySystem_Main != null)
        {
            playerSkillAbilitySystem_Main.abilityTaskRunningEndDelegate += OnPlayerMainSkillRunningEnd;

            playerSkillAbilitySystem_Main.AttackDelegate += OnPlayerMainSkillAttack;
        }
    }
}
```

```
private void OnPlayerMainSkillRunningEnd(AbilityTask inAbilityTask)
{
    if ((inAbilityTask.GetTID() != (int)PlayerSkillAbilityType.DefaultAttack)
        && playerSkillCoreDataSource != null)
    {
        playerSkillCoreDataSource.skillDamagePercentIncrease_01 = 0.0;
    }
}
```

. 스테이지 종료 시 리셋.

```
protected override void OnStageCleanup()
{
    if (playerSkillCoreDataSource != null)
    {
        playerSkillCoreDataSource.skillDamagePercentIncrease_01 = 0.0;
    }
}
```