프로젝트 소개

Sky Dragon Hunter

(시연 영상 보기) (깃허브 바로가기) (앱스토어 바로가기)

좌측 아이콘 클릭 시, 해당 페이지로 이동합니다.

기간 | 2025.03.19 ~ 2025.05.15 (8주)

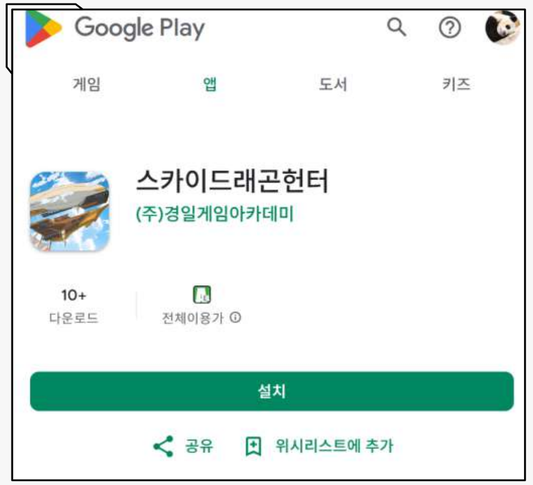
인원 | 6인 (**프로그래머 3인**, 기획 3인)

장르 | 기업협약 / 팀 / 2D / 방치형 / 육성

도구 | Unity(2022.3.55f1)

주요 역할

1. 노션, 스프린트 관리 및 팀원 작업 일정 배분
2. 게임 기반 시스템 및 컨텐츠 개발
3. 버그 이슈 대응
4. Git, Addressable, 서버 관리



01 스킬 시스템

(코드 바로가기)

01 목표

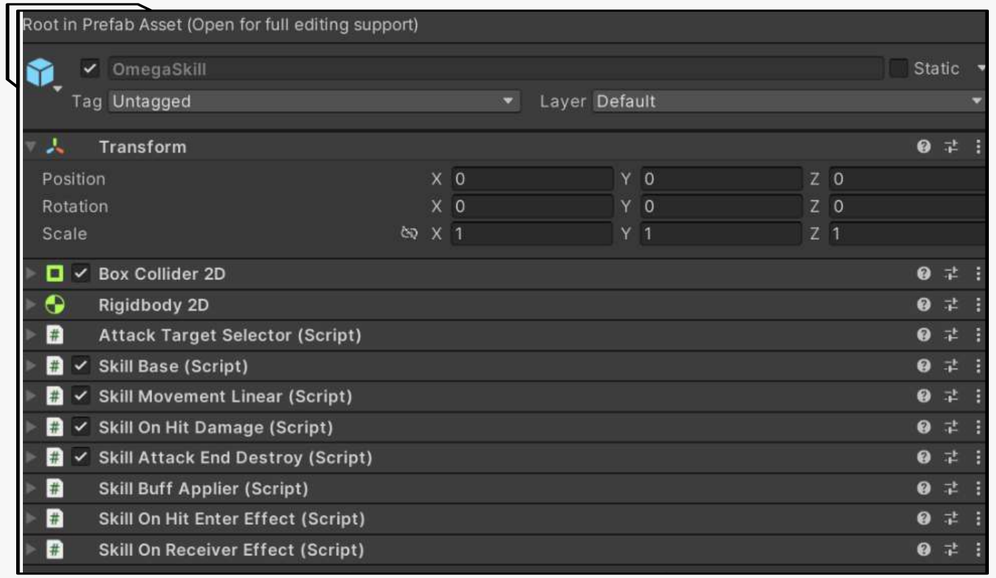
- 20가지 이상의 스킬을 구현
- 컴포넌트 단위로 스킬의 기능을 빠르게 양산할 수 있는 구조로 구현

02 방법

- SkillBase로 스킬 이벤트 중계
- SO로 데이터와 로직 분리
- 이벤트 시스템으로 스킬 특성 컴포넌트 구현

03 결과

- 기능 단위로 스킬 구현 간소화
- 모듈화로 재사용성 향상
- 의존성 감소로 확장성과 유연성 확보



(스킬의 기본 구성)

- SkillBase (Script Component)
  ISkillLifecycleHandler(스킬의 타이밍)이벤트를 기능 컴포넌트들에게 중계

- Attack Target Selector
  - 공격할 수 있는 대상의 tag를 설정하는 컴포넌트

01 스킬의 라이프 사이클 Details

01

구현 흐름

1. 데이터-로직 분리

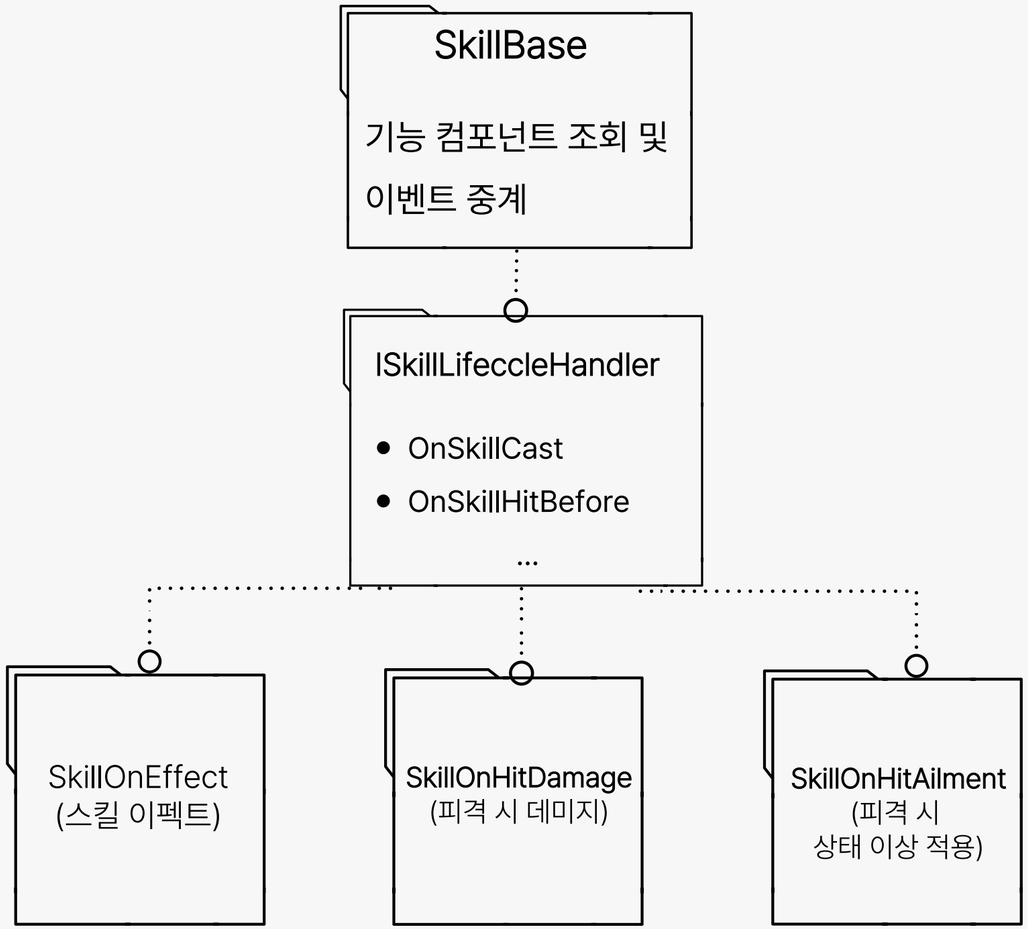
↳ 기획은 데이터 정의, 개발은 실행 로직에 집중 가능

2. 스킬 실행 컴포넌트화

↳ 스킬 재사용성과 유지보수성 향상

3. 기능 단위 모듈화

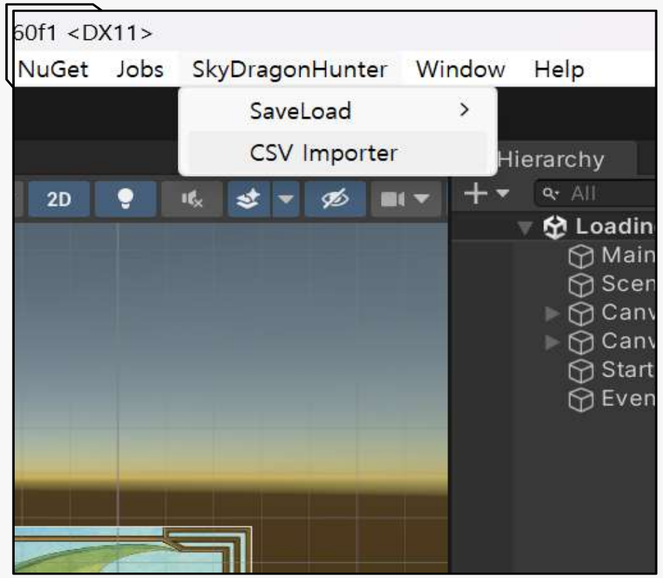
↳ 인터페이스로 효과별 컴포넌트 구현 가능



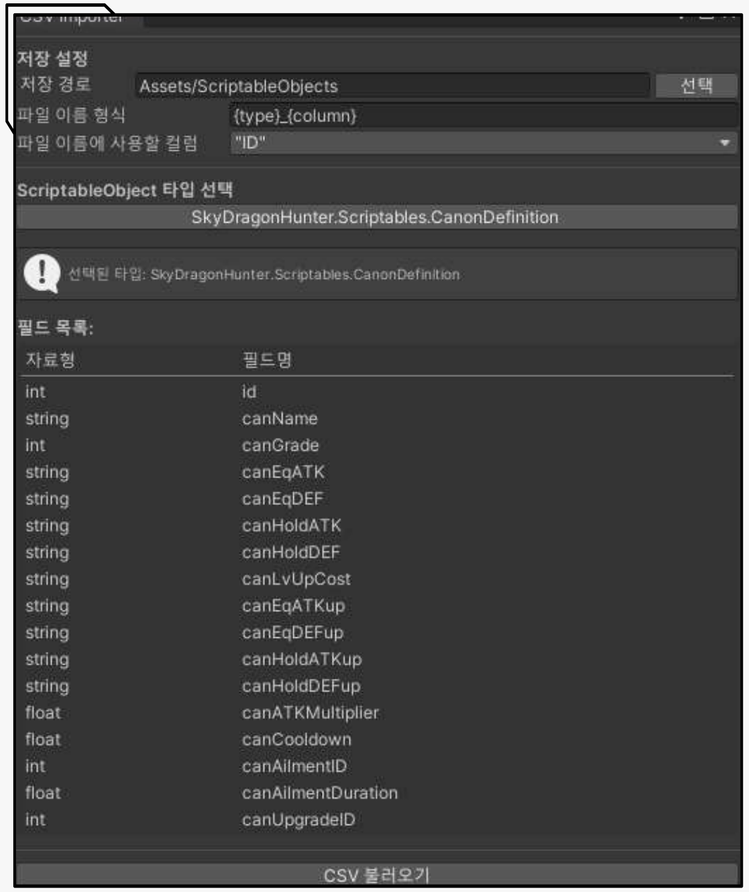
01 Csv Importer

Details

Csv Importer는
Csv Table을 로드한 값들을 원하는 ScriptableObject로 변환하여 생성해주는 기능입니다.



(SkyDragonHunter/CsvImporter)



(SkillDefinition과 CsvImporter의 필드 대조)

02 마스터리 시스템

(코드 바로가기)

01

목표 Csv Table에 따라 마스터리를 생성할 수 있도록 자동화한다.

02

방법
- 그래프 자료구조를 활용하여 마스터리 소켓을 연결
- 너비 우선 탐색을 통해서 마스터리의 레벨 상태와 활성화 여부를 관리

03

결과 기획자가 에디터를 열 필요 없이 테이블만으로 마스터리 제작 가능

	ID	NextNodeIds	SocketID	Level	Position
2	100001	100002/100003/100004	220001	21	0
3	100002	100005	217001	20	0
4	100003	100006	218001	20	1
5	100004	100007	212001	20	2
6	100005	100008	214001	19	0
7	100006	100008	219001	19	1
8	100007	100008	215001	19	2
9	100008	100009/100010/100011	216001	18	0
10	100009	100012	213001	17	0
11	100010	100012/100013	214001	17	1
12	100011	100013	215001	17	2
13	100012	100014	217001	16	0
14	100013	100014	218001	16	1

02 마스터리 시스템 Details

02

구현 흐름

1. Graph로 마스터리 소켓 구성

↳ 노드 간 연결과 위치 기반 탐색을 지원하는 구조 구현

2. CSV 기반 노드 자동 생성 및 배치

↳ CSV 데이터로 노드를 생성하고 방향에 따라 타일 자동 배치

3. 데이터 적용 및 캐싱 최적화

↳ 노드에 테이블 데이터를 적용하고 Dictionary로 빠른 접근 처리

```
public void DirtyMastery()
{
    base.ResetVisitedFlags();
    base.TraverseBFS();
}
```

(너비 우선 탐색을 이용한 Socket 정보 업데이트)

03 보물 시스템

(코드 바로가기)

01

목표 Csv Table에 따라 보물들을 생성하는 구조를 만든다.

02

방법
- RandomMgr를 통해 능력치 랜덤 생성하였습니다.
- 보물 합성 및 장착 로직과 보물 데이터 정의 분리해서 구현하였습니다.

03

결과
- 테이블에 맞게 보물의 능력들이 자동 생성되도록 구현하였습니다.

1	ID	Name	Desc	IconName
2	110000001	히아신스	보물	Hyacinth
3	110000002	닌자의 표창	보물	NinjaThrowingStar
4	110000003	황금 실타래	보물	GoldenThread
5	110000004	행운의 세잎클로버	보물	LuckyThreeLeafClove
6	110000005	박하맛 막대사탕	보물	PeppermintCandyCa
7	110000006	루비 반지	보물	RubyRing
8	110000007	의문의 선물상자	보물	MysteriousGiftBox
9	110000008	마녀의 마법봉	보물	WitchMagicWand
10	110000009	요정의 완드	보물	FairyWand
11	110000010	기초 마법 스크롤	보물	BasicMagicScroll
12	110000011	순수한 얼음 결정	보물	PureIceCrystal

03 보물 시스템

Details

02

구현 흐름

1. **Csv Table에 맞게 보물 데이터 클래스 생성합니다.**

↳ 데이터 구조와 코드의 결합도 감소

2. 생성된 보물 데이터에 맞게 등급 부여 및 랜덤 고정 스탯 계산합니다.

↳ RNG 일관성 확보로 디버깅 용이

3. 고정 스탯이 정해지면 추가 스탯 계산을 계산합니다.

↳ 조건 기반 확장 시 유연한 대응 가능

```
public static T RandomWithWeights<T>(params (T candidate, float weight)[] param)
{
    float total = 0f;
    for (int i = 0; i < param.Length; i++)
    {
        if (param[i].weight < 0f)
        {
            Debug.LogError(s_ArgNegativeError);
            throw new ArgumentException(s_ArgNegativeError);
        }
        total += param[i].weight;
    }

    float rand = UnityEngine.Random.Range(0f, total);
    float sum = 0f;

    for (int i = 0; i < param.Length; i++)
    {
        sum += param[i].weight;
        if (rand ⩽ sum)
            return param[i].candidate;
    }

    return param[^1].candidate;
}
```

(가중치를 이용하여 랜덤 생성을 하기 위한 랜덤 매니저의 Static 메서드)

04 아이템 인벤토리

(코드 바로가기)

01

목표 아이템들을 관리하는 UI 인벤토리 구현하기

02

방법
- Unity UI 기능들을 활용하여 레이아웃을 구성하고 Sprite 배치
- Content Layout Group과 스크립트를 이용하여 동적으로 구성

03

결과 UI 자동 정렬과 아이템 동적 생성이 가능합니다.

(아이템 인벤토리)

03 작업 스프린트 관리

(노션 바로가기)

01

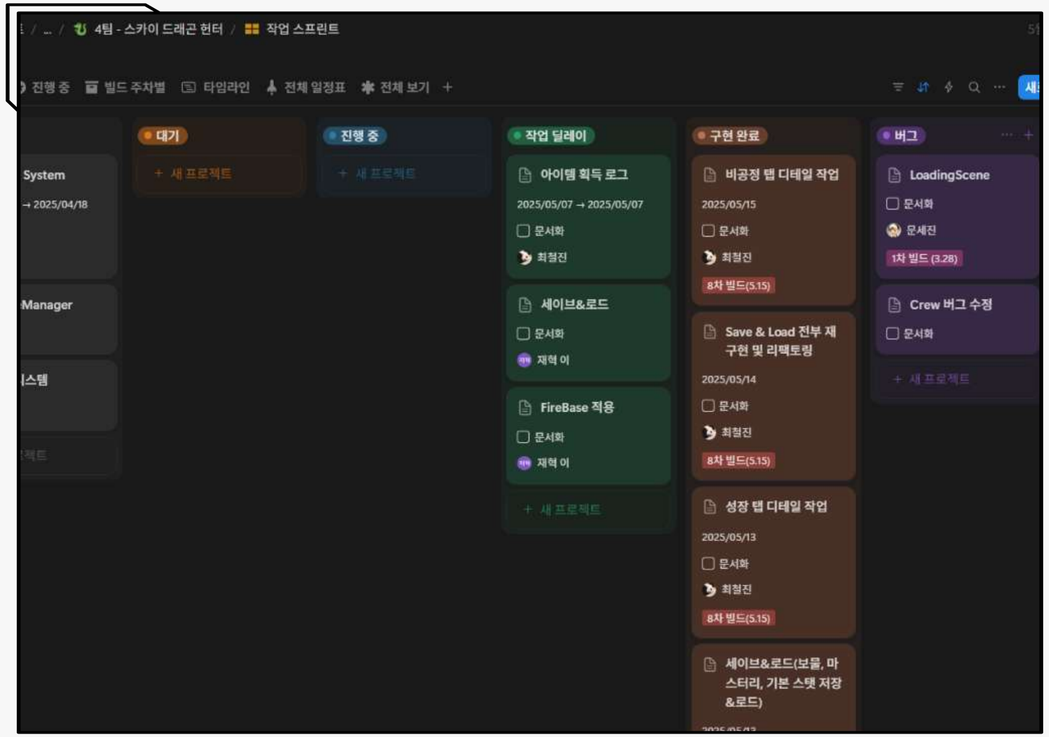
목표 팀원들의 작업 분배 및 진행 상황 파악을 위해 스프린트 구축

02

방법 Notion에 개발 진척에 대한 보고 사항, 진행 사항 관리하였습니다.

03

결과 문서화하여 현재 팀이 진행되는 과정을 보다 쉽게 파악할 수 있었습니다.



(팀 프로젝트 관리 페이지)

05 그외 구현 사항

기본 성장 시스템

계정, 캐릭터, 몬스터들을 나타내는 공통 능력치와 능력치를 강화할 수 있는 시스템을 구현하였습니다.

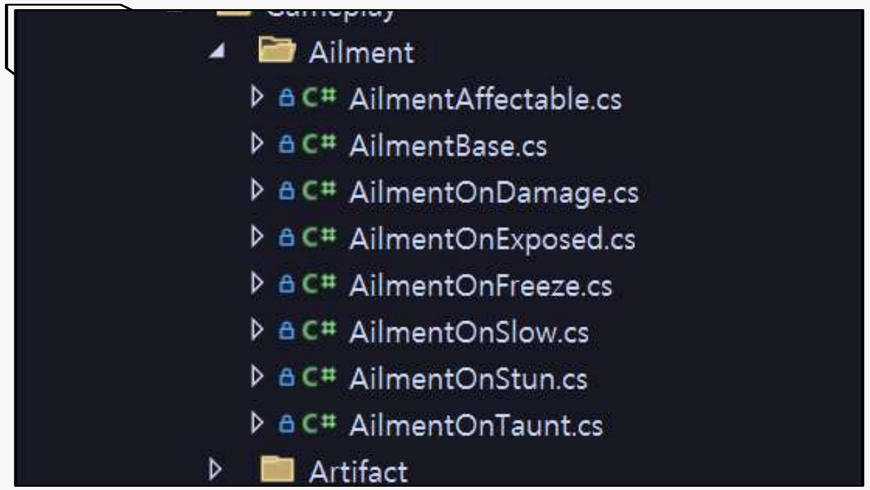
(코드 바로가기)

뽑기 시스템

csv 확률 테이블의 랜덤 값을 기반으로 결과가 뽑히도록 구현했습니다.

(코드 바로가기)

05 그외 구현 사항



상태 이상

대상에게 화상, 빙결, 둔화, 기절 등의 상태 이상을 적용할 수 있는 기능 구현하였습니다.

(코드 바로가기)

버프 & 디버프

대상에게 받는 피해 증가, 쿨다운 감소, 공격 속도 증가 등 버프 & 디버프를 줄 수 있는 구조 구현하였습니다.

(코드 바로가기)

05 그외 구현 사항

대포, 몬스터 무기 시스템

비공선에 장착할 수 있는 대포 구조 및 컨텐츠를 구현, 관련 UI들을 구현하였습니다.

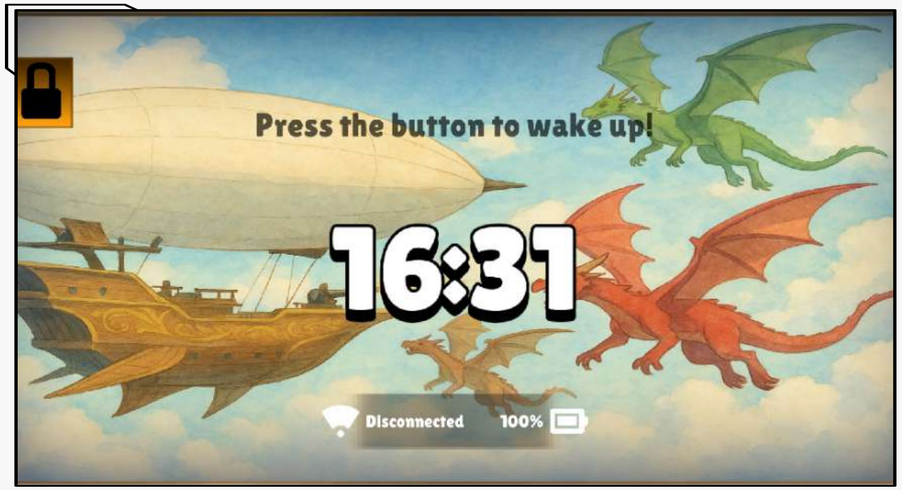
(코드 바로가기)

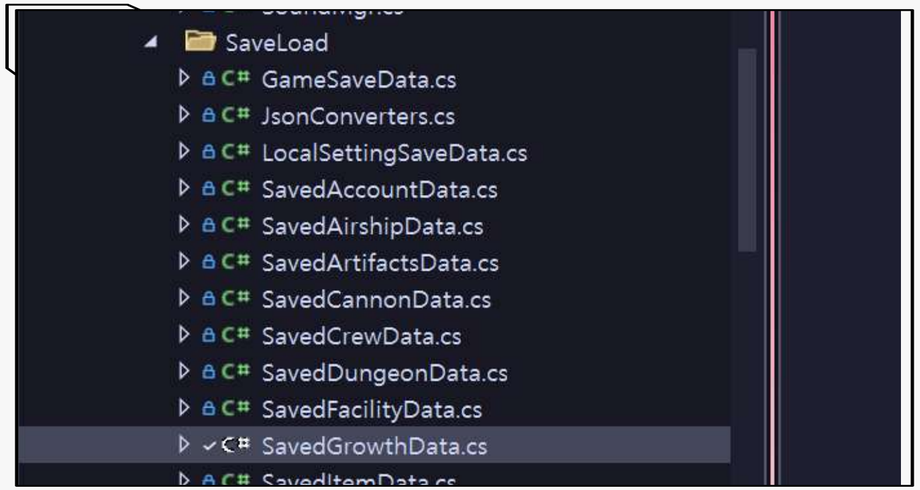
수리공 구현

비공선을 수리하거나 방어막, 무적등을 부여해줄 수 있는 수리공 시스템을 구현하였습니다.

(코드 바로가기)

05 그외 구현 사항





절전 모드

기기의 전력을 최소화하면서 게임을 유지시키기 위한 기능을 구현하였습니다.

(코드 바로가기)

세이브 & 로드 리팩토링

기존에 만들어진 세이브 & 로드의 버그 이슈로 인해 관련 기능들 리팩토링 진행하였습니다.

(코드 바로가기)

EOF END

감사합니다.

이름: 최철진
이메일: chlcjfwls1@naver.com
연락처: 010-2892-0417