

3D RPG 개발

김경훈

1. 게임 설명

2. 개발환경

3. 캐릭터구조

- a. 추상화를 활용한 다형성 구조
- b. 플레이어(FSM-유한상태머신) 구현내용
- c. 적(행동 트리) 구현 내용

4. 저장 및 불러오기

5. 셰이더

- a. RimLight
- b. Dissolve

6. 렌더링 최적화 작업

1. 게임 설명

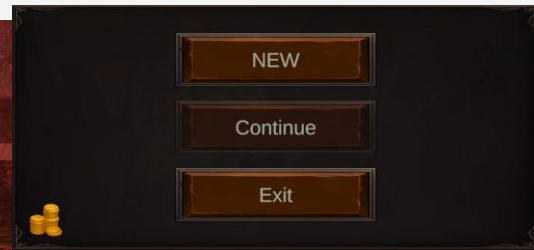
3D 기반의 3인칭 시점 RPG

장르: 3인칭 RPG

플랫폼: PC

플레이 방법

- 총 3개의 스테이지로 구성
- 무한히 스폰되는 적을 처치하여 레벨업을 통해 스탯(공격력, 방어력, 체력)을 상승시킬 수 있다.
- 각 스테이지마다 달성 해야할 레벨조건을 만족해야 다음 스테이지로 넘어갈 수 있다.

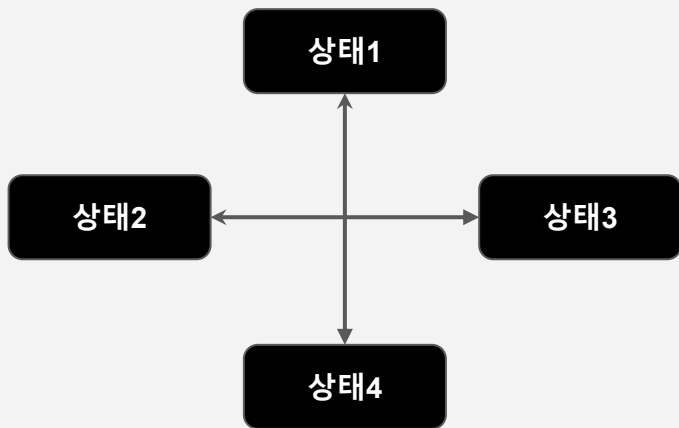


2. 개발 환경

유니티 버전	Unity Editor 22.3.44f1
스크립트 에디터	Visual Studio 2022 Community
모델 및 애니메이션 출처	Mixamo
에셋출처	Asset Store(Cemetery Kit, RPG_FPS_game_Asset, Skybox Series Free, Viking Village)

3. 캐릭터 구조

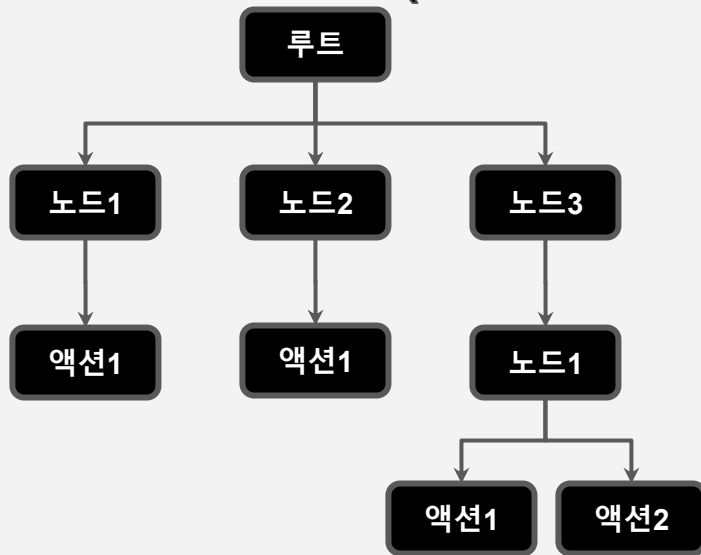
FSM-유한상태머신



- 각 상태의 전환이 명확
- 입력에 빠르게 반응

사용자가 “직접 컨트롤” 하는 캐릭터 적합

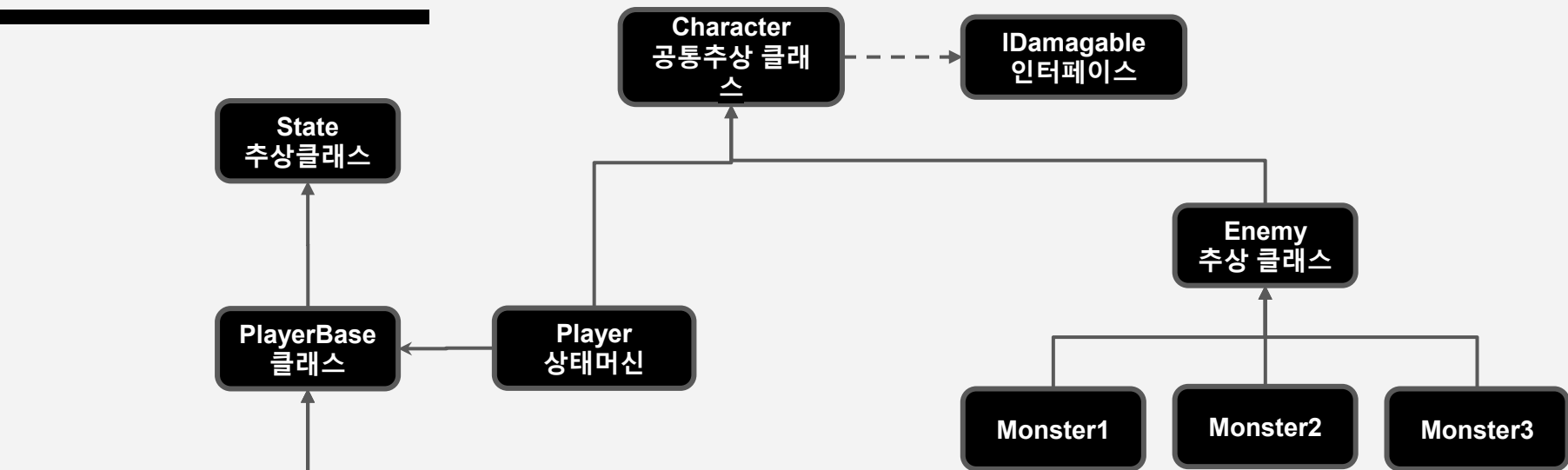
Behavior Tree(행동트리)



- 우선 순위 기반의 행동을 수행
- 유연한 의사결정 판단

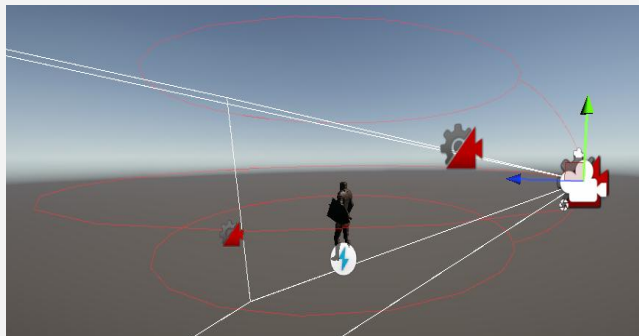
“스스로 행동” 하는 적 캐릭터에게 적합

3. 캐릭터 구조(추상화)



- 공통된 인터페이스 또는 부모를 상속받아 유닛간의 **데미지 교환 로직을 단일화** 시키도록 설계.
- 내부 데미지 계산 로직이나 스탯 처리 방식은 **개별 클래스에서 다르게 구현함**으로써 **다형성을 활용**

3. 플레이어(FSM)



FreeLook (자유시점 상태)

- Cinemachine FreeLook 카메라로 캐릭터의 3개 y축에서 360도 회전으로 주변 시야 확인



TargetLook(타겟 상태)

- Tab키 입력으로 타겟과 자유시점 상태 전이
- Cinemachine TargetGroup 카메라로 타겟을 중심축으로 고정시켜 확인
- 타겟이 된 적은 파란계열의 RimLight 효과



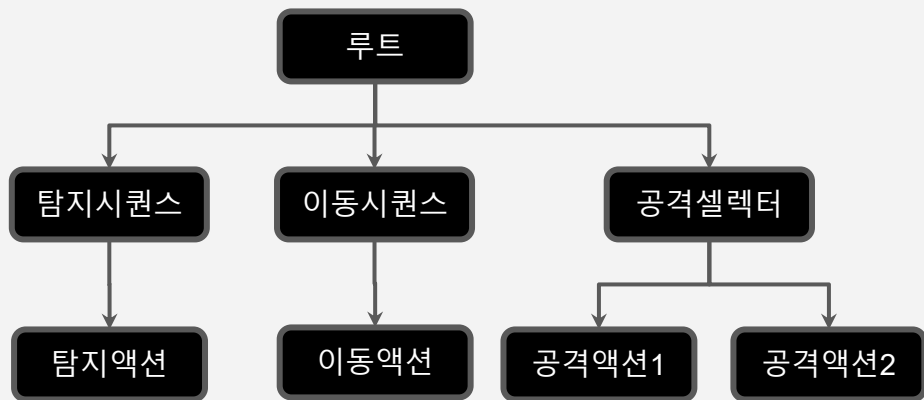
상태전이



Dead(죽음 상태)

- 체력이 0이되면 어느 상태에서도 전환되며 **Dissolve** 효과로 플레이어가 점차 사라지는 효과연출

3. 적 (행동트리)

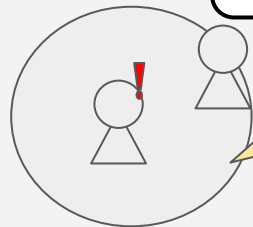


적 AI는 탐지 → 이동 → 공격의 흐름으로 구성되어,
실제 행동처럼 자연스럽게 순차 실행

탐지 성공 시 → 목표 지점으로 이동 → 도달 시 공격 수행

탐지

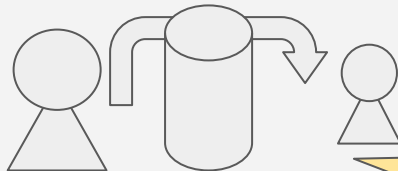
플레이어와의 거리가 사전에 설정해둔 탐지 거리보다 가까우면 플레이어를 탐지한 것으로 판단.
(거리 비교는 $y=0$ 으로 평면거리의 벡터 크기로 연산)



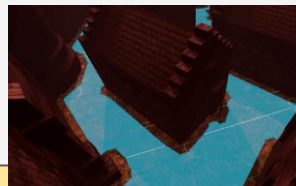
DisToPlayer = (플레이어 위치 - 내 위치)
DisToPlayer.y = 0
DetectedRange >= DisToPlayer.magnitude

탐지거리 > 플레이어와의 거리

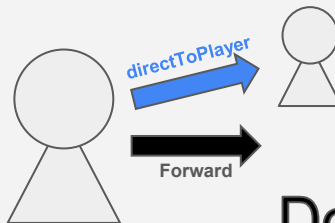
이동



탐지성공 시
AI Navigation으로 적이 맵 지형을 돌아서 이동할 수 있게 설정



공격로직

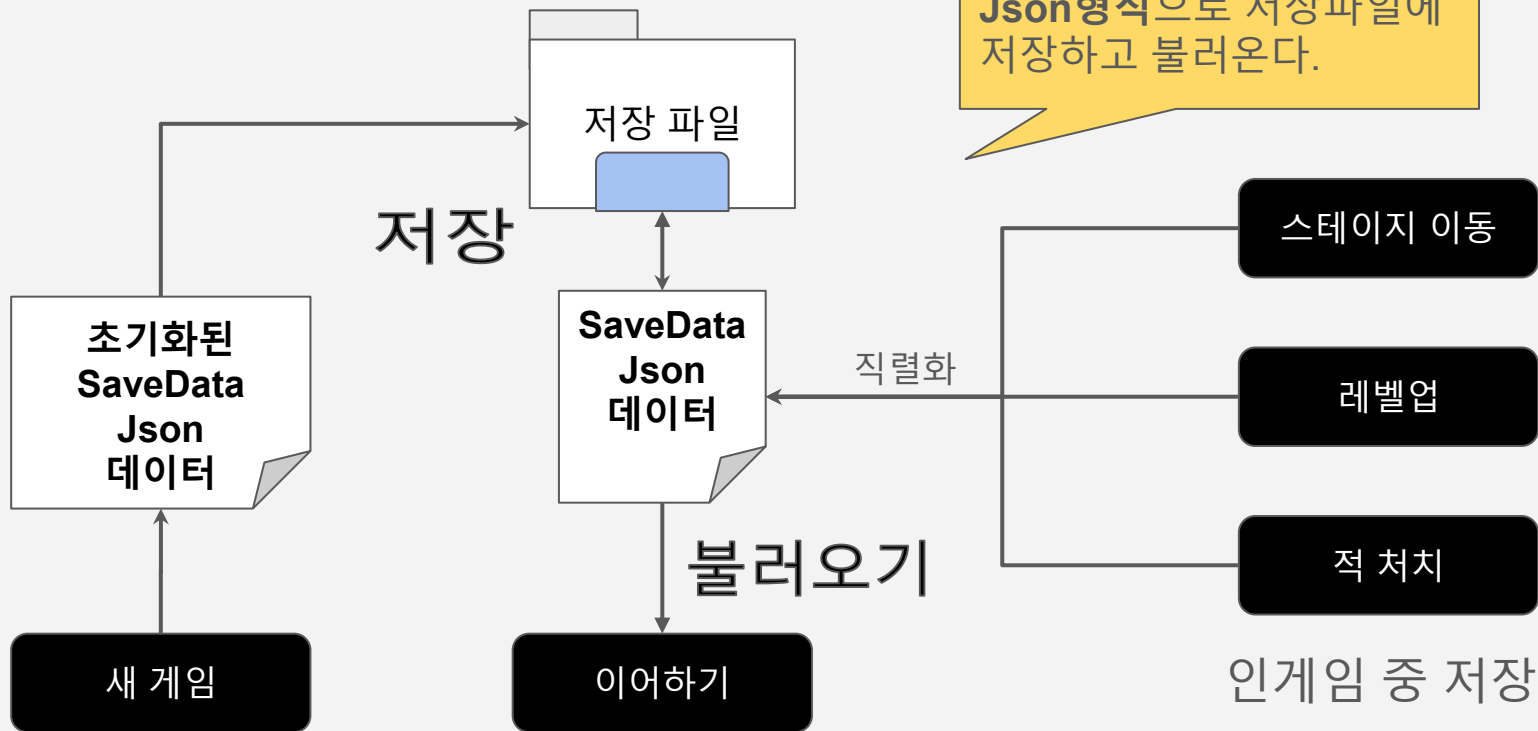


- 회전할때 적이 바라보는 방향벡터와 플레이어와의 방향벡터를 $\text{dot}(\text{내적})$ 연산으로 0.9보다 크면 대상을 바라보는 것으로 판단 후 공격 수행
- 회전 시, Quaternion slerp로 짐벌 락 현상(두 벡터가 겹치며 축이 사라지는 현상) 방지

Dot > 0.9

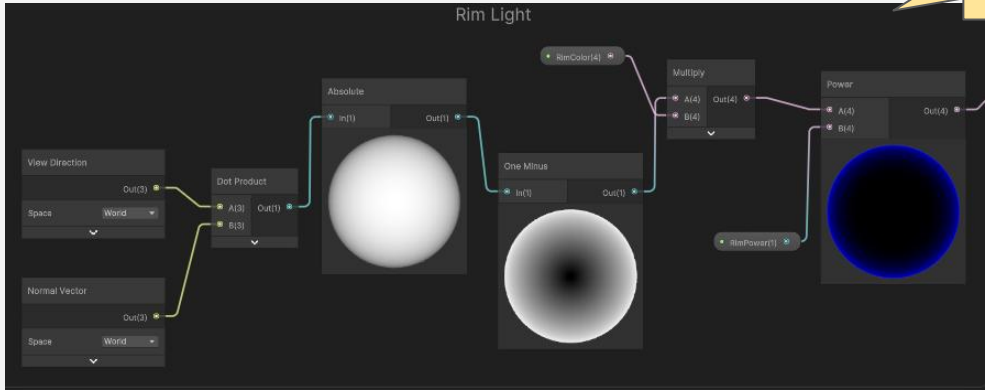
코사인 값을 기준 수치로
 $\text{dot} = V \cdot W \cdot \cos \theta$

4. 저장 및 불러오기

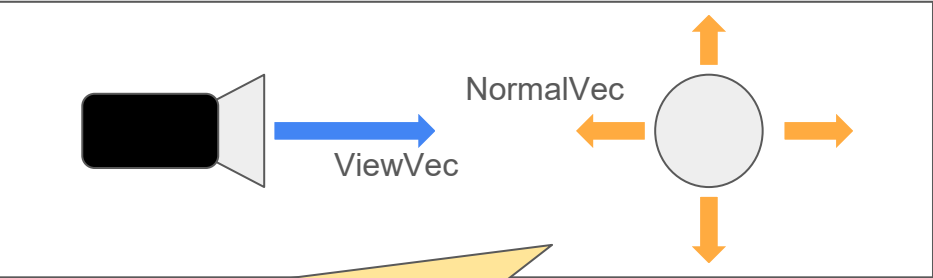


5. 셰이더(RimLight)

$$\text{RimLight} = 1 - (\text{Abs}(\text{dot}(\text{viewV}, \text{NormalV})))$$



탐지하여 플레이어를 쫓아오고 공격하는 적은 빨간색
플레이어의 타겟으로 지정된 적은 파란색
색을 다르게 한 이유는 타겟을 구별하기 위함

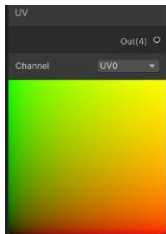
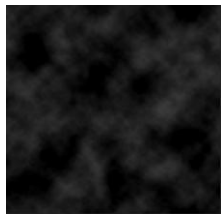


월드공간의 두 벡터로 내적연산하면 수직이 되는 값이 0, 평행하면 1또는 -1이므로 이를 절대값으로 반대쪽도 양수로 바꾼 후 oneMinus를 하면 수직에 대해서 1값을 갖는다. 이를 색상값과 곱연산하여 색상적용



RimLight 효과

5. 셰이더(Dissolve)



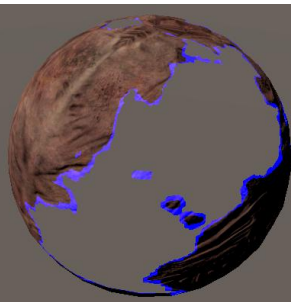
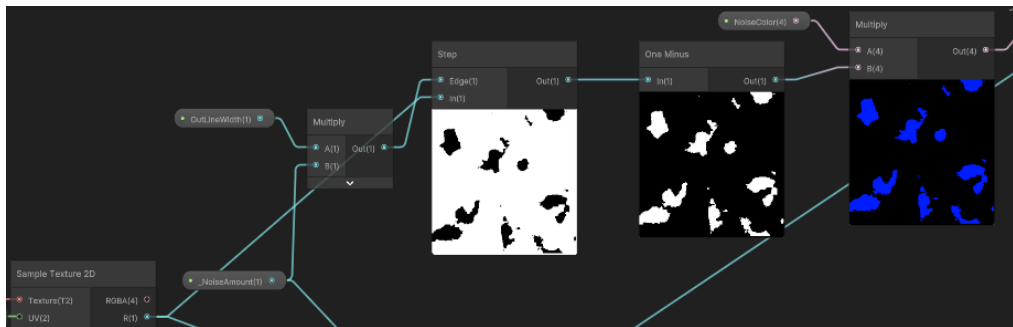
Alpha값 설정

흑백의 Noise 텍스처를 UV 좌표값으로 샘플링하여 RGB의 r값과 진행도(Amount) 파라미터와 Step 연산 후 알파값으로 설정.
=> [0,1]로 검은색~흰색순으로 투명화

$$\text{Alpha} = \text{Step}(\text{NoiseAmount}, \text{RGB.r})$$

Emission 설정

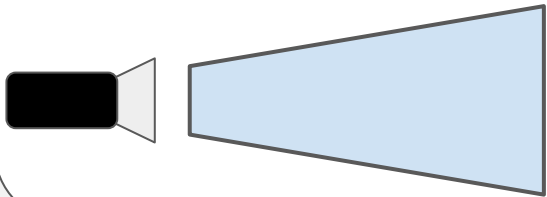
Dissolve되는 경계 = $1 - \text{Step}(\text{RGB.r} * \text{NoiseWidth})$
해당 경계에 색상을 곱하여 다음과 같은 결과



죽음시 사라져가는 연출하기 위한 Dissolve 머터리얼 개발

6. 렌더링 최적화

카메라 시야 내 렌더링 거리 감소



카메라는 절사팔면 형태로 해당 범위안에 있는 오브젝트들을 렌더링하며, 카메라 Lens의 Far Clip Plane을 1000-> 150으로 줄여서 최대 150거리까지의 오브젝트까지만 렌더링하도록 설정



빛 연산 감소

빛 오브젝트들을 Bake하여 미리 연산시켜서 씬 내의 렌더링 연산을 감소

기존 FPS성능이 39~45 에서 45~49로 **증가**
SetPassCall도 280대에서 260대로 **감소**

그림자 연산 설정

Fantastic_PipelineAsset_DeferredRenderer

- Shadow Resolution(그림자의 선명도) 4096->2048
- Max Distance(그림자가 렌더링되는 최대 거리)를 300 -> 50
- Cascade Count(Shadow Max Distance의 범위를 몇개로 나누어 가까운 곳은 고해상도, 멀어질수록 저해상도) 4 -> 2

영상 링크

<https://www.youtube.com/watch?v=bncWZRMoomo>