The Darkest

개발 기간	2023.10.02 ~ 2024.03.15
개발 인원	1명 (개인 프로젝트)

개발 환경

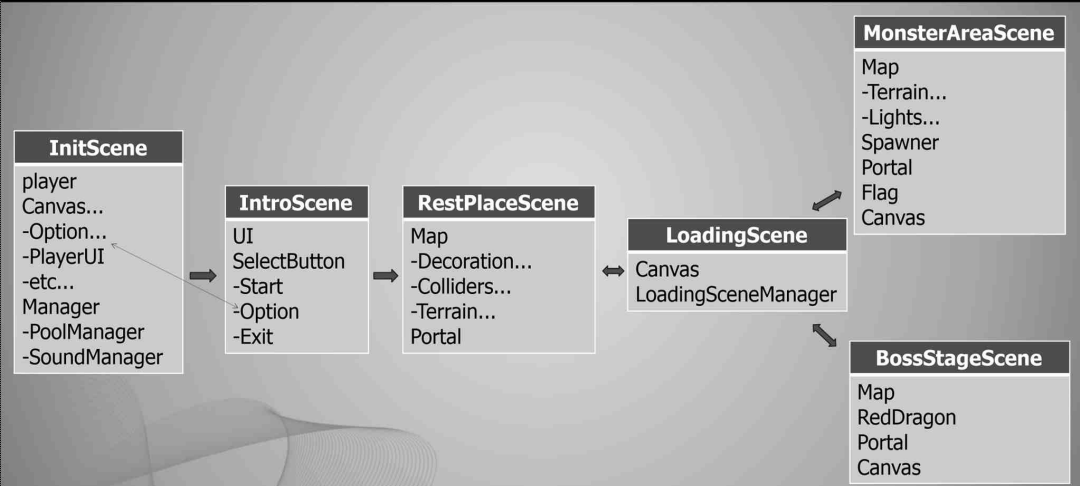
언어	C#
기술	Unity, Unity3D
작업Tool	UnityEngine, Git
Youtube	https://www.youtube.com/watch?v=yXY3ecVKgyY&t=197s

목차 a table of contents

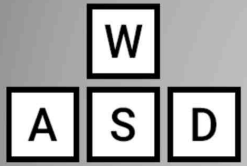
1. 어떤게임인가?

The Darkest는 다크소울 기반 RPG게임입니다.
아이템 수집, 레벨 업, 보스 전투 등의 요소가 있습니다.

2. 게임 Scene의 흐름



3. Player 작동 방식



이동키

달리기

장비 창 인벤토리 창

창닫기/옵션 창

조준

회전

조준 상태에서
발사

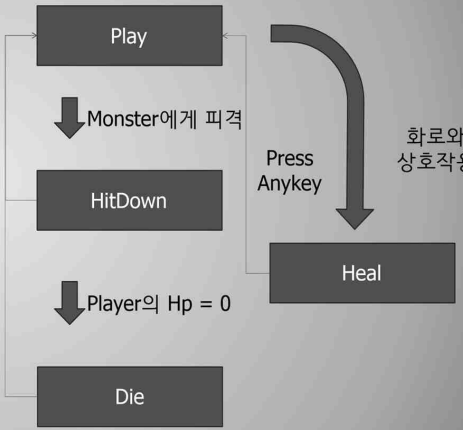
3. Player 작동 방식

모든 상태 변화가 생기는 Object들은
FSM패턴으로 구현하였습니다.

```
public PlayerState MyState = PlayerState.Play;

참조 7개
public void ChangeState(PlayerState s)// 상태가 바뀌면 한번 실행이 되는곳...

참조 1개
public void ProcessState()
{
    switch (MyState)
    {
        case PlayerState.Play:
            Play
        case PlayerState.Die:
            Die
        case PlayerState.HitDown:
            HitDown
        case PlayerState.Heal:
            Heal
            break;
    }
}
```



4. AI 작동방식

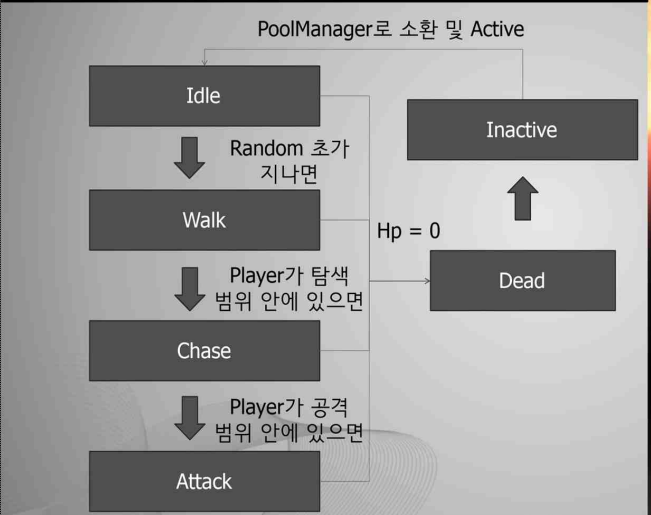
```
Unity 스크립트 | 참조 2개
public abstract class MonsterState : AnimatorAll
{
    public float Damage;
    public float Hp;
    public float moveSpeed;
    public string attackType;
    public float Exp = 0.0f;
    public LayerMask enemyMask;
    public Transform PlayerTransform;
    //공격 딜레이
    public static float attackDelay;
    //공격 범위
    public static float AttackRange;
    //공격 사거리
    public float AttackLength;
    public static PlayerController playerController;
    public Transform[] attackPos;
    참조 467개
    public enum MonsterStates
    {
        Idle, Walk, Fight, Chase, Attack, Dead, Scream, Sleep
    }
    Unity 메시지 | 참조 0개
    private void Awake()
    {
        playerController = GameObject.Find("Player").GetCo...<PlayerController>();
    }
    참조 2개
    public IEnumerator Attack()...
    참조 4개
    public abstract void monsterHit(float dam);
}
```

Monster들의 공통되는 부분은 중복 코드를 방지하기 위해 MonsterState Class를 추상화 하였습니다.

```
참조 2개
public IEnumerator Attack()
{
    for (int i = 0; i < attackPos.Length; i++)
    {
        Collider[] attack = Physics.OverlapSphere(attackPos[i].position, AttackRange, enemyMask);
        foreach (Collider coll in attack)
        {
            PlayerController playerController = coll.GetComponent<PlayerController>();
            if (playerController != null)
            {
                playerController.Attacked(Damage, attackType, this.transform);
                playerController.hit = true;
            }
        }
    }
    yield return null;
}
```

OverlapShpereCollider로 공격 범위 안에 PlayerCollider가 있으면 공격에 피해를 입습니다.

4. AI 작동방식


PoolManager로 소환 및 Active
Idle
Random 초가 지나면
Walk
Player가 탐색 범위 안에 있으면
Chase
Player가 공격 범위 안에 있으면
Attack
Hp = 0
Dead
Inactive

5. Object 관리

자주 생겼다 없어지는Object들은 PoolManager로 관리 하였습니다.
예를 들어 Monster를 3마리 까지 소환을 하고앞에 비활성화 된 Monster가
있으면 활성화 하고 소환하여 메모리 낭비를 최소화 했습니다.

Pooling 방식

```
//프리팹들을 보관할 변수
public GameObject[] prefabs;

//풀 담당을 하는 리스트
public List<GameObject>[] pools;

0 Unity 메시지 | 참조 0개
private void Awake()
{
    //배열 초기화
    pools = new List<GameObject>[prefabs.Length];
    //배열 안에 오브젝도 초기화
    for(int index = 0; index < pools.Length; index++)
    {
        pools[index] = new List<GameObject>();
    }
}

참조 1개
public void PoolManagerInit()
{
    pools = new List<GameObject>[prefabs.Length];
    for (int index = 0; index < pools.Length; index++)
    {
        pools[index] = new List<GameObject>();
    }
}

참조 1개
public GameObject Get(int index, float x, float z, float rotY, Transform par)
참조 1개
public GameObject Get(int index) // 최상위에 생성삭제를 해야하면 이곳에서
참조 1개
public GameObject Get(int index, Transform Pos) // 부모가 필요하면 이곳에서
```

Get()함수를 OverLoading을 사용하여 용도에 따라 사용할 수 있게 만들었습니다.

```
public GameObject Get(int index) // 최상위에 생성삭제를 해야하면 이곳에서
{
    GameObject select = null;

    // 선택한 풀의 놀고 (비활성화 된) 있는 게임오브젝트 접근
    foreach (GameObject item in pools[index])
    {
        // 발견하면 select 변수에 할당
        if (!item.activeSelf)
        {
            select = item;
            select.SetActive(true);
            break;
        }
    }

    if (!select) // 찾지 못했다면
    {
        // 새롭게 생성하고 select 변수에 할당
        select = Instantiate(prefabs[index]);
        pools[index].Add(select);
    }
    return select;
}
```

6. UI 작동 방식

UI GameObject는 list구현 하였습니다. list로 구현한 이유는 마우스로 누른 Object에 순서를 바꾸기 위해 list를 채택하였습니다.

```
Unity 메시지 | 참조 0개
void Update()
{
    if(list.Count != 0)
        Control.enabled = false;
    else
        Control.enabled = true;

    if(Input.GetKeyDown(KeyCode.Escape))
    {
        if (list.Count != 0)
        {
            //CloseUI Sound
            UISound(1);
            inven = false;
            stat = false;

            list[list.Count-1].SetActive(false);
            if(list[list.Count-1] == setting.transform.GetChild(0).gameObject)
            {
                playerController.enabled = true;
            }
            list.Remove(list[list.Count-1]);
        }
        else
        {
            UISound(0);
            SettingControll();
        }
    }
```

UI가 한개라도 켜져있으면 MouseControll 에 대한 제한을 두었습니다.

ESC를 누르면 가장 마지막에 켜졌거나 마우스로 누른 UI를 끌 수 있고 list가 null이면 Option이 켜질 수 있도록 구현 하였습니다.