내향형 트리오의 TRPG 프로젝트

매니저님밥4조
조윤진 이지윤 손형민

팀원소개

‘매니저님밥4조’는?

상세한 역할 분담은 노션 확인!
진짜 밥 사달라고 하지 않읍니다…

이지윤 (Lv.조장)

대문자 I

손형민 (Lv.조원)

극강의 I

조윤진 (Lv.조원)

확신의 I

<한끼줍쇼> 매니저님밥4조

게임소개

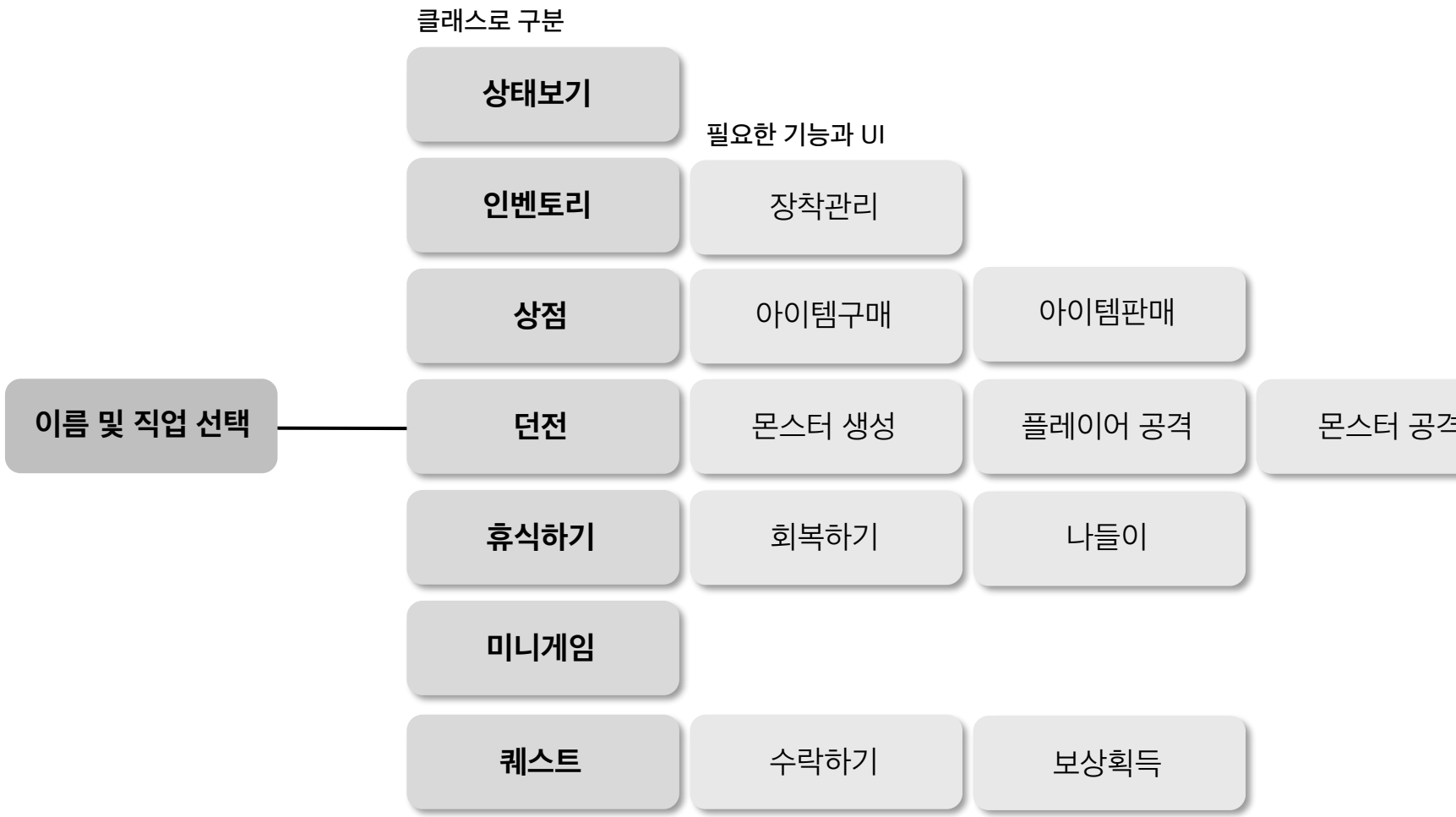
컨셉(concept)

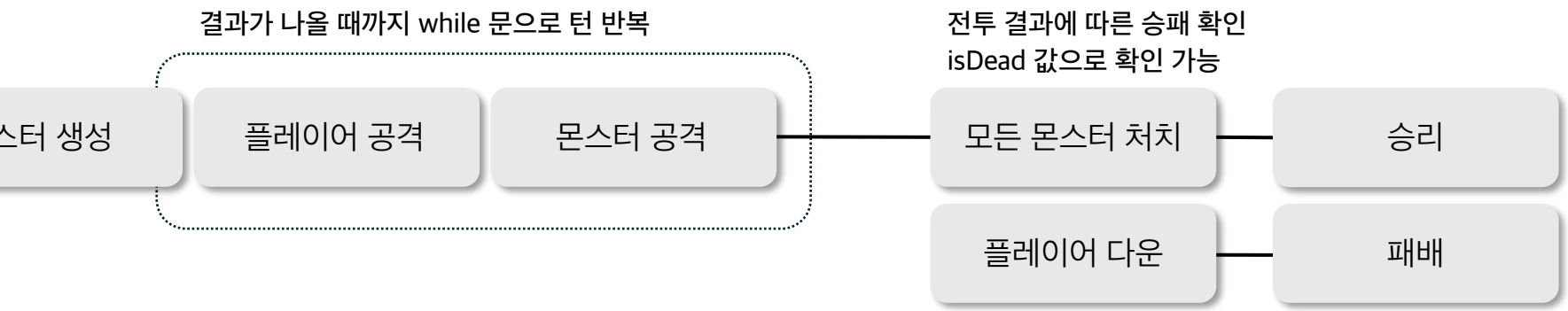
배고픈 내향형 I들이 한끼줍쇼하는 게임

2024년 10월 어느 날,
스파르타 던전에서 조용한 구걸 사태가 벌어지는데…

게임소개

와이어프레임


클래스로 구분
필요한 기능과 UI
이름 및 직업 선택
상태보기
인벤토리
장착관리
상점
아이템구매
아이템판매
던전
몬스터 생성
플레이어 공격
몬스터 공격
휴식하기
회복하기
나들이
미니게임
퀘스트
수락하기
보상획득


결과가 나올 때까지 while 문으로 턴 반복
스터 생성
플레이어 공격
몬스터 공격
전투 결과에 따른 승패 확인
isDead 값으로 확인 가능
모든 몬스터 처치
승리
플레이어 다운
패배

시연영상

실제로 구현된 <한끼줍쇼>는?

중심기능

핵심적인 로직은?

1.
던전 클래스

클론을 이용한
몬스터 대기열 생성

2.
스킬 구현

스킬 프리셋 생성

3.
퀘스트 클래스

진행도를 확인하기
위해 세가지 변수
이용

던전 클래스

던전에 입장하면
몬스터를 생성해요

```
public void EnterDungeon(string difficulty)
{
    //도망치기 변수 초기화
    isFleeing = false;

    // MonsterQueue 초기화
    MonstersQueue = new List<Monster>();

    //스킬 생성
    SkillList = addSkills(player.role, ref SkillList);

    // 몬스터 생성
    MonstersQueue = makeMonster(ref MonstersQueue, difficulty);
```

던전 클래스

For 문으로 몬스터Q에
랜덤 생성된 몬스터를
할당해요

```
public List<Monster> makeMonster(ref List<Monster> MonstersQueue, string Difficulty)
{
    int numberOfDraws = rand.Next(1, 5); // 뽑고 싶은 몬스터의 수를 설정 (1~4사이)
    if (Difficulty == "Easy")
    {
        for (int i = 0; i < numberOfDraws; i++)
        {
            int monsterIndex = rand.Next(MonsterPreset.baseMonster.Count);
            Monster tempMonster = MonsterPreset.baseMonster[monsterIndex].Clone();
            MonstersQueue.Add(tempMonster); // MonstersQueue에 새로운 몬스터 추가
        }
        return MonstersQueue;
    }
```

던전 클래스

For 문으로 몬스터Q에
랜덤 생성된 몬스터를
할당해요

▶ 몬스터 프리셋?

```
public class MonsterPreset
{
    public static List<Monster> baseMonster = new List<Monster>()
    {
        new Monster("김록기", 5, 4, 2),
        new Monster("안혜린", 6, 7, 10),
        new Monster("강채린", 7, 10, 5)
    };
    public static List<Monster> NormalMonster = new List<Monster>()
    {
        new Monster("보다 강해진 김록기", 8, 8, 5),
        new Monster("보다 강해진 안혜린", 9, 14, 15),
        new Monster("보다 강해진 강채린", 10, 20, 10)
    };
    public static List<Monster> HardMonster = new List<Monster>()
    {
        new Monster("눈부시게 강한 김록기", 15, 20, 10),
        new Monster("눈부시게 강한 안혜린", 20, 30, 20),
        new Monster("눈부시게 강한 강채린", 25, 50, 15)
    };
    public static List<Monster> EpicMonster = new List<Monster>()
    {
        new Monster("김록기(대장)", 1000, 100, 20),
        new Monster("안혜린(대장)", 1500, 150, 100),
        new Monster("강채린(대장)", 2000, 200, 50)
```

던전 클래스

For 문으로 몬스터Q에
랜덤 생성된 몬스터를
할당해요

```
public List<Monster> makeMonster(ref List<Monster> MonstersQueue, string Difficulty)
{
    int numberOfDraws = rand.Next(1, 5); // 뽑고 싶은 몬스터의 수를 설정 (1~4사이)
    if (Difficulty == "Easy")
    {
        for (int i = 0; i < numberOfDraws; i++)
        {
            int monsterIndex = rand.Next(MonsterPreset.baseMonster.Count);
            Monster tempMonster = MonsterPreset.baseMonster[monsterIndex].Clone();
            MonstersQueue.Add(tempMonster); // MonstersQueue에 새로운 몬스터 추가
        }
        return MonstersQueue;
    }
```

던전 클래스

Clone은 객체를
동일하게 복사해서
새로운 객체를 만드는 기능

```
public Monster(string name, int hp, int attackPower, int LV)
{
    this._name = name;
    this._hp = hp;
    this._atkPower = attackPower;
    this._level = LV;

    Init();
public void Init()
{
    name = _name;
    hP = _hp;
    atkPower = _atkPower;
    level = _level;
}

public Monster Clone()
{
    return new Monster(_name, _hp, _atkPower, _level);
```

던전 클래스

While 문으로
몬스터나 플레이어가
전멸할 때까지 전투를
반복해요

```
public void PlayBattle()
{
    // 플레이어 혹은 몬스터집단 중 한 쪽이 전멸할때까지 반복
    while (!MonstersQueue.All(x => x.isDead) && !player.isDead)
    {
        // 전투 UI 띄우기
        DisplayBattleUI(MonstersQueue, player);

        //도망치기 선택 경우 던전입장 UI로 return
        if (isFleeing)
        {
            Console.Clear();
            Console.WriteLine("당신은 쫄아서 튀었다..!");
            Console.ReadKey();
            return;
        }
```

던전 클래스

While 문으로
몬스터나 플레이어가
전멸할 때까지 전투를
반복해요

```
        // 몬스터 전멸일 경우
        if (MonstersQueue.All(x => x.isDead))
        {
            DisplayBattleResult(MonstersQueue, player);
        }

        // 연속된 몬스터 공격 Ui 띄우기
        DisplayEnemyAttack();

        // 플레이어 사망일 경우
        if (player.isDead)
        {
            DisplayBattleResult(MonstersQueue, player);
        }
    }
}
```

스킬 클래스

스킬 클래스에서
스킬 세부사항을 넣을
틀을 만들어요

```
public class Skill
{
    참조 4개
    public string Name { get; set; }  // 스킬 이름
    참조 5개
    public int Damage { get; set; }   // 스킬의 데미지
    참조 3개
    public bool IsAoE { get; set; }   // 광역딜 여부 (true면 광역딜, false면 단일딜)

    참조 2개
    public string Description { get; set; }
    참조 5개
    public int ConsumeMP {  get; set; }
    // 생성자
    참조 3개
    public Skill(string name, int damage, bool isAoE, int consumeMP, string description)
    {
        Name = name;
        Damage = damage;
        IsAoE = isAoE;
        ConsumeMP = consumeMP;
        Description = description;
    }
```

스킬셋 클래스?

```
public static class SkillSet
{
    // 수강생 스킬 리스트
    public static List<Skill> StudentSkills = new List<Skill>()
{
    new StudentSkill("깨물고 싶어! 수강생의 폭풍 애교!♥", 50, false, 10, "폭풍애교로 매니저님 한 명을 함락시킵니다. 공략지수 50"),  // 단일딜
    new StudentSkill("귀여움 폭발 수강생의 손하트♥ ", 30, true, 15, "손하트로 모든 매니저님 마음에 하트 도장을 찍습니다. 공략지수 30")   // 광역딜
};

    // 튜터 스킬 리스트
    public static List<Skill> TutorSkills = new List<Skill>()
{
    new TutorSkill("매혹적인 튜터의 윙크♥", 70, false, 10, "윙크로 그의 마음을 흔들어봅시다. 공략지수 70"),  // 단일딜
    new TutorSkill("심쿵유발 튜터의 사랑의 눈빛♥", 40, true, 15, "그윽한 눈빛으로 모든 매니저님의 하-또를 뺏아옵시다 . 공략지수 40") // 광역딜
};

    // 매니저 스킬 리스트
    public static List<Skill> ManagerSkills = new List<Skill>()
{
    new ManagerSkill("앙큼폭스! 매니저의 볼콕!", 60, false, 10, "너무 앙큼하다! 저 손가락이 내 입덕계기야! 매니저님 한 분께 공략지수 60"),  // 단일딜
    new ManagerSkill("반전매력! 매니저의 깜찍 댄스♥", 50, true, 15, "아니 이런 매력이! 넘어가지 않을 수가 없다! 모든 매니저님께 공략지수 50")    // 광역딜
};
}
```

스킬 클래스

스킬 리스트를 생성해요

```
참조 1개
public List<Skill> addSkills(string role, ref List<Skill> Skills)
{
    Skills.Clear();

    switch (role)
    {
        case "수강생":
            foreach (Skill skill in SkillSet.StudentSkills) Skills.Add(skill);
            return Skills;
        case "튜터":
            foreach (Skill skill in SkillSet.TutorSkills) Skills.Add(skill);
            return Skills;
        case "매니저":
            foreach (Skill skill in SkillSet.ManagerSkills) Skills.Add(skill);
            return Skills;
        default:
            foreach (Skill skill in SkillSet.StudentSkills) Skills.Add(skill);
            return Skills;

    }
}
```

▶ 던전 입장 함수를 다시 봅시다!

던전 클래스

던전에 입장하면
몬스터를 생성해요

```
public void EnterDungeon(string difficulty)
{
    //도망치기 변수 초기화
    isFleeing = false;

    // MonsterQueue 초기화
    MonstersQueue = new List<Monster>();

    //스킬 생성
    SkillList = addSkills(player.role, ref SkillList);

    // 몬스터 생성
    MonstersQueue = makeMonster(ref MonstersQueue, difficulty);
```

퀘스트 클래스

퀘스트의 내용에 따라
자식클래스 생성해요

```
참조 3개
public class DungeonClearQuest ...
```

▶ 일정 횟수 이상 던전을 클리어하세요!

```
참조 3개
public class InventoryQuest ...
```

▶ 인벤토리에 특정 아이템을 장착하세요!

퀘스트UI

퀘스트UI에서 퀘스트
성공여부를 항상 체크해요

```
참조 1개
public void DisplayQuestUI()
{

    bool isSelected = false;

    while (isSelected == false)
    {
        foreach (KeyValuePair<int, Quest> i in quests)      //모든 퀘스트 성공여부 체크
        {
            i.Value.CheckComplete(player);
        }
        Console.Clear();
        string text =
           $"{ShowQuests()}\r\n" +
           "0. 나가기 \r\n\r\n" +
            "원하시는 퀘스트를 선택해주세요.\r\n";
        Console.WriteLine(text);

        string input = Console.ReadLine();
        if (!int.TryParse(input, out int inputIDX))
        { Console.WriteLine("잘못된 입력입니다."); continue; }
```

퀘스트 클래스

각 클래스에서 성공여부를
체크하는 함수를 넣었어요

```
public override void CheckComplete(Player player)
{
    CurrentClear = dungeon.dungeongClearCount;

    if (CurrentClear >= RequiredClears)
    {
        CurrentClear = RequiredClears;
        readyToClear = true;
```

```
public override void CheckComplete(Player player)
{
    foreach(KeyValuePair<Item, int> i in player.inventory.inventory)
    {
        if(i.Key.name == targetItem && i.Value > 0 && player.inventory.equipList.Contains(i.Key))
        {
            readyToClear = true;
```

퀘스트UI

현재 퀘스트가
어떤 상태인지 확인해요

```
참조 1개
string ShowQuests()
{
    string ShowAllQuest = "";
    int number =1;
    foreach (KeyValuePair<int, Quest> i in quests)
    {
        string isRunning = "";
        if (!i.Value.isAccepted && !i.Value.readyToClear && !i.Value.isCleared)
        {
            isRunning += "";
            //아직 받지 않은 상태
        }
        else if (i.Value.isAccepted && !i.Value.readyToClear && !i.Value.isCleared)
        {
            isRunning += "(진행중)";
            //받고 진행중인 상태
        }
        else if (i.Value.isAccepted && i.Value.readyToClear && !i.Value.isCleared)
        {
            isRunning += "(보상 획득 가능)";
            //완료버튼 누르면 ok인 상태
        }
        else if (i.Value.isAccepted && i.Value.readyToClear && i.Value.isCleared)
        {
            isRunning += "(완료)";
            //이미 완료한 상태
            Console.ForegroundColor = ConsoleColor.Red;
```