TacticRPG

KMU2025 capstone design team 43

택틱시스템을 적용한 수집형 RPG 제작

김동휘 , 강문정 , 유민석 , 최치원

→ 개발 배경

기존 수집형 **RPG**의 특징

01 **전투의 단조로움**

자동 전투는 반복적, 턴제는 느리고 번거로움

02 **개발 기간의 중요성**

수집형 RPG의 경우 지속적이고 빠른 수집 요소 추가가 매우 중요

03 **맵 시스템의 전략성 부족**

맵을 점령하는 전략적 의미가 약함

04 **낮은 영웅 수집 접근성**

영웅 수집을 과금에 의존하는 경우가 많음

Key Systems & Design Intent

각 시스템은 다음과 같은 설계목표를 가진다.

- **전투의 단조로움**을 줄이기 위한 **TacticSystem**
- **애니메이션** 컨트롤러 형태를 통일시키기 위한 **AnimationSystem**
- **개발 기간을 단축**하면서 다양한 캐릭터를 만들기 위한 **TacticSystem**
- **맵을 점령하는 의미**를 추가하기 위한 **MapSystem**
- **낮은 영웅 수집 접근성**을 해결하기 위한 **상점시스템**과 **특별한 적 시스템**

BattleScene

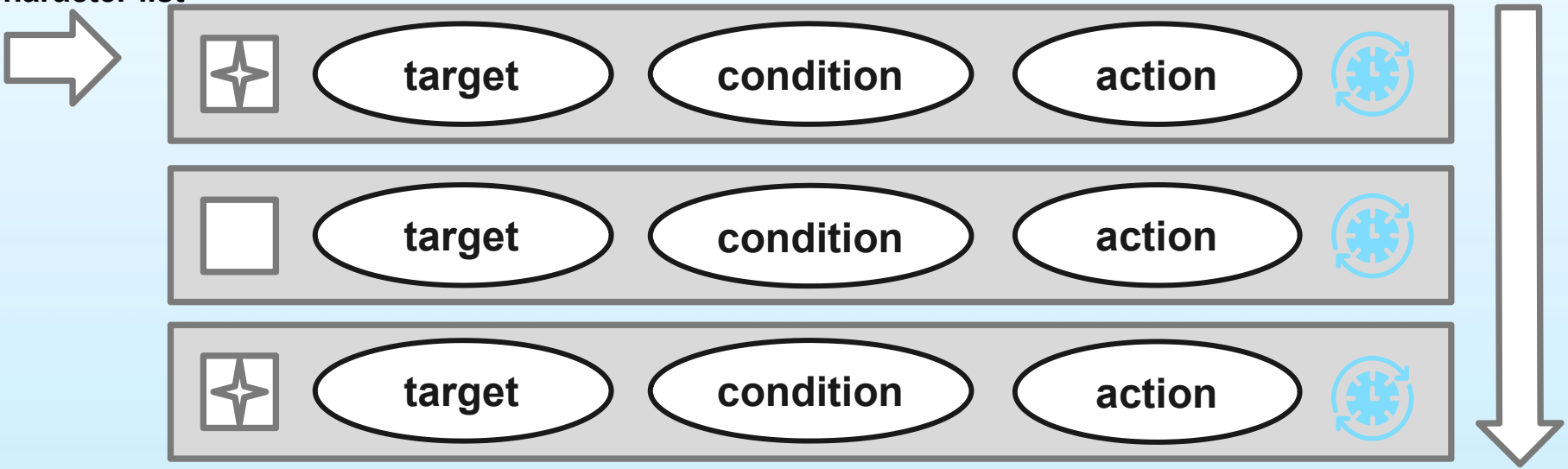
01

Tactic System

전투의 단조로움을 줄이기 위한 TacticSystem

→ Tactic System

character list



* **Character**의 모든 행동은 **TacticSystem**을 통해 결정
* 한 **Character**는 하나의 **TacitcSystem**을 가짐.

→ Tactic’s Component

Target

target

Ally, Self, Enemy 등 1차적으로 대상을 분류한다.

Condition

condition

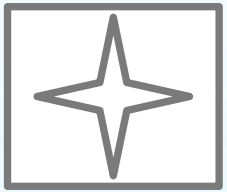
HP, 거리 등 조건을 기준으로 대상 리스트를 필터링한다.

Action

action

필터링된 대상에게 공격, 회복, 버프 등의 행동을 수행한다.
쿨다운 변수를 가지고 있다.

→ Tactic’s Component



Enable

해당 Tacitc을 실행할 지 여부를 결정한다.

TacticCoolDown

해당 Tactic의 쿨타임을 나타낸다. Action마다 쿨타임 변수를 가지고 있으며, 실행한 Action과 동일한 Action을 지닌 Tactic은 모두 해당 Action의 쿨다운을 적용

→ Tactic System Cycle

- Character는 하나의 TacitcSystem을 가짐.
- TacticSystem은 Tacitc의 리스트를 가짐.
- Character의 GlobalCooldown마다 TacticList를 순회하며 조건에 맞는 Action을 실행
- 실행한 Action과 동일한 Action을 지닌 Tactic은 모두 해당 Action의 쿨다운을 적용

실행조건은 다음과 같다.

- Tactic의 Enable이 True
- Tactic의 쿨타임이 끝남.
- target에서 필터링한 객체가 있음.
- condition에서 필터링한 객체가 있음.
- action이 수행 가능

ProtoType Example

캐릭터 마우스호버링 시 아웃라인 발생,
우클릭 시 해당 캐릭터의 TacticUI팝업

- 버튼으로 Enable 수정.
- 드랍다운으로 택틱 컴포넌트 수정
- 드래그 드랍으로 우선순위 수정
- 마지막 택틱은 Enemy Nearlest Attack
- 마지막 택틱은 드래그,편집 불가
- 몬스터의 경우는 모든 택틱이 사전 정의 및 드래그 편집 불가.

Tactic System Advantages

- 전투중 택틱을 수정할 수 있기 때문에 전략적 요소 증가 및 사용자 참여형 자동전투 시스템 가능
- 몬스터와 영웅(플레이어의 캐릭터)의 설계 방식이 동일하기 때문에 개발 기간 단축
- 몬스터에 사용한 택틱 컴포넌트를 영웅에 재사용 가능
- 같은 영웅 3D 모델이더라도 보유 택틱 컴포넌트를 다르게 하면 다른 영웅처럼 활용 가능

BattleScene

02

Battle System

전반적인 전투관련 BattleSystem 설계

→ BattleManager

MonsterWavePreset

몬스터 웨이브의 정보를
담고있는 자료형
몬스터의 종류와 위치를 가짐

Heroes

문자열을 통해
InitializeHeroes함수에서
prefab참조로 생성

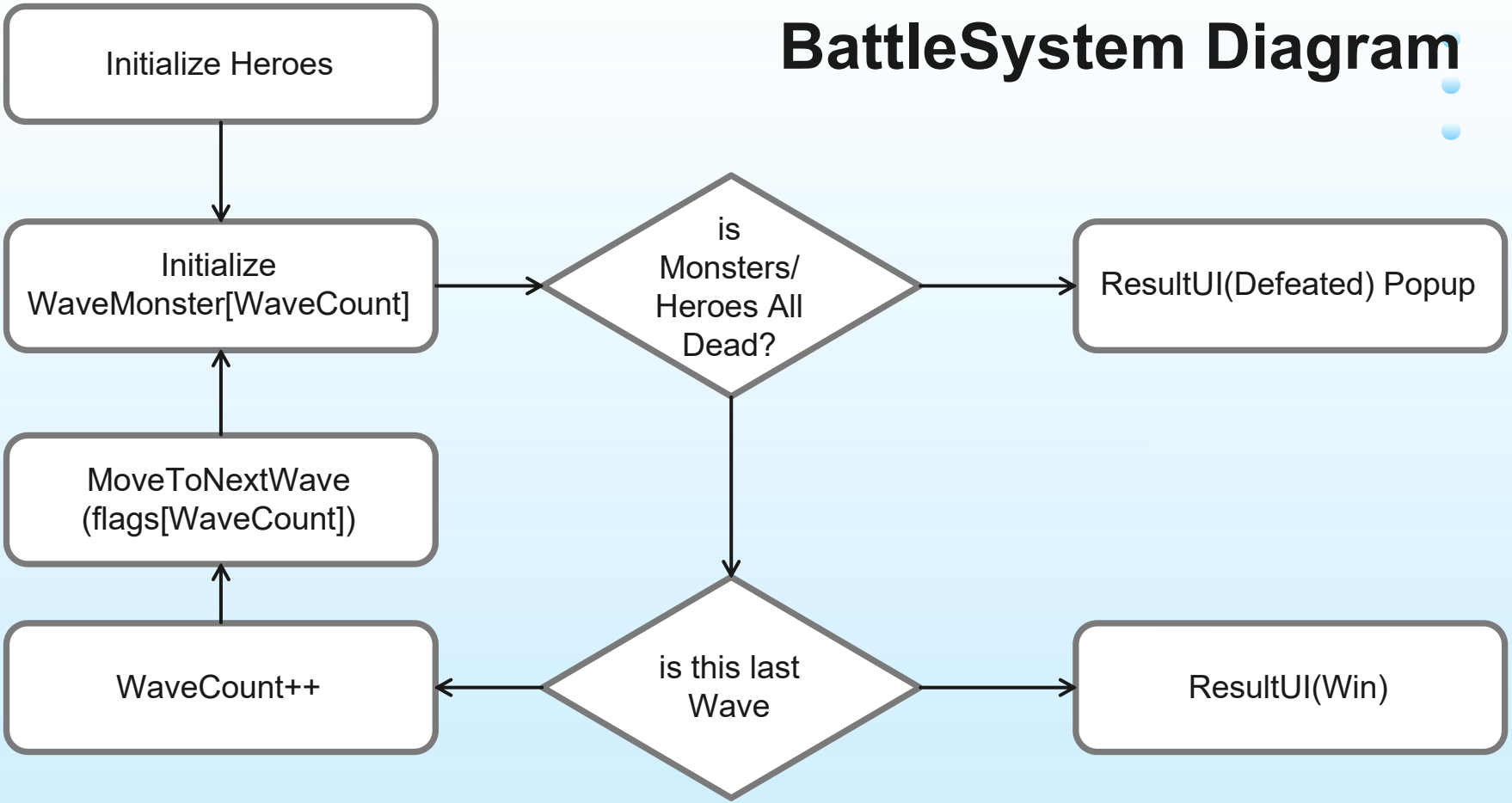
Flag&Grid

BattleField에서 받은
Flag위치정보와 MapScene
에서 받은 Grid정보로 원하는
위치에 원하는 영웅 배치
Monster도 Flag에 대한
상대위치에 배치

ResultUI

승리 혹은 패배를 나타내는 UI
MapScene으로 돌아가는
버튼을 지님

BattleSystem Diagram



BattleScene

03

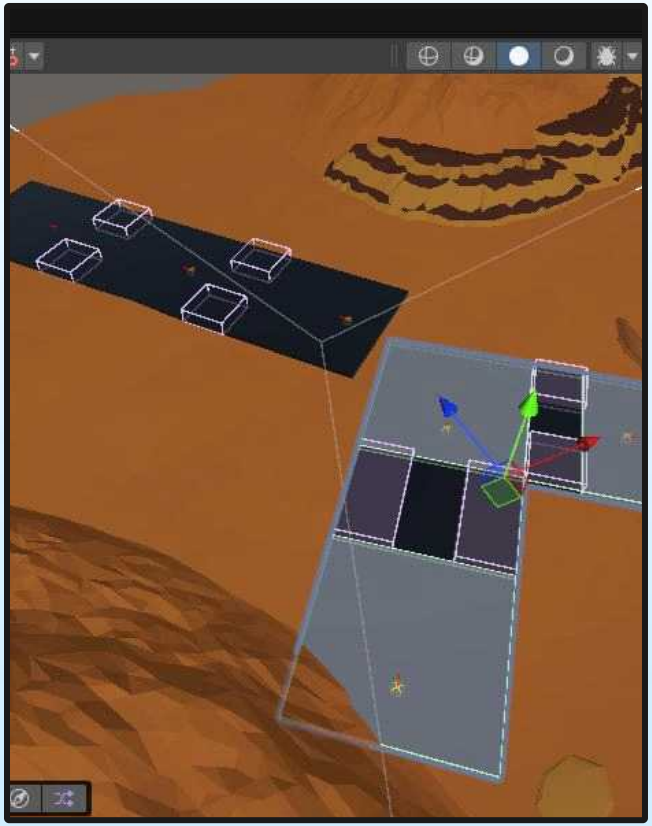
Field System

전투가 벌어지는 장소를 Field라고 지칭

BattleField 와 EnvironmentField를 분리

- BattleField는 실질적인 전투에 대한 길찾기, 충돌과 관련된 물리적 처리를 담당
- BattleField는 FieldManager에 의해 전투 시작 시 투명화
- EnvironmentField는 단순한 환경을 나타내고 물리적 처리는 하지 않음

=> 최적화 및 배경환경 재사용 가능!!



FieldManager

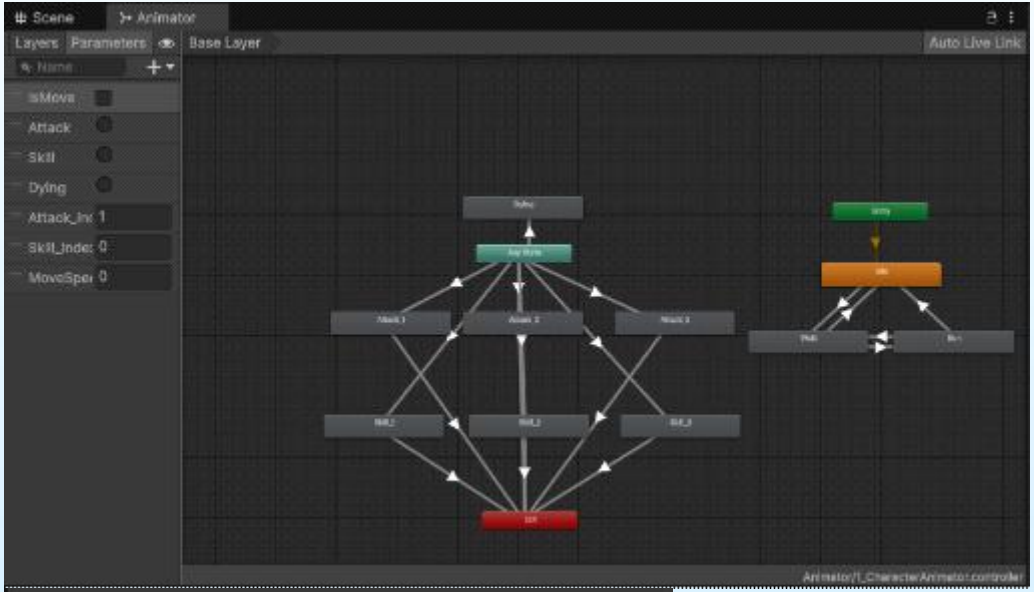
- FieldType, EnvironmentPrefab과 BattleFieldPrefab 그리고 그 둘의 위치정보를 가진 “FieldSetData”의 리스트를 가지고 있음
- MapNode에서 FieldType을 넘겨받음
- FieldSetDataList에서 FieldType에 맞는 Field를 생성
- BattleField에 NavMeshSurface를 통해 길찾기 정보 생성
- BattleField를 Material 변경으로 투명화

BattleScene

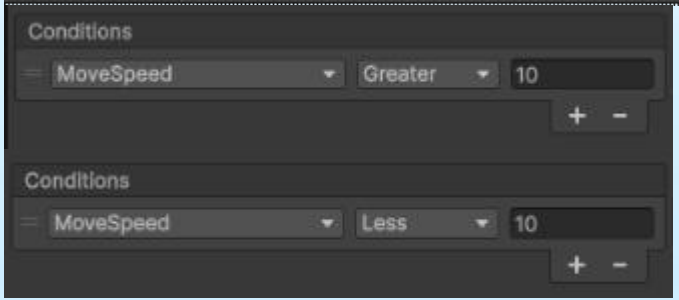
04

Animation

character animation control


Scene
Animator
Layers
Parameters
Base Layer
Auto Live Link
isMove
Attack
Skill
Dying
Attack_Inc
Skill_Index
MoveSpee
Animator/1_CharacterAnimator.controller


Animator
1_CharacterAnimator
ArcherAnimator
AssassinAnimator
GoblinAnimator
GoblinAnimator_Archer
GoblinAnimator_Paladin
GoblinAnimator_Wizard
PaladinAnimator
WizardAnimator


Conditions
MoveSpeed
Greater
10
Conditions
MoveSpeed
Less
10

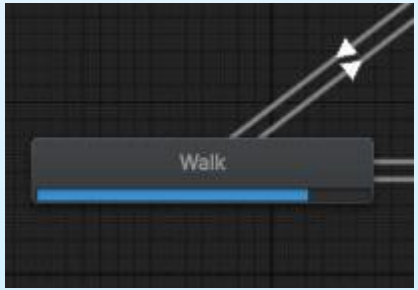
Paladin Animator (Animator Override Controller)

Open

Addressable

Controller 1_CharacterAnimator

Original	Override
Attack_1	Paladin_Attack_1
Attack_2	Paladin_Attack_2
Attack_3	Paladin_Attack_3
Breathing Idle	Paladin_Idle
Dying	Paladin_Dying
Fast Run	Paladin_Run
Walking	Paladin_Walk



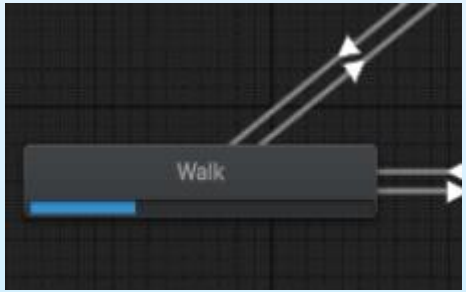
Archer Animator (Animator Override Controller)

Open

Addressable

Controller 1_CharacterAnimator

Original	Override
Attack_1	Archer_Attack_1
Attack_2	Archer_Attack_2
Attack_3	Archer_Attack_3
Breathing Idle	Archer_Idle
Dying	Archer_Dying
Fast Run	Archer_Run
Walking	Archer_Walk



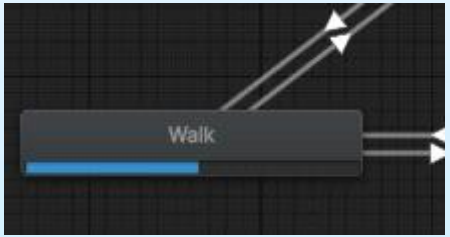
Goblin Animator (Animator Override Controller)

Open

Addressable

Controller 1_CharacterAnimator

Original	Override
Attack_1	Goblin_Attack_1
Attack_2	Goblin_Attack_2
Attack_3	Goblin_Attack_3
Breathing Idle	Goblin_Idle
Dying	Goblin_Dying
Fast Run	Goblin_Run
Walking	Goblin_Walking




Audio Manager (Script)
Script
AudioManager
Bgm Source
BGM (Audio Source)
Sfx Source
SFX (Audio Source)
Bgm Clip
BGM_1
Sfx List
2
Element 0
Type
Foot Step
Clip
4
Element 0
Footsteps_Sand_Run_01
Element 1
Footsteps_Sand_Jump_Land_01
Element 2
Footsteps_Sand_Jump_Land_02
Element 3
Footsteps_Sand_Jump_Land_03
Element 1
Type
Attack
Clip
5
Element 0
Battleaxe1
Element 1
Battleaxe2
Element 2
Battleaxe3
Element 3
Battleaxe4
Element 4
Battleaxe5

Map Scene

MapScene

→ Managers

1. GameManager
    1. 맵 생성 및 게임 흐름 전체 제어
   UiManager
    1. UI 열기/닫기 및 스택 기반 UI 상태 관리
   DataManager
    1. 유닛, 지역 등 데이터 로드 및 저장
   ResourceManager
    1. 게임 내 object 동적 생성/파괴 관리

   씬 내에서 쉽게 접근 가능한 **싱글톤 패턴**으로 구현됨

→ Map

절차적 생성

Base Image 생성
- Node의 생성 범위를 결정

Node 생성 및 필터링
- Local 정보를 가지고 있는 object

Rode 연결
- Node들을 연결한 도로 – 들로네 삼각분할법

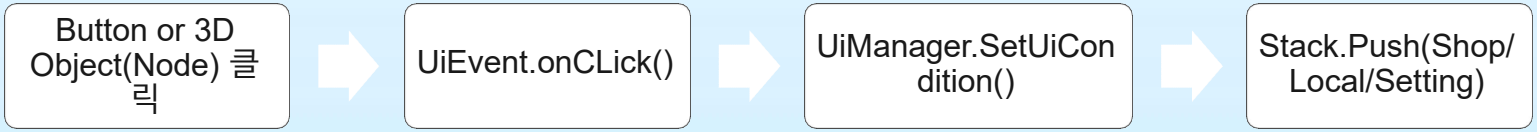
Env 생성
- Node와 충돌시켜서 Node의 환경을 결정

매 게임마다 다른 맵을 생성해 **유저에게 매번 새로운 경험** 제공

→ Ui

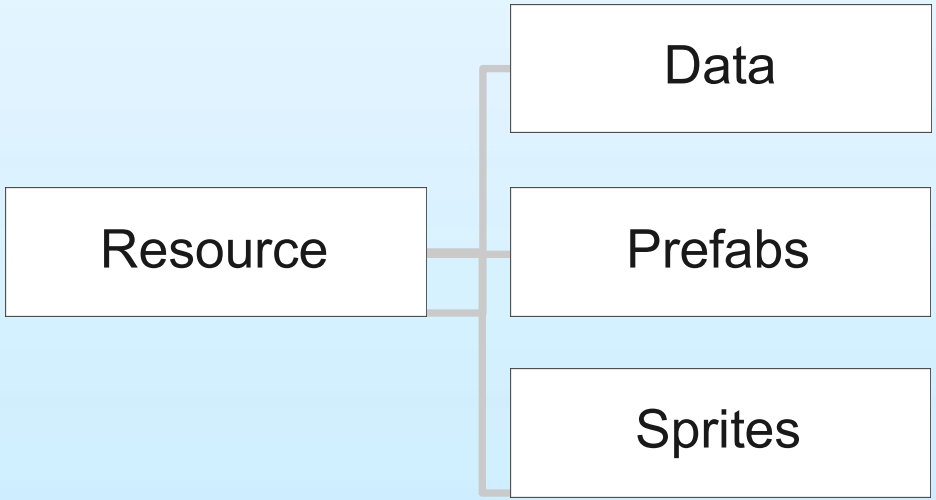
Default , Shop , Local , Setting…

Stack 구조로 Ui 상태 관리
Default Ui는 항상 유지하면서 나머지 Ui는 그 위에 쌓이도록 설계



최상단 ui만 상호작용 가능하고 하위 ui들은 block
Ui가 열린 상태에서는 3D 오브젝트(Node) 클릭 방지

→ Data , Resource



리소스를 코드에서 직접 경로로
불러올 수 있어
데이터 접근이 빠르고 구조화됨

```
GameObject prefab =
Resources.Load<GameObject>($"
Prefab/{path}");
```

Thanks!

Do you have any questions?

ckw0655@google.com
+82 10 6243 1142