Survival

작성자 : 김재완

목차

[개발 컨셉]

1. 개요
2. 구현 목록

[게임 구성]

1. 게임 흐름도
2. 각 화면 설명
3. 시스템 설명

개발 컨셉

1. 개요

항목	설명
제목	Surival(서바이벌)
장르	2D 1인용 캐주얼, 핵 엔 슬래시 게임
디바이스 / 플랫폼	- 모바일 (안드로이드)
기획 의도	1. 간단한 포트폴리용 프로젝트 2. 2D게임 제작해보기
특징	1. 라운드 마다 나오는 몬스토를 잡으며 재화를 얻을 수 있다. 2. 게임 진행을 통해 스킬 과 능력치를 올려 여러 조합을 만들 수 있다. 3. 현재 1-5 까지 진행가능
한 줄 소개	매 라운드마다 몬스터를 잡으며 스킬과 능력치를 강화하여 보스를 쓰러뜨리자

개발 컨셉

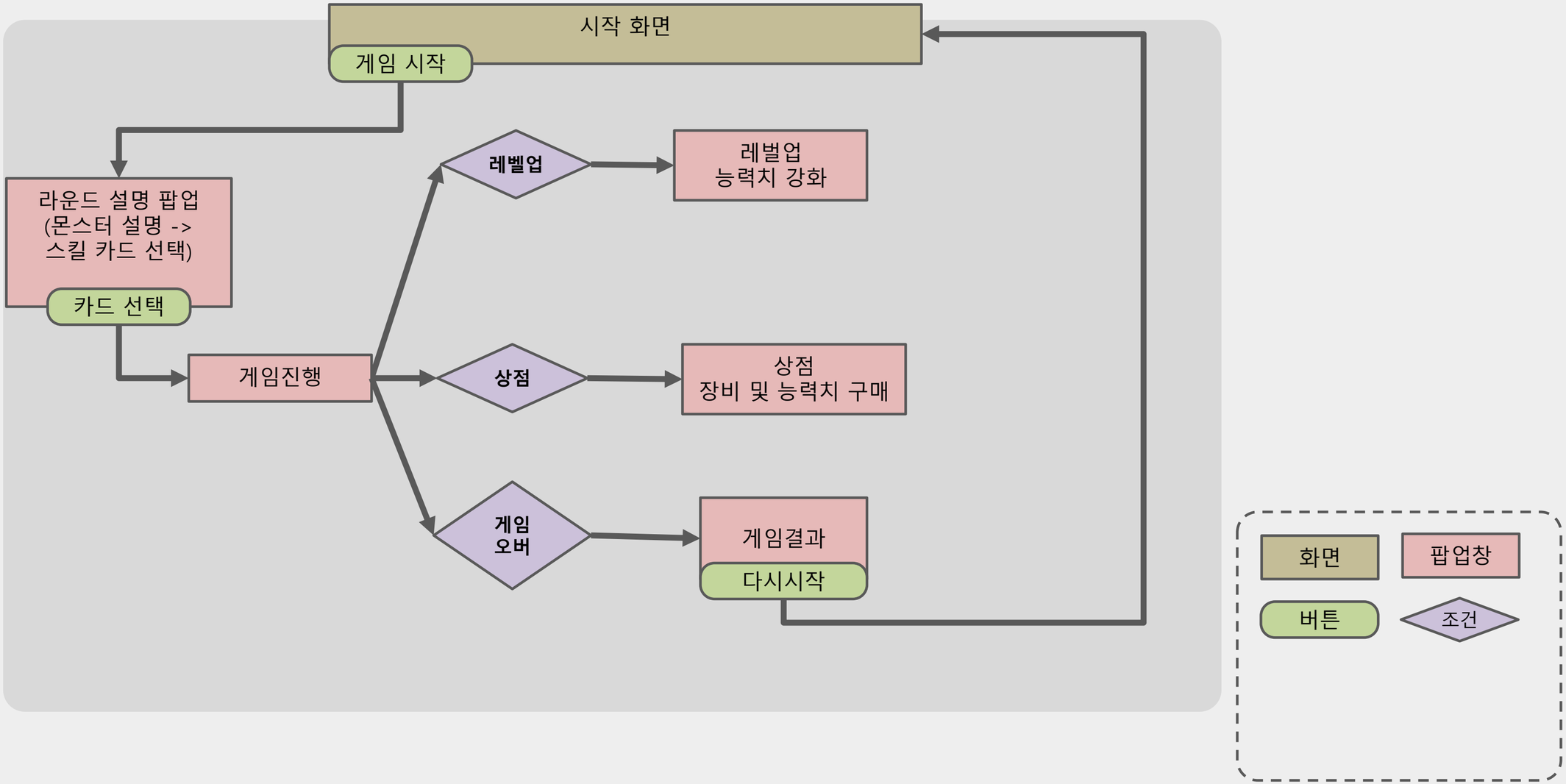
2. 구현 목록 – 화면/시스템

필수 구현 화면	설명
-	이번 프로젝트는 씬 1개로 작업하여 상황에 맞는 UI를 켜준다.
게임 시작 화면	간단한 게임 설명과 게임 시작버튼
라운드 시작 화면	등장 몬스터 와 스킬 선택
레벨 업 화면	능력치 강화 카드 선택
인 게임 화면	게임에 필요한 UI와 조이스틱 등

필수 구현 시스템	설명
기타	오브젝트풀, 카메라, 조이스틱 이동
몬스터 스폰 시스템	몬스터를 스폰하게 해주는 시스템
스킬 시스템	스킬 시스템(총 6개 스킬 구현)
상점 시스템	준비된 물건을 팔 수있도록

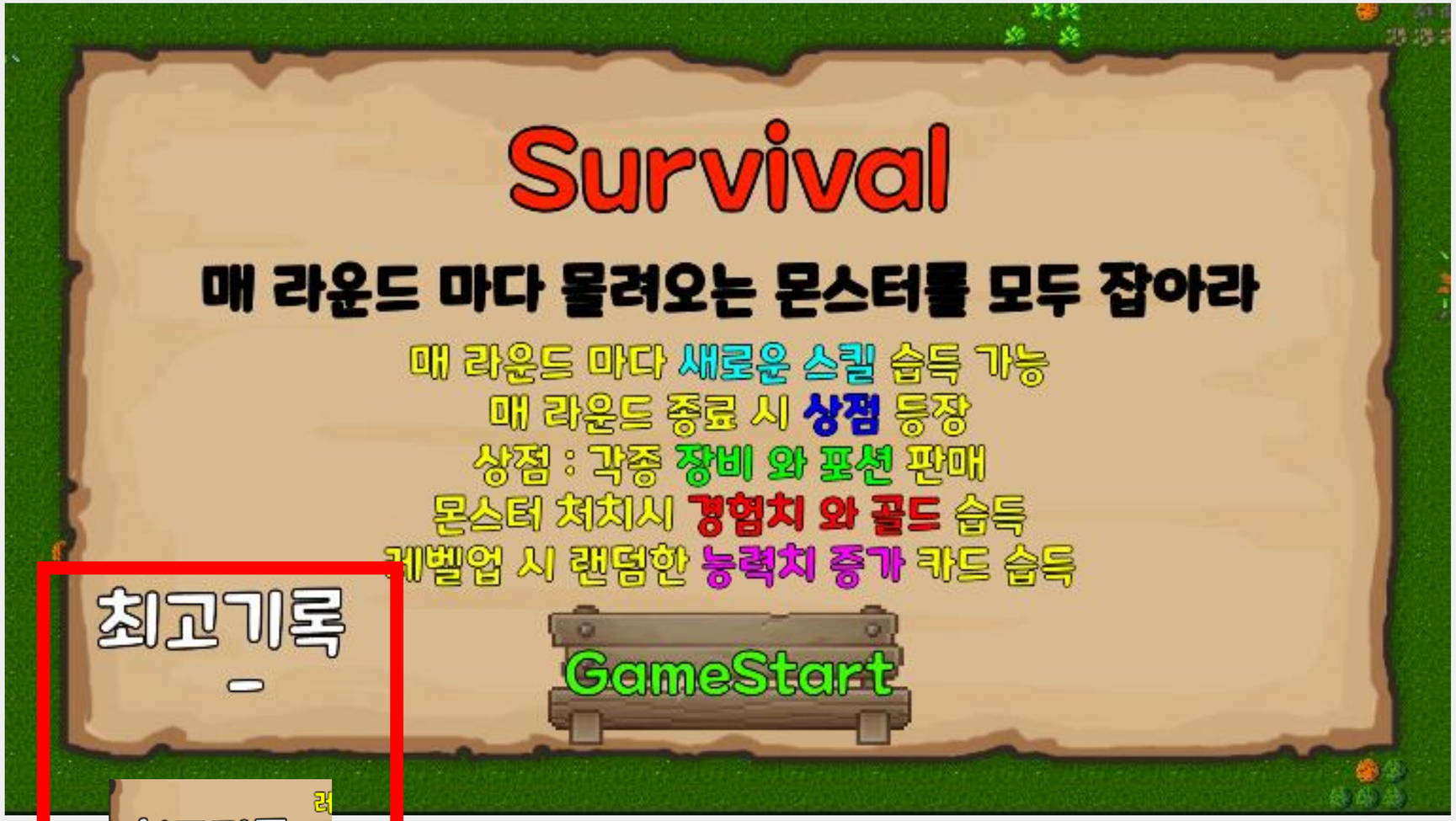
게임 구성

1. 게임 흐름도


시작 화면
게임 시작
라운드 설명 팝업
(몬스터 설명 ->
스킬 카드 선택)
카드 선택
게임진행
레벨업
레벌업
능력치 강화
상점
상점
장비 및 능력치 구매
게임
오버
게임결과
다시시작
화면
팝업창
버튼
조건

게임 구성

2. 각 화면 설명 – 게임 시작



1. 최고 기록 UI
   - 최고 기록 존재 시
   - 리셋 버튼 활성화
2. 게임 시작 버튼
3. 간단한 게임 설명

2. 각 화면 설명 – 게임 시작

1) GameMgr.cs 일부 처음 시작시 셋팅

```
Unity 메시지 | 참조 0개
private void Start()
{
    //최고 기록 정보 가져오기
    BestStage = PlayerPrefs.GetInt("BestStage", 0);

    //기록 점수 텍스트 셋팅하기
    if (BestStage == 0)
    {
        bestScoreTxt.text = "-";
        resetScoreBtn.gameObject.SetActive(false);
    }
    else
        bestScoreTxt.text = (BestStage / 5 + 1) + "-" + (BestStage % 5 + 1);

    //오디오클립을 이름별로 찾을수 있게 딕셔너리에 저장
    for (int i = 0; i < audioClips.Length; i++)
        Dic_AudioClip.Add(audioClips[i].name, audioClips[i]);

    gameStartBtn.onClick.AddListener(GameStart);
    eqInfoBoxBtn.onClick.AddListener(OffEqItemInfoBox);
    nextBtn.onClick.AddListener(RoundStart);
    resetScoreBtn.onClick.AddListener(ResetScore);

    //캐릭터 레벨업시 호출될 함수 등록
    hero.LevelUP_Event += OnLevelUpPanel;
}
```

2) 게임 시작 버튼

```
참조 1개
public void GameStart() //본 게임시작
{
    LobbyPanel.SetActive(false); //로비 패널 off
    hero.SetEqItem(startWeapon);  //시작 무기 장착
    RoundStart();   //라운드 시작
}

참조 2개
public void RoundStart()//라운드 시작
{
    StageMonsterInfoShow(); //스테이지정보 및 등장몬스터 소개

    StartCoroutine(MonsterSpawner());   //몬스터 스폰 코루틴 실행
    InGameUIs.SetActive(true); //InGameUIs On
    nextBtn.gameObject.SetActive(false); //다음라운드로 가는 버튼 off
    ShopMgr.Inst.Shop = false;              //상점 off
}
```

3) 최고 기록 리셋

```
참조 1개
private void ResetScore() //기록 초기화
{
    PlayerPrefs.DeleteAll(); //저장된 기록 지우기

    bestScoreTxt.text = "-";
    resetScoreBtn.gameObject.SetActive(false);
}
```

2. 각 화면 설명 – 라운드 시작

매 라운드 시 등장

- **등장 몬스터 소개**

1. 등장 몬스터의 능력치와 종류
2. 마리 등장 수

스킬 카드 선택

1. 새로운 스킬을 얻거나 스킬 레벨 업 가능
2. 선택 후 본 라운드 시작

게임 구성

2. 각 화면 설명 – 라운드 시작

1) StageStartCtrl.cs 라운드 시작시 나타나는 UI창

선언 변수 와 셋팅

```
public class StageStartCtrl : MonoBehaviour
{//스테이지 시작할시 나타나는
    [Header("UI")]
    public GameObject monsterCardPanel; //등장몬스터를 보여주는
    public GameObject skillCardPanel;       //스킬 선택하는

    public TextMeshProUGUI monsterCountTxt; //등장몬스터 마리수txt
    public Button nextBtn;                          //다음으로 넘어가는
    [Header("Card_UI")]
    public Card[] monsterCards; //몬스터카드
    public Card bossCard;           //보스카드
    public SkillCard[] Skillcards;  //스킬카드

    StageData stageData;            //현재 스테이지 정보
    Skill[] skills;                         //스킬목록

    Unity 메시지 | 참조 0개
    private void Awake()
    {//스킬 목록 가져오기
        skills = GameMgr.Inst.skills;
    }

    Unity 메시지 | 참조 0개
    private void Start()
    {
        nextBtn.onClick.AddListener(NextBtn);

        //스킬 셋팅해놓기
        for (int i = 0; i < Skillcards.Length; i++)
            Skillcards[i].skill = skills[i];
    }
```

2) 창 오픈시 등장 몬스터 공개

```
private void OnEnable()
{//오픈시
    //패널정리
    monsterCardPanel.gameObject.SetActive(true);
    skillCardPanel.gameObject.SetActive(false);
    //현재 스테이지 정보 가져오기
    stageData = GameMgr.Inst.StageData;
    //버튼 활성화
    nextBtn.gameObject.SetActive(true);
    //등장몬스터 카드 셋팅하기
    for (int i = 0; i < stageData.monsterDatas.Length; i++)
    {
        monsterCards[i].SetCard(stageData.monsterDatas[i].GetCard());
        monsterCards[i].gameObject.SetActive(true);
    }
    //만약 보스몬스터가 존재하면 보스카드 셋팅
    if (GameMgr.Inst.StageData.bossMonsterPrefab)
    {
        bossCard.SetCard(GameMgr.Inst.StageData.bossMonsterPrefab.GetComponent<SetCard>().GetCard());
        bossCard.gameObject.SetActive(true);
    }
    //등장 마리수txt 셋팅
    monsterCountTxt.text = stageData.monsterCount + "마리";
}
```

3) 스킬 카드 공개

```
참조 1개
void NextBtn()
{//다음 버튼 몬스터정보 -> 스킬 카드
    monsterCardPanel.gameObject.SetActive(false);
    skillCardPanel.gameObject.SetActive(true);
    //레벨업이 가능한지 체크후
    for (int i = 0; i < Skillcards.Length; i++)
    {
        Skillcards[i].gameObject.SetActive(true);

        if (skills[i].skill_Lv ==skills[i].skill_MaxLv)//최대 레벨이면
            Skillcards[i].gameObject.SetActive(false);  //비활성화
        else
            Skillcards[i].SetCard(skills[i].GetCard());
    }
    //다음버튼은 필요없으니 비활성화
    nextBtn.gameObject.SetActive(false);
}
```

게임 구성

2. 각 화면 설명 – 라운드 시작

1) Card .cs

```cs
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using CardHelp; //카드 관련 필요 인터페이스 사용을 위해

Unity 스크립트|참조 6개
public class Card : MonoBehaviour
{
    public Image image; //카드 이미지
    public TextMeshProUGUI info;    //카드 설명란

    참조 4개
    public void SetCard(CardData card) //카드기본 셋팅
    {
        image.sprite = card.img;
        info.text = card.info;
    }

}
```

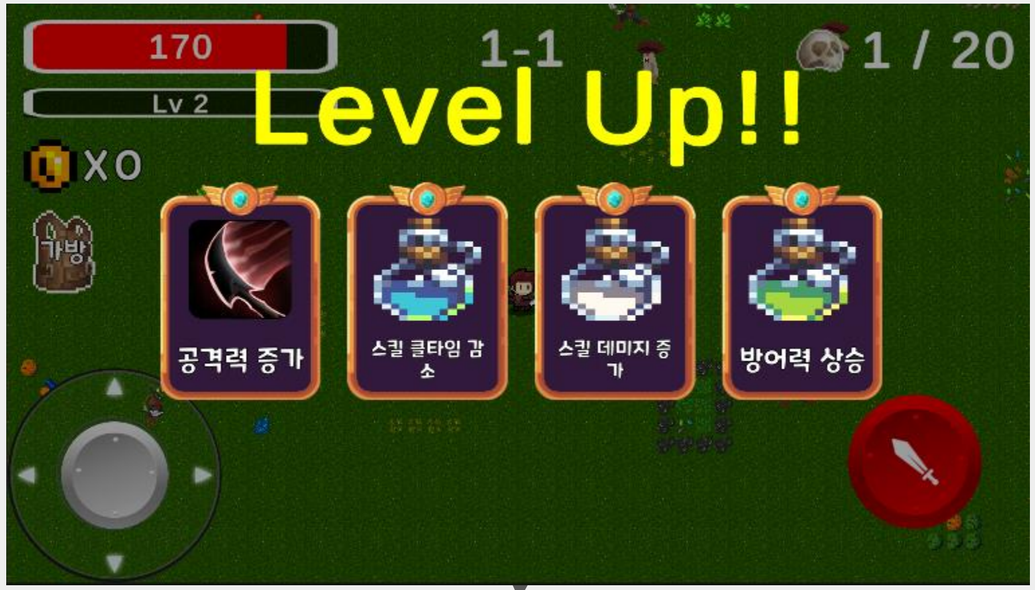
2) 카드 작업을 도와주는 인터페이스

```cs
namespace CardHelp //카드로 보여줄수있게 해주는
{
    참조 11개
    public struct CardData //카드 데이터
    {
        public Sprite img;
        public string info;
    }

    참조 11개
    public interface ICardLvUp //레벨업이 가능한 카드들
    {
        참조 4개
        public bool LevelPossible();
        참조 4개
        public void LevelUp();
        참조 4개
        public CardData GetCard();
    }

    참조 3개
    public interface SetCard //단순한 카드 정보를 보여줄수 있게하는
    {
        참조 4개
        public CardData GetCard();
    }
}
```

3) 스킬 카드

```cs
Unity 스크립트|참조 1개
public class SkillCard : Card
{
    //스킬 카드 매 라운드 시작시
    //선택하는 카드
    public Skill skill; //등록된 스킬
    public Button skillBtn; //레벨업 스킬 버튼
    Unity 메시지|참조 0개
    private void Start()
    {
        skillBtn.onClick.AddListener(GetSkill);//버튼등록
    }

    참조 1개
    void GetSkill()
    {
        skill.LevelUp();//스킬 레벨업
        GameMgr.Inst.StageGameStart();//스테이지 시작
    }
}
```

2. 각 화면 설명 – 레벨 업



레벨업 시 등장

- **스킬 및 능력치 강화**

1. 가지고 있는 스킬 중 3가지 까지 랜덤 스킬 강화 카드 등장
2. 나머지 카드는 랜덤 능력 카드 등장
3. 레벨업시 시간이 멈추고 선택 후 다시 시작
4. “Level Up!!” 이라는 타이틀의 색깔은 실시간으로 색깔이 변경되게 하였다.

게임 구성

2. 각 화면 설명 – 레벨 업

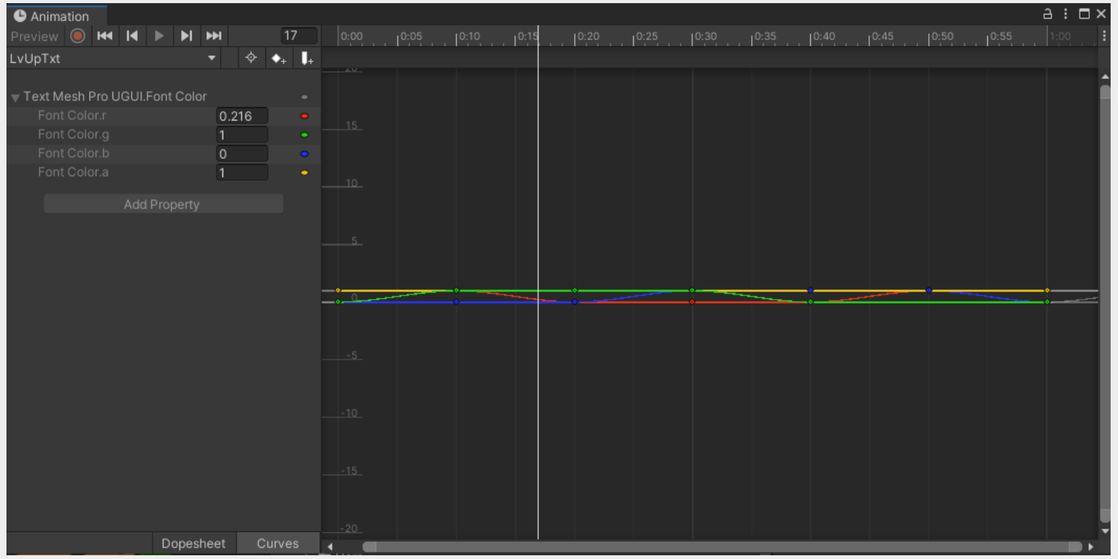
1) LevelUpPanel.cs 레벨업 시 나타나는 UI 창

```
Unity 스크립트 | 참조 2개
public class LevelUpPanel : MonoBehaviour
{//레벨업시 나타날 UI Panel
    public static LevelUpPanel inst;

    [Header("LvUpLable")]
    public TextMeshProUGUI lable;
    public Animation lableAnimation;

    [Header("LvUpObj")]
    public GameObject[] lvUpObjs; //레벨업이 가능한 스킬목록
    public Ability[] abilities;     //레벨업이 가능한 능력치목록

    List<ICardLvUp> skillCardList = new List<ICardLvUp>(); //스킬카드 목록
    List<ICardLvUp> abilityCardList = new List<ICardLvUp>(); //능력치카드 목록
    List<ICardLvUp> skillLvUpAbleList = new List<ICardLvUp>();//최종 레벨업이 가능한 목록

     [Header("LvUpCard")]
    public LvUpCard[] lvUpCard; //레벨업을 시켜주는 카드UI

    //Time 계산을 위해
    float realTimeDalta = 0.0f;
    float animationTime = 0.0f;

    Unity 메시지 | 참조 0개
    private void Awake()
    {
        inst = this;
        //목록 채워주기
        for (int i = 0; i < lvUpObjs.Length; i++)
            skillCardList.Add(lvUpObjs[i].GetComponent<ICardLvUp>());

        for (int i = 0; i < abilities.Length; i++)
            abilityCardList.Add(abilities[i]);
    }

    Unity 메시지 | 참조 0개
    private void Update()
    {
        //이 오브젝트가 켜져있을때는  Time.timeScale = 0이기 때문에
        //시간계산을 해주어 애니메이션을 돌려준다.
        float curTime = Time.realtimeSinceStartup;
        float deltaTime = curTime - realTimeDalta;
        realTimeDalta = curTime;

        animationTime += deltaTime;

        LableTxt_Update();
    }
```

2) TimeScale 이 0이라서 직접 시간을 구해 애니메이션을 돌려준다.

```
void LableTxt_Update()
{
    //직접 시간을 계산하여 애니메이션을 돌려준다.
    AnimationState a = lableAnimation["LvUpTxt"];
    a.normalizedTime = animationTime % a.length;
}
```

애니메이션을 만들어 색깔을 바꾸게 하였다.



2. 각 화면 설명 – 레벨 업

3) 무작위 스킬 및 능력치 카드 셋팅

```
참조 1개
void CheckLevelPossible() //레벨업이 가능한지 체크
{
    skillLvUpAbleList.Clear(); //리스트 초기화
    for (int i = 0; i < skillCardList.Count; i++)
    {
        if (skillCardList[i].LevelPossible() == true) //레벨업이 가능한지 체크 후
            skillLvUpAbleList.Add(skillCardList[i]);      //리스트에 넣어준다.
    }
}
void SetLvUpCard() //레벨업 카드 셋팅
{
    int idx = 0; //4개의 카드선택 가능
    List<int> random = new List<int>(); //무작위 선정을 위해
    if(skillLvUpAbleList.Count < 3) //스킬이 3개 미만일경우
    {
        //가지고 있는 스킬 모두 선택가능
        for (int i = 0; i < skillLvUpAbleList.Count; i++)
            lvUpCard[i].SetCard(skillLvUpAbleList[i]);

        idx += skillLvUpAbleList.Count;
    }
    else //스킬이 3개 이상일경우
    {
        while (random.Count <= 2) //2개까지 랜덤으로 선정
        {
            int a = Random.Range(0, skillLvUpAbleList.Count);
            if (random.Contains(a))
                continue;
            else
                random.Add(a);
        }
        //선정된 순서에 맞는 스킬을 셋팅
        for (int i = 0; i < random.Count; i++)
        {
            lvUpCard[i].SetCard(skillLvUpAbleList[random[i]]);
            idx++;
        }
    }
    //스킬선정 후 능력치 선정
    random.Clear();
    while (random.Count < 4 - idx) //중복되지 않게 랜덤
    {
        int a = Random.Range(0, abilityCardList.Count);
        if (random.Contains(a))
            continue;
        else
            random.Add(a);
    }

    for (int i = 0; i < random.Count; i++)  //선정된 순서에 맞는 능력치을 셋팅
        lvUpCard[idx + i].SetCard(abilityCardList[random[i]]);

}
```

4) 레벨업을 시켜주는 카드 cs

```
using UnityEngine;
using UnityEngine.EventSystems;
using CardHelp;

Unity 스크립트 | 참조 1개
public class LvUpCard : Card, IPointerClickHandler
{
    public ICardLvUp cardFunc; //카드 레벨업시 필요한 인터페이스

    참조 3개
    public void SetCard(ICardLvUp cardFunc) //카드를 셋팅하는 함수
    {//오버라이딩을 함으로써 추가필요한 정보를 가져옴
        this.cardFunc = cardFunc;
        //기본 카드 셋팅
        base.SetCard(cardFunc.GetCard());
        //카드 오픈(레벨업이 가능한 상태면)
        if (cardFunc.LevelPossible())
            this.gameObject.SetActive(true);
    }

    참조 3개
    public void OnPointerClick(PointerEventData eventData)
    {//카드를 눌렀을때 반응하는
        cardFunc.LevelUp(); //레벨업
        LevelUpPanel.inst.OffPanel(); //레벨업 패널 off
    }
}
```

게임 구성

2. 각 화면 설명 – 인게임

캐릭터 HP, EXP, Coin

현재 라운드

현재 킬수 / 목표 킬수

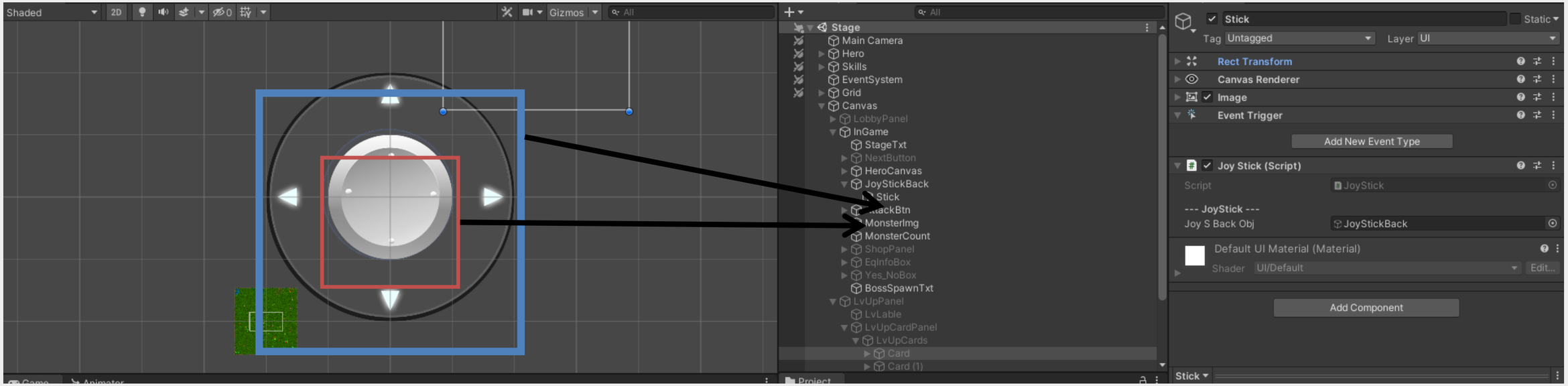
캐릭터 정보

조이 스틱(이동 키)

공격 버튼

3. 시스템 설명 - 이동구현

1) 캐릭터 이동을 위한 조이스틱 구현



뒤에 방향을 표시해주는 그림과

중앙 드래그를 해주는 스틱 이미지로 구성

드래그 판정을 판단을 하기위해 EventTrigger 컴포넌트

추가 와 JoyStick.cs 를 추가

3. 시스템 설명

2) JoyStick.cs

```cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;

Unity 스크립트 | 참조 0개
public class JoyStick : MonoBehaviour , IDragHandler, IPointerUpHandler
{
    //IDragHandler 마우스 드래그
    //IPointerUpHandler 마우스 클릭 후 땔때
    HeroCtrl heroCtrl;
    [Header("--- JoyStick ---")]
    public GameObject joySBackObj = null;
    float radius = 0.0f;
    Vector2 orignPos = Vector3.zero;
    Vector2 axis = Vector3.zero;
    Vector2 jsCacVec = Vector3.zero;
    float jsCacDist = 0.0f;

    Unity 메시지 | 참조 0개
    private void Start()
    {
        heroCtrl = GameMgr.Inst.hero; //캐릭터연결
        Vector3[] v = new Vector3[4];
        joySBackObj.GetComponent<RectTransform>().GetWorldCorners(v);
        //[0]:좌측하단 [1]:좌측상단 [2]:우측상단 [3]:우측하단
        //v[0] 좌측하단이 0, 0 좌표인 스크린 좌표(Screen.width, Screen.height)를 기준으로
        radius = v[2].y - v[0].y;
        radius = radius / 3.0f;
        //중앙 위치
        orignPos = transform.position;
    }
```

처음 이미지의 크기를 계산하여 최대 이동 거리를 계산 해놓는다.

드래그

손을 때면

```cs
public void OnDrag(PointerEventData eventData)
{ //IDragHandler 마우스 드래그

    jsCacVec = eventData.position - orignPos;
    jsCacDist = jsCacVec.magnitude; // 얼마나 갔는지
    axis = jsCacVec.normalized; //방향확인

    //조이스틱 백그라운드를 벗어나지 못하게 막는 부분
    if (radius < jsCacDist)
        transform.position = orignPos + axis * radius;
    else
        transform.position = orignPos + axis * jsCacDist;

    //얼마나 스틱을 움직였나에 따라 달리기 가능
    bool sprint = false;
    if (radius * 0.7 < jsCacDist)
        sprint = true;

    //캐릭터 이동 처리
    if (heroCtrl != null)
        heroCtrl.SetJoyStickMv(axis, sprint);
}

참조 0개
public void OnPointerUp(PointerEventData eventData)
{
    axis = Vector3.zero; //초기화
    jsCacDist = 0.0f;
    transform.position = orignPos; //원래 위치로

    //캐릭터 정지 처리
    if (heroCtrl != null)
        heroCtrl.SetJoyStickMv(axis);
}
```

다시 가운데 위치로 이동 후 캐릭터에게 값을 전달하여 정지

방향 과 길이를 체크하여 캐릭터 클래스의 실제로 움직이는 함수에 전달

HeroCtrl.cs

```cs
public void SetJoyStickMv(Vector3 dir, bool sprint = false) //조이스틱을 이용한 이동함
{
    if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack"))
        return; //만약 공격중이면

    mvDir = dir; //이동 방향 적용
    //애니메이터 적용
    if (mvDir.Equals(Vector3.zero)) animator.SetBool("move", false);
    else animator.SetBool("move", true);
    //달리기 적용
    if (sprint)
    {
        animator.speed = 1.2f;
        speed = 4.0f;
    }
    else
    {
        animator.speed = 1.0f;
        speed = 2.0f;
    }

    //이미지 좌우 변경
    if (mvDir.x < 0)
        heroModelTr.localScale = originScale;
    else if(mvDir.x > 0)
    {
        Vector3 temp = originScale;
        temp.x *= -1;
        heroModelTr.localScale = temp;
    }
}
```

게임 구성

3. 시스템 설명 - 오브젝트 풀링

1) 자주 쓰는 오브젝트를 관리하기 위한

ObjectPool_Stack.cs

```
Unity 스크립트|참조 5개
public class ObjectPool_Stack<T> : MonoBehaviour where T : MonoBehaviour
{
    private Stack<T> poolList = new Stack<T>(); //배열
    public GameObject classObj; //프리팹
    [SerializeField]private int maxPoolSize = 10;   //생성갯수

    //객체 생성
    Unity 메시지|참조 0개
    private void Start()
    {//오브젝트 생성후 스택에 추가하기
        for (int i = 0; i < maxPoolSize; i++)
        {
            GameObject obj = Instantiate(classObj, transform);
            poolList.Push(obj.GetComponent<T>());
            obj.SetActive(false);
        }
    }

    참조 6개
    public T GetObj() //사용가능한 오브젝트 순서대로 리턴
    {
        if (poolList.Count <= 1) //여분 생성
        {
            GameObject obj = Instantiate(classObj, transform);
            poolList.Push(obj.GetComponent<T>());
            obj.SetActive(false);
        }
        return poolList.Pop();
    }

    참조 5개
    public void ReturnObj(T Obj)//다시 오브젝트 풀용 스택에 넣기
    {
        if (poolList.Count >= maxPoolSize)
            Destroy(Obj.gameObject);
        else
            poolList.Push(Obj);
    }


}
```

필요로 할때 오브젝트를 가지고 오고

다시 오브젝트를 반환할수 있도록 구현

부족한 오브젝트는 그때 그때 만들수 있도록 하였고

만약 다시 리턴 받을때 이미 많은 오브젝트 존재시

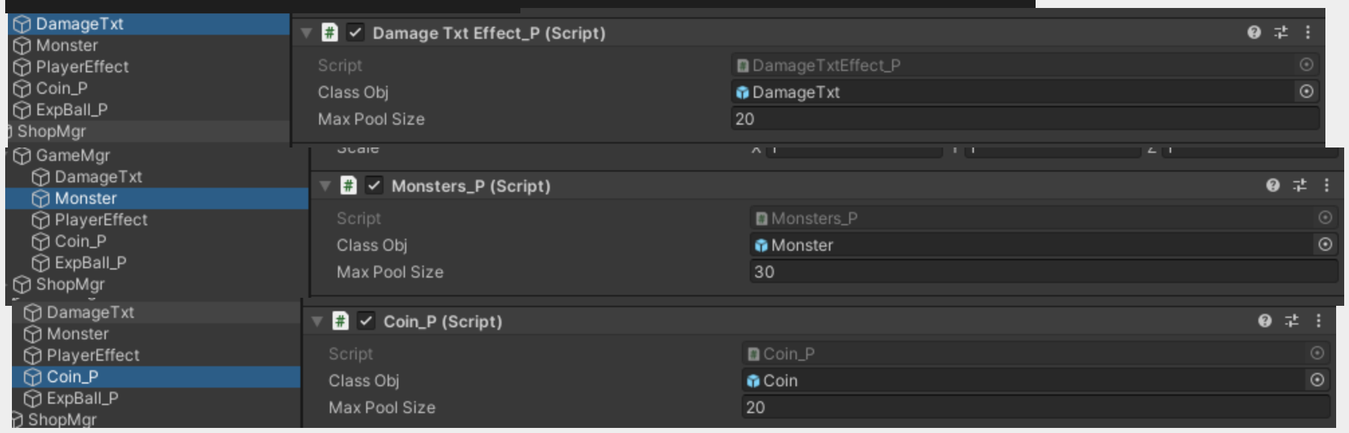
그 오브젝트는 삭제

필요한 오브젝트 마다 ObjectPool_Stack.cs를 사용하여

새로운 스크립을 만든다.

```
Unity 스크립트(자산 참조 1개)|참조 1개
public class Coin_P : ObjectPool_Stack<CoinObj>
{

}
```

```
Unity 스크립트|- 참조
public class Monsters_P : ObjectPool_Stack<Monster>
{

}
```

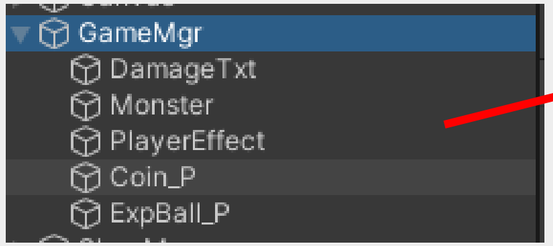


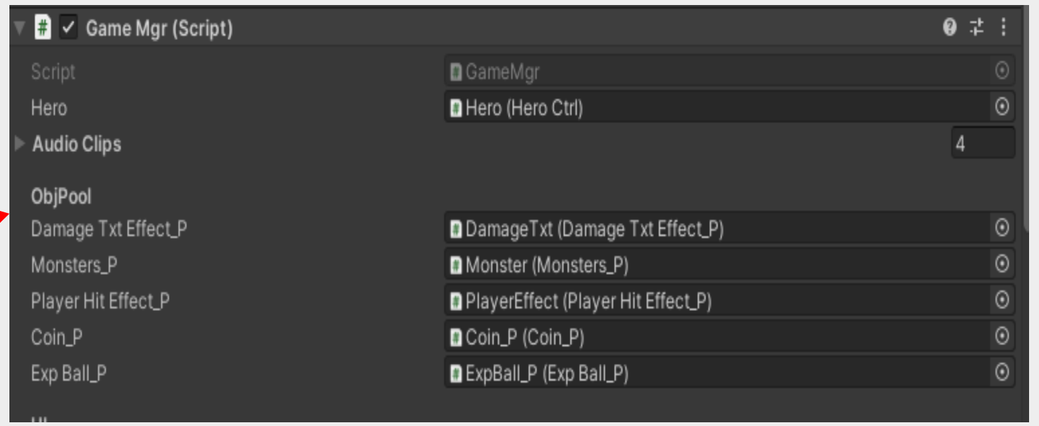
3. 시스템 설명 - 오브젝트 풀링

2) 오브젝트풀링을 사용하기 위해 게임매니저에 추가하여 사용

GameMgr.cs

```
[Header("ObjPool")]//오브젝트풀을 사용하는 오브젝트들
public DamageTxtEffect_P DamageTxtEffect_P; //데미지적용시 수치를 나타내는
public Monsters_P monsters_P;                    //몬스터 오브젝트
public PlayerHitEffect_P playerHitEffect_P;      //플레이어 피격 이펙트
public Coin_P coin_P;                            //코인 오브젝트
public ExpBall_P expBall_P;                     //경험치 오브젝트
```





사용 예시) 코인

```
참조 2개
public void SpawnCoin(Vector2 spawnPos)
{//몬스터 처치시 호출
    //지정된 위치에서 원으로 랜덤한 위치 에 스폰
    spawnPos = spawnPos + Random.insideUnitCircle;
    coin_P.GetObj().SetCoin(spawnPos);
}
```

코인 오브젝트풀링에서 하나의 오브젝트를 가져와 소환

```
Unity 스크립트(자산 참조 1개)|참조 1개
public class CoinObj : MonoBehaviour
{//코인 오브젝트 클래스
    [SerializeField]private int coin = 1; // 코인 값

    참조 1개
    public void SetCoin(Vector2 pos) //코인 위치와 활성화 시키기
    {
        gameObject.transform.position = pos;
        gameObject.SetActive(true);
    }

    Unity 메시지|참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.CompareTag("Player"))//플레이어 체크
        {
            gameObject.SetActive(false);
            GameMgr.Inst.GetCoin(coin); //코인 획득
            GameMgr.Inst.coin_P.ReturnObj(this);
            //오브젝트풀링을 위해 오브젝트풀에 오브젝트 리턴
        }
    }
}
```

습득시

게임매니저를 걸쳐 오브젝트 반환

3. 시스템 설명 - 카메라 이동

플레이어를 따라다니는 카메라

단. 맵을 벗어나지 않도록 구현

```
public Transform targetTr; //쫓아다닐 목표
private Vector3 cameraPos;
private Vector3 camWMax;

private float camSizeX = 0.0f; //카메라뷰의 크기
private float camSizeY = 0.0f;
//계산용
private float xVelocity = 0.0f;
private float yVelocity = 0.0f;
private float smoothTime = 0.2f;
public Tilemap tilemap; //타일맵(크기를 가져올려고)
Vector3 mapSize;

Unity 메시지 | 참조 0개
private void Awake()
{
    cameraPos = transform.position;
}

Unity 메시지 | 참조 0개
private void Start()
{
    mapSize.x = tilemap.size.x / 2; //타일맵 사이즈
    mapSize.y = tilemap.size.y / 2; //시작이 중심(0,0)이기때문에 나누기2를 했다.

    camWMax = Camera.main.ViewportToWorldPoint(Vector3.one);
    camSizeX =  camWMax.x - transform.position.x + 1;
    camSizeY = camWMax.y - transform.position.y + 1;
}
```

처음

맵(타일맵)으로 부터 맵의 크기를 구한다.

카메라의 뷰크기값을 구한다

```
Unity 메시지 | 참조 0개
private void LateUpdate()
{
    cameraPos = transform.position; //현재 위치
    //캐릭터의 움직임에 맞춰 이동하기
    cameraPos.x = Mathf.SmoothDamp(transform.position.x, targetTr.position.x, ref xVelocity, smoothTime);
    cameraPos.y = Mathf.SmoothDamp(transform.position.y, targetTr.position.y, ref yVelocity, smoothTime);

    //카메라 지형 밖으로 안나가게 //최대 최솟값 좌표 구하기
    if (cameraPos.x + camSizeX > mapSize.x)
        cameraPos.x = mapSize.x - camSizeX;

    if (cameraPos.x - camSizeX < -mapSize.x)
        cameraPos.x = -mapSize.x + camSizeX;

    if (cameraPos.y + camSizeY > mapSize.y)
        cameraPos.y = mapSize.y - camSizeY;

    if (cameraPos.y - camSizeY < -mapSize.y)
        cameraPos.y = -mapSize.y + camSizeY;
    //위치 값
    transform.position = cameraPos;
}
```

LateUpdate에서 캐릭터 움직임이 끝난후

이동할수 있도록 한다

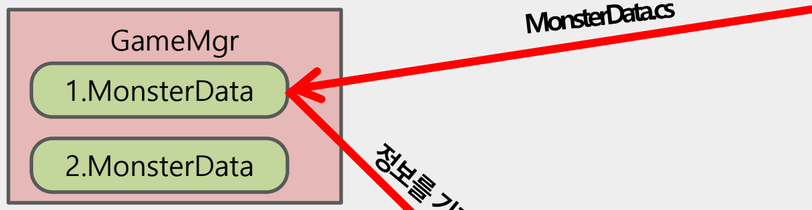
게임 구성

3. 시스템 설명 - 몬스터 시스템

기본 몬스터

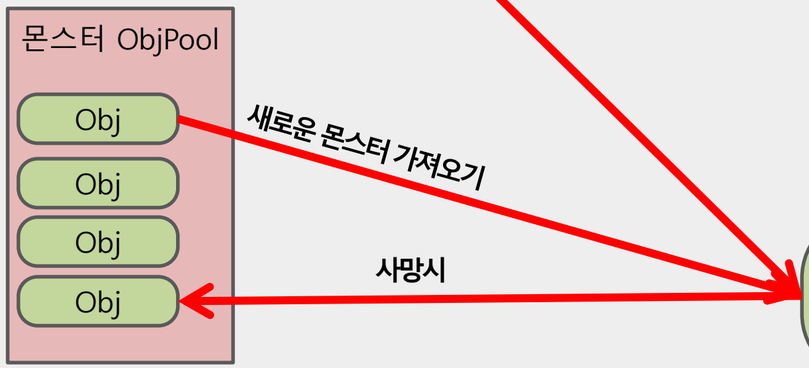
기본적으로 몬스터오브젝트 하나에

필요한 MonsterData를 가지고 변경하여 사용



MonsterData.cs

정보를 가지고 셋팅

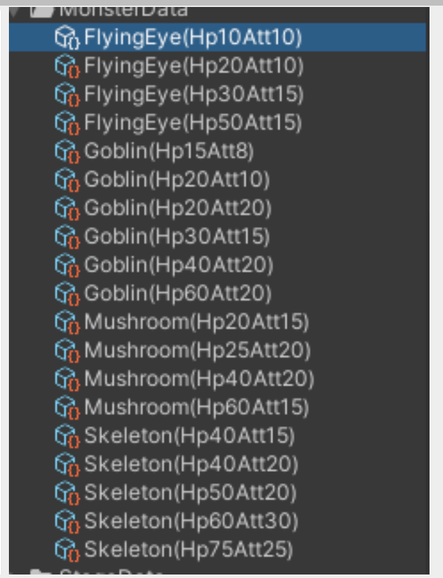


새로운 몬스터 가져오기

사망시

MonsterObj
Monster.cs

```cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using CardHelp;

[CreateAssetMenu(fileName = "MonsterData", menuName = "Scriptable Object Asset/MonsterData")]
Unity 스크립트 | 참조 2개
public class MonsterData : ScriptableObject , SetCard
{//몬스터 데이터 ScriptableObject
    public Sprite monsterImg; //몬스터 카드 이미지
    public RuntimeAnimatorController monsterAnimator; //애니메이터
    [Header("Colider")]
    public Vector2 offset;
    public float coliderSize;
    [Header("Status")]
    public Vector2 attackBoxSize;
    public int hp;
    public int AttPw;
    public float Speed;

    참조 3개
    public CardData GetCard() //카드 설명
    {
        CardData card;
        card.img = monsterImg;
        card.info = "Hp : " + hp + "\nAttack : " + AttPw;
        return card;
    }
}
```




```cs
참조 1개
public void SetStatus(MonsterData monsterData)
{//몬스터 데이터를 가지고 셋팅
    runtimeAnimatorController = monsterData.monsterAnimator; //애니메이터
    collider.offset = monsterData.offset;   //몸체 콜라이더
    collider.radius = monsterData.coliderSize;  //사이즈
    //공격박스
    attackSize = monsterData.attackBoxSize;
    //능력치
    attackPower = monsterData.AttPw;
    this.hp = monsterData.hp;
    speed = monsterData.Speed;

    monster_State = Monster_State.Idle;
    gameObject.SetActive(true);
}
```

게임 구성

3. 시스템 설명 - 몬스터 시스템

Monster.cs 일부

MonsterObj
Monster.cs

거리에 의한 상태

```
Unity 메시지 | 참조 0개
private void Update()
{
    //방향과 거리 계산
    targetToThis = targetTr.position - transform.position; //타겟과의 거리관계
    dir = targetToThis.normalized;      //방향값
    float dis = targetToThis.sqrMagnitude;  //거리 길이 변환
    //길이에 따른 상태 변환
    if (dis > attakcDis)
        MonsterState_Update(Monster_State.Move);
    else if (dis <= attakcDis)
        MonsterState_Update(Monster_State.Attack);

    Flip_Update();//이미지 좌우변환
}
```

상태 변화 체크

```
참조 3개
void MonsterState_Update(Monster_State newStatue) //몬스터의 상태 변화에 따른 애니메이터 변환해주기
{
    if (monster_State.Equals(newStatue)) return; //이전과 같은 상태면 리턴

    if (monster_State.Equals(Monster_State.Die)) return; //죽은 상태면

    //새로운 상태 적용
    monster_State = newStatue;
    //속도 초기화
    rigidbody.velocity = Vector2.zero;

    switch (monster_State)//상태별 애니메이션
    {
        case Monster_State.Idle:
            animator.SetBool("Move", false);
            break;
        case Monster_State.Move:
            animator.SetBool("Move", true);
            break;
        case Monster_State.Attack:
            {
                animator.SetBool("Move", false);
                animator.SetTrigger("Attack");
            }
            break;
        case Monster_State.Die:
            {
                animator.SetBool("Move", false);
                animator.SetTrigger("Death");
                StartCoroutine(Die_Co());
            }
            break;
        default:
            break;
    }
}
```

피격

데미지를 전달하는

인터페이스 의한

```
참조 2개
public void TakeDamage(int value = 10) //데미지를 받는 함수
{
    if (hp <= 0)
        return;

    hp -= value;
    rigidbody.velocity = Vector2.zero;
    //데미지 이펙트
    GameMgr.Inst.DamageTxtEffect_P.GetObj().SetDamageTxt(value, damageTxtPos.position);

    //공격중이고 죽을체력이아니면
    if (monster_State.Equals(Monster_State.Attack) && hp > 0)
        return;

    //애니메이션 실행 ... 이미 애니메이션 실행중이면
    if (!animator.GetCurrentAnimatorStateInfo(0).IsName("Hit"))
        animator.SetTrigger("Hit");

    if (hp <= 0)
        MonsterState_Update(Monster_State.Die);

}
```

사망시

```
IEnumerator Die_Co()
{
    //게임매니저
    GameMgr.Inst.MonsterKill();
    //경험치볼 생성
    GameMgr.Inst.SpawnExpBall(transform.position, 2);
    //코인 생성
    if(Random.Range(0, 2) == 0)
        GameMgr.Inst.SpawnCoin(transform.position);

    //넉백 타겟과의 반대 방향으로
    rigidbody.velocity = Vector2.zero;
    rigidbody.AddForce(targetToThis.normalized * -1 * 300.0f);

    float a = 1;
    Color color = Color.white;
    collider.enabled = false;

    while (a > 0) //투명해지면서 사라지는
    {
        yield return null;
        a -= Time.deltaTime;
        color.a = a;
        spriteRenderer.color = color;
    }

    Die_Event();
}
//DIE 이벤트
참조 1개
public void Die_Event()
{
    gameObject.SetActive(false);
    spriteRenderer.color = Color.white;
    GameMgr.Inst.monsters_P.ReturnObj(this);
}
```

공격 이벤트

애니메이션 이벤트를 이용

```
public void Attack_Event()
{
    //공격포인터 중심에서 네모 크기 만큼 펼쳐 충돌된 콜라이더 가져오기
    Collider2D hit = Physics2D.OverlapBox(transform.position + targetToThis.normalized, attackSize, 0, heroLayer);
    if (hit && hit.CompareTag("Player"))
    {
        //hit.SendMessage("TakeDamage", attackPower);
        targetHero.TakeDamage(attackPower);
    }
}
```

게임 구성

3. 시스템 설명 - 스킬 시스템

베이스 Skill.cs

모든 스킬에 상속을 하여 관리하기 편하게 만듬

```
Unity 스크립트 | 참조 9개
public  class Skill : MonoBehaviour , ICardLvUp
{//스킬의 베이스가 되는 클래스
    protected HeroCtrl hero; //캐릭터
    public int skill_Lv = 0; //스킬레벨
    public int skill_MaxLv = 0; //스킬 최대 레벨 기본적으로 7단계
    public bool getSkill = false; //스킬을 얻은 스킬인지

    public string[] skillLvInfo = new string[7]; //스킬 레벨별 상승 설명
    public int[] skillPw = new int[7];              //스킬 레벨별 데미지
    public Sprite skillSprite;

    [SerializeField] protected float skillCool; //스킬 쿨타임
    protected AudioSource audioSource;
    protected Coroutine skill_Co; //실행중이 스킬 코루틴 함수 저장용

    참조 1개
    string CardInfo  //카드에 사용할 설명문
    {
        get
        {
            if (getSkill) //이미 얻은 스킬이면 다음레벨 설명
                return skillLvInfo[skill_Lv + 1];
            else
                return skillLvInfo[0];
        }
    }
    //적용 스킬 쿨타임 (스킬 쿨타임 * 캐릭터의 스킬 쿨타임 적용 100 => 1 적용 80 => 0.8)
    참조 5개
    public float SkillCool { get { return skillCool * (hero.SkillCool *0.01f); } }

    //실제 스킬 데미지 캐릭터의 추가 스킬데미지도 합산하여 계산
    참조 1개
    public int SkillDamage { get { return skillPw[skill_Lv] + hero.skillPower; } }

    Unity 메시지 | 참조 0개
    private void Awake()
    {
        hero = FindObjectOfType<HeroCtrl>();
        audioSource =GetComponent<AudioSource>();
    }
```

```
Unity 메시지 | 참조 0개
private void Start()
{
    Skill_Init(); //스킬 초기 설정을 하는
}
참조 6개
public virtual void Skill_Init(){ }//스킬 초기 설정을 하는
참조 5개
public void SkillStart() //스킬을 실행하는 함수
{
    if(skill_Co != null) //만약 기존 스킬이 돌아가는 중이면
    {//중단 후 다시 시작
        StopCoroutine(skill_Co);
        skill_Co = null;
    }
   //스킬 시작하기
   skill_Co =  StartCoroutine(SkillStart_Co());
}

참조 5개
public void TakeMonsterDamage(ITakeDamage monster)
{//몬스터에게 데미지를 주는 콜라이더에서 이 함수를 호출
    monster.TakeDamage(SkillDamage);
}

참조 7개
public virtual IEnumerator SkillStart_Co() { yield break; }//스킬이 돌아가는 코루틴 (필수)
참조 9개
public virtual void SkillRefresh() { }//스킬 상태를 초기화 해줌 //스킬 정지와 같은 효과
```

```
참조 1개
public void SkillLvUp() //스킬 레벨업 시켜주는
{
    skill_Lv++;
    SkillRefresh();
    SkillStart();
}

참조 3개
public CardData GetCard() //카드데이터 반환
{
    CardData card;
    card.img = skillSprite;
    card.info = CardInfo;
    return card;
}

참조 3개
public bool LevelPossible() //레벨업이 가능한지
{
    if (!getSkill || skill_Lv == skill_MaxLv)
        return false;
    else
        return true;
}

참조 3개
public void LevelUp() //카드를 통한 레벨업
{
    if (getSkill)
        SkillLvUp();
    else
    {
        getSkill = true;
        this.gameObject.SetActive(true);
    }
}
```

스킬의 데미지 판정을 도와주는 콜라이더.cs

```
Unity 스크립트 | 참조 10개
public class SkillDamageCollider : MonoBehaviour
{  //스킬의 데미지 판정에 도움을주는 콜라이더 클래스

    public delegate void Event(ITakeDamage monster);
    public Event OnTriggerMonster;

    protected Collider2D collider2D;

    Unity 메시지 | 참조 2개
    public virtual void Awake()
    {
        collider2D = GetComponent<Collider2D>();
    }
}
```

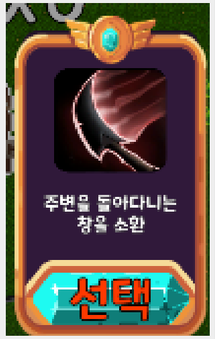
3. 시스템 설명 - 스킬 시스템

Spear_Skill.cs

캐릭터의 주변을 돌면서 공격하는 창을 소환

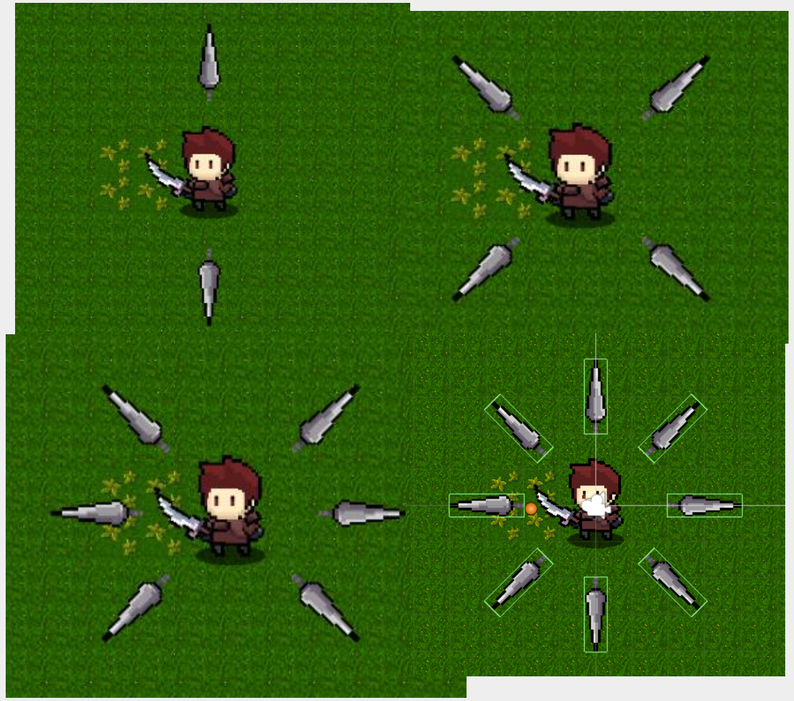
지속시간동안 나왔다 쿨타임 대기후 다시 소환

창에 다을때마다 데미지를 준다




```
Unity 스크립트(자산 참조 2개) | 참조 0개
public class Spear_Skill : Skill
{
    //주변을 돌아다니면 공격하는 창 소환
    [Header("Spear")]
    public GameObject[] Lv_Group; //레벨별 창 그룹
    //레벨에 따른 그룹 오브젝트 리턴
    참조 2개
    GameObject CurSkillObj { get { if (skill_Lv == 0) return Lv_Group[0]; else return Lv_Group[skill_Lv / 2]; } }

    참조 2개
    public override void Skill_Init()
    {
        for (int i = 0; i < Lv_Group.Length; i++)
        {
            //SkillDamageCollider 에 데미지를 주는 함수를 적용한다.
            var skillDamageBoxes = Lv_Group[i].GetComponentsInChildren<SkillDamageCollider>(true);
            foreach (var colider in skillDamageBoxes)
                colider.OnTriggerMonster += TakeMonsterDamage;
        }
    }
```

스킬 레벨에 따라 나타나는 창의 갯수가 다름



```
참조 2개
public override IEnumerator SkillStart_Co()
{
    float timer = 5.0f; //5초동안
    float angle = 0.0f; //회전 각도

    CurSkillObj.SetActive(true); //레벨별 오브젝트 on
    audioSource.Play();//효과음
    while(timer >  0.0f)
    {
        timer -= Time.deltaTime;
        angle += Time.deltaTime * 200;
        //위치와 회전 조정
        transform.position = hero.transform.position;
        transform.eulerAngles = new Vector3(0, 0, angle);
        yield return null;
    }
    //지속시간 끝나고
    audioSource.Stop();
    CurSkillObj.SetActive(false);

    //스킬 쿨타임 이후 다시 스타트
    yield return new WaitForSeconds(SkillCool);
    SkillStart();

}

참조 4개
public override void SkillRefresh()
{
    audioSource.Stop();
    //모든 오브젝트 끄기
    for (int i = 0; i < Lv_Group.Length; i++)
        Lv_Group[i].SetActive(false);
    //실행중인 코루틴 중지
    StopAllCoroutines();
}
```

창의 콜라이더의 충돌시 등록된 데미지 함수를 전달

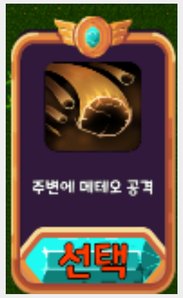
```
Unity 스크립트(자산 참조 40개) | 참조 0개
public class Spear_Collider : SkillDamageCollider
{
    Unity 메시지 | 참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Monster"))
            OnTriggerMonster?.Invoke(collision.GetComponent<ITakeDamage>());
    }
}
```

게임 구성

3. 시스템 설명 - 스킬 시스템

Meteors_Skill.cs

캐릭터 주변에 매테오를 소환하여 주변 적에게 데미지를 준다



스킬 레벨에 따라 떠러지는 매테오의 갯수와 데미지가 다름

```
Unity 스크립트(자산 참조 2개) | 참조 0개
public class Meteors_Skill : Skill
{
    [Header("Meteor")]
    public GameObject[] meteorObj; //메테오 오브젝트들
    public int[] meteors;                //스킬 레벨별 갯수

    참조 2개
    public override void Skill_Init()
    {
        for (int i = 0; i < meteorObj.Length; i++)
        {
            //스킬콜라이더를 찾아 가져와 데미지적용 함수 적용//처음에는 오브젝트 꺼져있으니 true 로 설정
            SkillDamageCollider skillDamageBoxes = meteorObj[i].GetComponentInChildren<SkillDamageCollider>(true);
            skillDamageBoxes.OnTriggerMonster += TakeMonsterDamage;
        }

    }
    참조 2개
    public override IEnumerator SkillStart_Co()
    {
        audioSource.Play();

        for (int i = 0; i < meteors[skill_Lv]; i++) //레벨별 메테오 스폰
        {
            //캐릭터 주변 원좌표를 구해 소환
            Vector2 startPos = (Vector2)hero.transform.position + Random.insideUnitCircle * 10;
            meteorObj[i].transform.position = startPos;
            meteorObj[i].SetActive(true);
            yield return new WaitForSeconds(0.1f); //스폰주기
        }
        audioSource.Stop();
        //쿨타임대기후 다시 스킬 시작
        yield return new WaitForSeconds(SkillCool);
        SkillStart();
    }

    참조 4개
    public override void SkillRefresh()
    {
        audioSource.Stop();
        StopAllCoroutines();

        for (int i = 0; i < meteorObj.Length; i++)
            meteorObj[i].SetActive(false);
    }
}
```

매테오에 오브젝트에 붙어있는

Meteors_Collider.cs

```
Unity 스크립트(자산 참조 18개) | 참조 0개
public class Meteors_Collider : SkillDamageCollider
{
    public AudioSource audioSource;
    Vector2 dir = new Vector2(1.0f, -1.0f).normalized; //대각선 방향
    bool move = false;    //움직이는 지
    Unity 메시지 | 참조 0개
    private void Update()
    {
        if (move)//이동 가능하면 대각선으로 떨어지는 것처럼 이동
            transform.position += (Vector3)dir * Time.deltaTime * 10.0f;
    }

    Unity 메시지 | 참조 0개
    private void OnEnable()
    {
        //콜라이더를 끄고 이동
        collider2D.enabled = false;
        move = true;
    }

    참조 0개
    public void Meteors_Evenet()
    {//메테오 애니메이션에 맞추어 터지는 순간
        //콜라이더를 키고 움직임을 막는다.
        audioSource.Play();
        collider2D.enabled = true;
        move = false;
    }
    참조 0개
    public void MeteorsEnd_Evenet()
    {//애니메이션 종료시 오브젝트 끄기
        this.gameObject.SetActive(false);
    }

    Unity 메시지 | 참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Monster")) //몬스터에게 데미지 판정
            OnTriggerMonster?.Invoke(collision.GetComponent<ITakeDamage>());
    }
}
```

게임 구성

3. 시스템 설명 - 스킬 시스템

GuidedSword_Skill.cs

캐릭터를 따라 다니면

유도검을 날려 데미지를 준다.

스킬 레벨에 따라 칼의모형과 데미지

날아가는 속도가 바뀜

```
Unity 스크립트(자산 참조 2개)|참조 0개
public class GuidedSword_Skill : Skill
{
    //몬스터를 추적하여 공격하는 칼 소환
    [Header("GuidedSword")]
    public Sprite[] Swordsprites; //레벨별 칼 이미지변경을 위한리스트
    public int[] speed;           //레벨별 속도
    public Transform[] swordPos;    //칼 스폰 위치
    public GameObject[] swordObj;//칼의 오브젝트

    //추적 칼의 정보 변경을위한 리스트
    List<GuidedSword_Collider> guidedSword_Colliders = new List<GuidedSword_Collider>();
    //추적 칼을 발사할때 순서를 정하기 위한
    Queue<GuidedSword_Collider> sword_Qu = new Queue<GuidedSword_Collider>();

    참조 2개
    public override void Skill_Init()
    {
        for (int i = 0; i < swordObj.Length; i++) //추적 칼 오브젝트에서 필요한 정보 셋팅
        {
            var skillDamageBoxes = swordObj[i].GetComponentInChildren<GuidedSword_Collider>(true);
            skillDamageBoxes.OnTriggerMonster = TakeMonsterDamage; //데미지함수 적용
            skillDamageBoxes.DeQuObj = this.DequObj; //다시 큐에 들어오기 위한
            skillDamageBoxes.InitSword(swordPos[i].transform); //위치 셋팅
            guidedSword_Colliders.Add(skillDamageBoxes);    //리스트 추가
            swordObj[i].SetActive(false); //우선 오브젝트 비활성화
        }
    }

    Unity 메시지|참조 0개
    private void Update()
    { //스킬 본체는 캐릭터를 따라가게끔
        transform.position = hero.transform.position;
    }

    참조 2개
    public override IEnumerator SkillStart_Co() //스킬 코루틴
    {
        for (int i = 0; i < swordObj.Length; i++)
        {
            swordObj[i].SetActive(true);
            sword_Qu.Enqueue(guidedSword_Colliders[i]); //큐에 넣기
        }

        while (true)
        {
            yield return new WaitForSeconds(SkillCool);

            if (sword_Qu.Count > 0)//대기중인 추적칼이 있다면
            {
                GameObject targetObj = FindNearestObjectByTag("Monster"); //가까운 몬스터를 찾아 리턴
                if (targetObj != null)
                {
                    GuidedSword_Collider sword = sword_Qu.Dequeue(); //큐에서 빼오기
                    sword.SetTarget(targetObj); //타겟 설정
                    audioSource.Play(); //효과음
                }
            }
        }
    }
```

GuidedSword_Collider.cs

날아가는 칼의 콜라이더

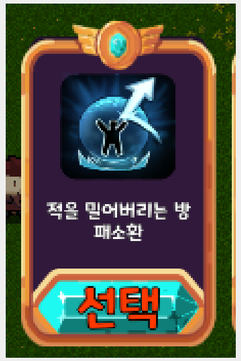
```
Unity 스크립트(자산 참조 10개)|참조 9개
public class GuidedSword_Collider : SkillDamageCollider
{
    public delegate void GuidedSword_Event(GuidedSword_Collider guidedSword_Collider);
    public GuidedSword_Event DeQuObj; //다시 사용할수 있도록 돌려놓기 위한

    public SpriteRenderer img; //추적칼 이미지
    bool move = false;          //움직이는지

    public GameObject target;  //타겟
    public float speed = 10.0f; //날아가는 속도
    Transform originPos;        //돌아와야하는 위치
    Vector2 dir = Vector2.zero; //방향
    Unity 메시지|참조 0개
    private void Update()
    {
        if (move)
        {   //타겟이 활성화 된다면
            if (target.activeSelf)//방향을 정한다
                dir = (target.transform.position - transform.position).normalized;
            else//비활성화 상태 라면 추적칼 사라지게
                gameObject.SetActive(false);

            //이동 및 방향
            transform.position += (Vector3)dir * Time.deltaTime * speed;
            transform.up = dir; //칼 머리가 이동방향을 향해 돌아간다.

        }

    }
    Unity 메시지|참조 0개
    private void OnEnable()
    {
        //시작 위치 잡기
        transform.position = originPos.position;
        transform.rotation = Quaternion.identity;
        //벨사전까지 대기
        collider2D.enabled = false;
        move = false;
    }
    Unity 메시지|참조 0개
    private void OnDisable()
    { //움직이고 있는도중 사라지면
        if (move)
        {
            move = false;
            DeQuObj?.Invoke(this); //다시 스킬본체에 돌려주기
        }

    }
    참조 1개
    public void SetTarget(GameObject target)
    {//타겟 설정시 움직이기 시작
        this.target = target;
        move = true;
        collider2D.enabled = true;
    }

    참조 1개
    public void InitSword(Transform origin)
    {//처음 칼 셋팅
        originPos = origin; //돌아오는 위치
        transform.position = origin.position;
    }

    참조 2개
    public void SetSword(Sprite sprite, float speed)
    {//레벨별 이미지와 날아가는 속도 설정을 위한
        img.sprite = sprite;
        this.speed = speed;
    }

    Unity 메시지|참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Monster")) //몬스터와 충돌시
        {//데미지 주기
            OnTriggerMonster?.Invoke(collision.GetComponent<ITakeDamage>());
            target = null;
            gameObject.SetActive(false);
        }

    }
```

게임 구성

3. 시스템 설명 - 스킬 시스템

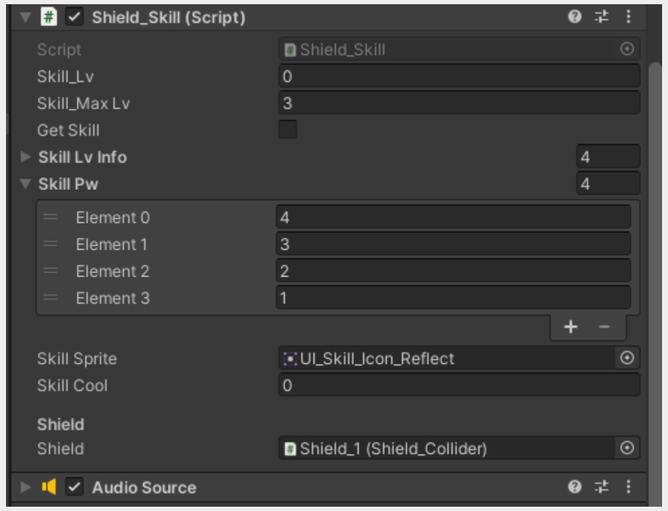
Shield_Skill.cs

가까이 온 몬스터를 방패로 밀쳐 낸다.



방패 스킬은
데미지가 없다.
단, 스킬데미지 값을 스킬 쿨타임 값으로 사용하여
스킬 레벨업시 스킬 쿨이 감소

```
public class Shield_Skill : Skill
{
    //가까운 적이 있으면 밀쳐주는 방패스킬
    [Header("Shield")]
    public Shield_Collider Shield; //방패 오브젝트

    //방패 스킬은 스킬데미지가 곧 쿨타임이 된다

    - 참조
    public override IEnumerator SkillStart_Co()
    {
        while(true)
        {
            GameObject monsterObj = FindNearestObjectByTag("Monster"); //가까운 몬스터를 찾아 리턴
            if(monsterObj)
            {
                Vector2 temp = monsterObj.transform.position - hero.transform.position;
                if (temp.magnitude < 5.0f) //거리 체크
                {
                    audioSource.Play();
                    //밀치는방패 소환
                    Shield.SpawnShield(hero.transform.position, temp.normalized);
                }
                //0.5초후 사라지게
                yield return new WaitForSeconds(0.5f);
                Shield.gameObject.SetActive(false);
            }
            yield return new WaitForSeconds(skillPw[skill_Lv]);
        }
    }

    - 참조
    public override void SkillRefresh()
    {
        Shield.gameObject.SetActive(false);
        StopAllCoroutines();
    }
}
```



Shield_Collider.cs

방패 콜라이더
스킬 사용시 위치와 방향만 정해준다.

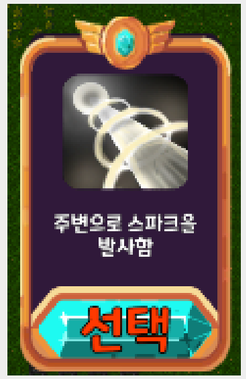
```
Unity 스크립트 | 참조 1개
public class Shield_Collider : MonoBehaviour
{
    Rigidbody2D rigidbody; //밀치기 위한 물리 추가
    public float speed = 0; //속도

    Vector2 dir; //밀치는 방향

    // Start is called before the first frame update
    Unity 메시지 | 참조 0개
    private void Awake()
    {
        rigidbody = GetComponent<Rigidbody2D>();
    }

    참조 1개
    public void SpawnShield(Vector2 spawnPos, Vector2 dir)
    {
        this.dir = dir;
        transform.position = spawnPos;
        gameObject.SetActive(true);

        //방향에 따른 오브젝트 방향결정
        if (dir.x < 0)
            transform.right = -dir;
        else
            transform.right = dir;

        rigidbody.velocity = dir * speed;
    }

}
```

3. 시스템 설명 - 스킬 시스템

Spark_Skill.cs

랜덤한 방향으로 스파크를 날립니다.

레벨업 시 스파크의 개수가 증가한다.




```
public class Spark_Skill : Skill
{
    //주변으로 스파크를 발사함
    public Spark_Collider[] sparkObj;//스파크 오브젝트
    public int[] count; //레벨별 갯수배열

    참조 6개
    public override void Skill_Init()
    {
        for (int i = 0; i < sparkObj.Length; i++) //모든 스파크 콜라이더에 데미지함수 적용
        {
            SkillDamageCollider skillDamageBoxes = sparkObj[i].GetComponentInChildren<SkillDamageCollider>(true);
            skillDamageBoxes.OnTriggerMonster += TakeMonsterDamage;
        }
    }

    참조 7개
    public override IEnumerator SkillStart_Co()
    {
        audioSource.Play();
        for (int i = 0; i < count[skill_Lv]; i++)
        {
            //캐릭터 중앙에서  랜덤한 방향으로 발사함
            sparkObj[i].SetSpark(Random.insideUnitCircle.normalized);
            sparkObj[i].transform.position = hero.transform.position;
            sparkObj[i].gameObject.SetActive(true);
        }

        yield return new WaitForSeconds(SkillCool);
        SkillStart();
    }

    참조 9개
    public override void SkillRefresh()
    {
        for (int i = 0; i < sparkObj.Length; i++)
            sparkObj[i].gameObject.SetActive(false);

        StopAllCoroutines();
    }
}
```

Spark_Collider.cs

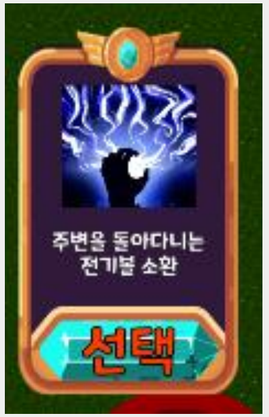
```
public class Spark_Collider : SkillDamageCollider
{
    Vector2 dir; //방향
    Animator animator; //애니메이션 변경을 위한
    float lifeTime = 3.0f; //생존시간
    float speed = 10.0f; //날아가는 속도

    Unity 메시지 | 참조 2개
    public override void Awake()
    {
        base.Awake();
        animator = GetComponent<Animator>();
    }

    Unity 메시지 | 참조 0개
    private void OnEnable() //스파크 발사
    {
        collider2D.enabled = true; //콜라이더 on
        speed = 10.0f;             //속도
        lifeTime = 3.0f;           //3초후 제거
    }
    참조 1개
    public void SetSpark(Vector2 dir)
    {
        this.dir = dir;
        transform.right = dir; //방향에따른 회전
        gameObject.SetActive(true);
    }

    Unity 메시지 | 참조 0개
    private void Update()
    {
        //움직이기
        transform.position += (Vector3)dir * Time.deltaTime * speed;

        lifeTime -= Time.deltaTime;
        if (lifeTime <= 0.0f)
            gameObject.SetActive(false);
    }
    Unity 메시지 | 참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Monster")) //충돌시
        {
            //데미지 적용후
            OnTriggerMonster?.Invoke(collision.GetComponent<ITakeDamage>());
            //콜라이더 off
            collider2D.enabled = false;
            //애니메이션 히트 판정애니메이션 실행
            animator.SetTrigger("Hit");
            speed = 0.0f;
            lifeTime = 0.5f;
        }
    }
```

게임 구성

3. 시스템 설명 - 스킬 시스템

Electricball_Skill.cs

캐릭터의 주변을 돌아다니며
구체에 닿은 몬스터에게 데미지를 준다.
레벨업 시 구체의 크기가 커지고
데미지가 증가한다.




```
Unity 스크립트 | 참조 0개
public class Electricball_Skill : Skill
{
    //캐릭터 주변을 돌아다니며 전기구체 소환
    public Electricball_Collider electricball;
    public float speed = 1.0f;

    참조 6개
    public override void Skill_Init()
    {
        //전기구체 콜라이더에 데미지함수 적용
        electricball.OnTriggerMonster += TakeMonsterDamage;
    }

    Unity 메시지 | 참조 0개
    private void Update()
    {
        //스킬 본체 위치조정
        transform.position = hero.transform.position;
    }

    참조 7개
    public override IEnumerator SkillStart_Co()
    {
        electricball.gameObject.SetActive(true);

        //구체는 레벨별로 크기가 커지게
        if (skill_Lv == 0)
            electricball.transform.localScale = Vector3.one;
        else
            electricball.transform.localScale = Vector3.one * (1 + skill_Lv / 2);

        while (true)//주변을 계속돌아감
        {
            transform.Rotate(Vector3.forward * Time.deltaTime * speed);
            yield return null;
        }
    }

    참조 9개
    public override void SkillRefresh()
    {
        electricball.gameObject.SetActive(false);
        StopAllCoroutines();
    }
}
```

Spark_Collider.cs

```
Unity 스크립트 | 참조 1개
public class Electricball_Collider : SkillDamageCollider
{
    public float speed = 1.0f;
    Vector3 pos = Vector3.zero;

    Unity 메시지 | 참조 0개
    private void Update()
    {
        //주변을 돌며 원을 그리듯 움직임
        pos.x = 5 * Mathf.Sin(Time.time * speed) ;
        transform.localPosition = pos;
    }

    Unity 메시지 | 참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Monster"))
            OnTriggerMonster?.Invoke(collision.GetComponent<ITakeDamage>());

    }
}
```

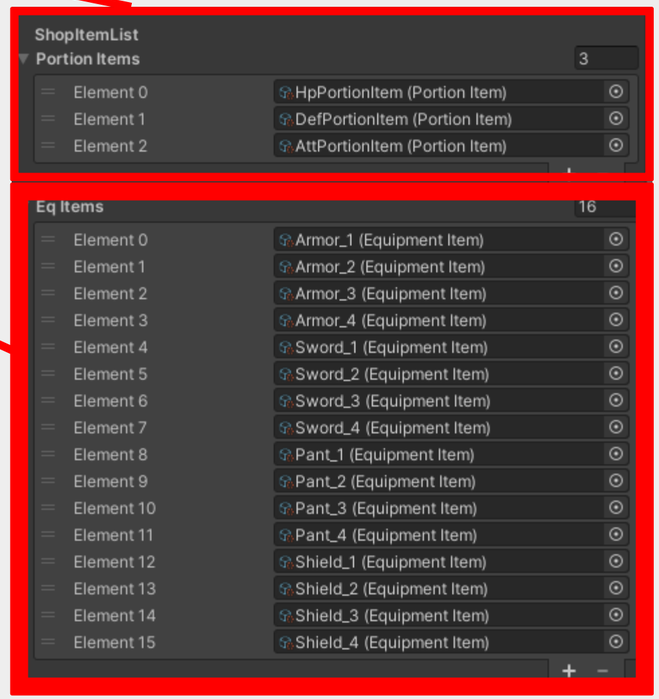
3. 시스템 설명 – 상점 시스템

라운드 종료 시

맵 가운데에 상점 등장

상정 을 클릭 하면 상점UI 오픈


```
public void ShopSetting() //판매목록 셋팅
{
    BuyEvent += RefreshCoin; //구매시 호출될 함수 넣어주기
    //판매목록 랜덤값을 이용하여 채워주기
    shopPortionCards[0].SetCard(portionItems[Random.Range(0, portionItems.Length)]);
    shopPortionCards[1].SetCard(portionItems[Random.Range(0, portionItems.Length)]);

    shopEqCards[0].SetCard(eqItems[Random.Range(0, eqItems.Length)]);
    shopEqCards[1].SetCard(eqItems[Random.Range(0, eqItems.Length)]);
    shopEqCards[2].SetCard(eqItems[Random.Range(0, eqItems.Length)]);
}
```



3. 시스템 설명 – 상점 시스템

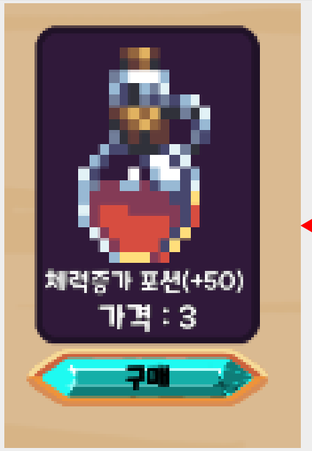
상점에 나오는 장비를클릭시

그 아이템의 능력치를 볼수 있다.

장비를 구매시 기존 장비는 파괴 되고

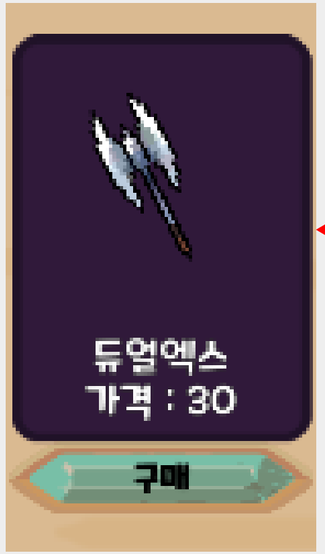
장착 된다.

포션 아이템은 바로 능력치에 적용된다.




```
[CreateAssetMenu(fileName = "PortionItem", menuName = "Scriptable Object Asset/PortionItem")]
Unity 스크립트|참조 4개
public class PortionItem : ScriptableObject
{//포션아이템을 만들기 위한
    public Sprite img;  //물건 이미지
    public  AbilityType AbilityType;    //어떤물약인지
    public string itemName; //아이템 이름
    public int value;   //수치
    public int price; //가격
}
```

Script	PortionItem
Img	small Potions_0
Ability Type	Hp
Item Name	체력증가 포션(+50)
Value	50
Price	3




```
[CreateAssetMenu(fileName = "EquipmentItem", menuName = "Scriptable Object Asset/EquipmentItem")]

Unity 스크립트|참조 13개
public class EquipmentItem : ScriptableObject
{
    public EquipmentType Type;    //어떤장비인지
    public Sprite[] img;  //장비 이미지
    public string itemName; //아이템이름
    public int value;   //수치  방어력, 공격력, 등등
    public int price; //가격
}
```

Script	EquipmentItem
Type	Weapon_R
Img	1
Element 0	New_Weapon_08
Item Name	듀얼엑스
Value	20
Price	30

3. 시스템 설명 – 상점 시스템

상점 구매 카드

```
Unity 스크립트 | 참조 1개
public class ShopEqCard : Card,IPointerClickHandler
{
    EquipmentItem buyItme;//판매하는 장비아이템

    //장비별 보이는 이미지
    //기본 베이스 Card의 image변수에는 무기이미지를 등록
    public Image Sheild;
    public Image[] Armors;
    public Image[] Pant;

    public TextMeshProUGUI priceTxt; //가격표시
    public Button buyBtn;   //구매 버튼

    Unity 메시지 | 참조 0개
    private void Start()
    {
        buyBtn.onClick.AddListener(BuyItem); //버튼등록
    }
```

```
public void SetCard(EquipmentItem item)//장비 아이템정보를 가지고 카드셋팅
{
    buyItme = item; //구매 아이템 등록
    //우선 모든 이미지 off 하기
    image.gameObject.SetActive(false);
    Sheild.gameObject.SetActive(false);
    for (int i = 0; i < Armors.Length; i++)
        Armors[i].gameObject.SetActive(false);

    for (int i = 0; i < Pant.Length; i++)
        Pant[i].gameObject.SetActive(false);
    //우선 모든 이미지 off 하기

    //부위별로 맞추어 이미지스프라이트 적용
    if (item.Type == EquipmentType.Weapon_L || item.Type == EquipmentType.Weapon_R)
    {
        image.gameObject.SetActive(true);
        image.sprite = item.img[0];
    }
    else if (item.Type == EquipmentType.Shield)
    {
        Sheild.gameObject.SetActive(true);
        Sheild.sprite = item.img[0];
    }
    else if (item.Type == EquipmentType.Armor)
    {
        for (int i = 0; i < Armors.Length; i++)
            Armors[i].gameObject.SetActive(true);

        Armors[0].sprite = item.img[0];
        Armors[1].sprite = item.img[1];
        Armors[2].sprite = item.img[2];
    }
    else if (item.Type == EquipmentType.Plant)
    {
        for (int i = 0; i < Pant.Length; i++)
            Pant[i].gameObject.SetActive(true);

        Pant[0].sprite = item.img[0];
        Pant[1].sprite = item.img[1];
    }
    //부위별로 맞추어 이미지스프라이트 적용
    //아이템 이름 및 가격 등록
    info.text = item.itemName;
    priceTxt.text = "가격 : " + item.price;

    Refresh();//구매가 가능한지 판별
    //상점매니저의 구매시 다른카드들도 새로고침을 위한
    //델리게이트 함수에 등록
    ShopMgr.Inst.BuyEvent += Refresh;
}
```

```
참조 2개
void Refresh()//구매가 가능한지 판별
{//구매 버튼을 활성할지 안할지
    if (GameMgr.Inst.hero.Coin >= buyItme.price)
        buyBtn.interactable = true;
    else
        buyBtn.interactable = false;
}

참조 1개
public void BuyItem()
{ //구매 하기
    ShopMgr.Inst.BuyEqItem(buyItme);
}

참조 3개
public void OnPointerClick(PointerEventData eventData)
{//카드 선택시 아이템 설명 패널 On 이름과 능력치 정도
    GameMgr.Inst.ShowEqItemInfo(buyItme, transform.position);
}
```

부위별 맞는 이미지에 장착