Portfolio : 드래곤 알 지키기

황민성

게임 소개

1. 게임 장르

모티브가 된 스타크래프트2 유즈맵

포트폴리오 “드래곤 알 지키기”

- 스타크래프트2의 “램덤 타워 디펜스”에 영감을 받아 만든 타워 디펜스 게임입니다.
- 평화로운 드래곤 숲에 침략한 인간들로부터 드래곤 알을 지키기 위해 몰려드는 적군들을 막아냅니다.

2. 기술 스택

- Unity 2022.3.30f1, Rider 2024.1.2

3. 개발 기간

- 2024.08.01 ~ 2024.09.09 (약 1개월)

4. 게임 영상 및 코드

- https://www.youtube.com/watch?v=ovLMa_hdL_Y
- https://github.com/Sicheu/Dragon-Egg-Defense.git

사용 기술

1. 디자인 패턴

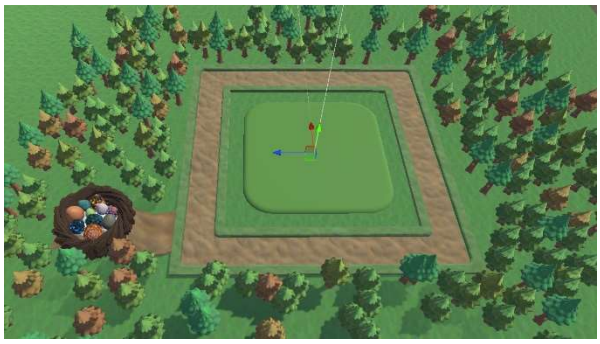
싱글턴 (Singleton)	다양한 매니저 클래스들에 사용되었으며 게임 전역에서 상태, 데이터, 리소스 등을 하나의 인스턴스로 관리하고 접근성을 확보하였습니다.
유한 상태 기계 (FSM)	캐릭터들의 상태를 세분화하여 독립적으로 설계해 각 상태의 로직 충돌을 방지하고 특정 상태에서 발생하는 이벤트 처리를 분리하여 복잡한 전투 상황에서도 상태 간 캐릭터의 흐름을 명확히 유지하였습니다.
전략 (Strategy)	모든 캐릭터는 공통 기반 클래스를 상속받아 기본 전투 흐름을 공유하면서도 고유한 스킬, 이동, 공격 방식을 오버라이드하여 다양성을 확보하였습니다.
인터페이스 (Interface)	동일한 역할을 수행하는 다양한 클래스에 인터페이스를 구현함으로써, 클래스 간 결합도를 낮추고 기능을 행위 단위로 명확히 분리하였습니다. 이를 통해 각 컴포넌트가 어떤 기능을 제공하는지 명시적으로 정의할 수 있었으며, 새로운 캐릭터나 오브젝트가 추가되더라도 동일한 인터페이스를 통해 같은 기능을 일관되게 처리할 수 있어 유지보수성과 확장성이 크게 향상되었습니다.

사용 기술

2. 응용 기술

[Post-Processing]

Bloom, 색보정, 피사계 심도 등 Post-Process 효과를 적용하여 게임의 전반적인 시각 효과를 향상시켰습니다.

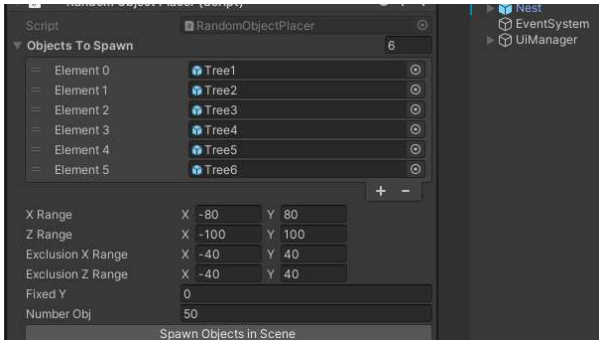


PP 사용 전의 모습

PP 사용 후의 모습

[랜덤 배치 자동화 툴]

맵 디자인의 효율성을 높이기 위해 사용자 지정 범위 내에서 오브젝트 프리팹을 자동 배치하는 커스텀 툴을 개발하였습니다. 오브젝트 간 최소 간격을 유지하며 중첩을 방지하는 배치 알고리즘을 적용하였고, 태그 기반 필터링 기능을 통해 특정 유형의 오브젝트만 선택적으로 배치할 수 있도록 구현하였습니다. 또한 에디터 확장을 통해 씬 내에서 시각적으로 조작 가능하도록 하여 개발 편의성을 높였습니다.



나무 오브젝트 랜덤 배치 툴의 인스펙터

나무가 도넛 형태로 무작위 배치된 모습

사용 기술

2. 응용 기술

[유닛 선택 및 다중 제어 시스템]

게임 '스타크래프트'와 유사한 유닛 선택 및 제어 시스템을 구현하였습니다. Ray를 활용하여 클릭한 유닛을 선택할 수 있도록 구현했으며, 드래그 박스를 통해 다수의 유닛을 동시에 선택할 수 있는 기능도 함께 개발하였습니다. 이 기능은 OverlapBox 기반의 충돌 감지를 활용하여 정확도를 확보하고, 선택된 유닛 리스트를 구성하여 후속 조작(합성, 판매 등)에 활용할 수 있도록 설계하였습니다.

게임 화면에 드래그를 입력하여 유닛들을 선택한 모습

[FSM 기반의 애니메이션 및 전투 제어]

유한 상태 기계(FSM)를 기반으로 캐릭터의 상태를 관리하였으며, 각 상태 전이 시점에 애니메이션 트리거와 이펙트를 연동하여 전투 흐름이 시각적으로 명확하게 드러나도록 설계하였습니다. 이를 통해 복잡한 전투 상황에서도 캐릭터 동작과 상태 간의 연결이 자연스럽게 유지되었습니다.

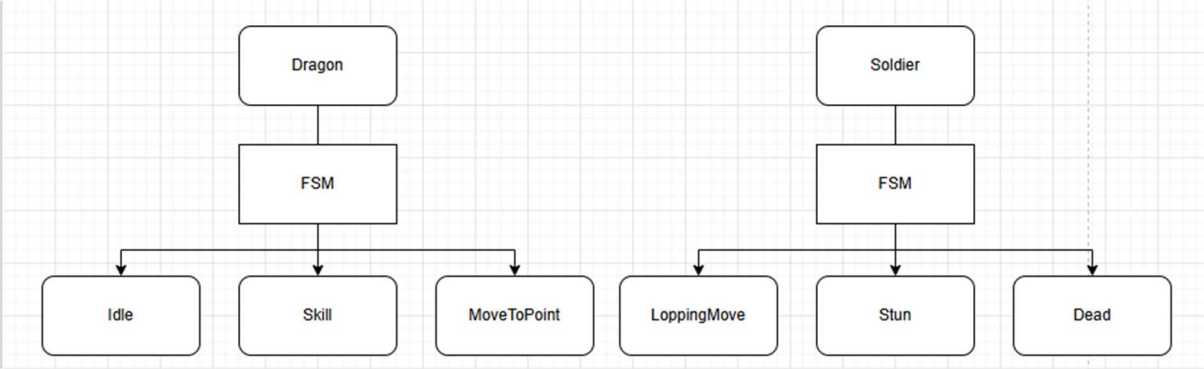
Soldier의 LoppingMove 상태

Soldier의 Stun 상태

Soldier의 Dead 상태

기능 구현

1. 드래곤 및 적군의 AI



[FSM]

- 모든 캐릭터들의 행동 기반이 되는 상태 기계로 enum을 통해 상태를 지정하고 모든 상태는 상태 진입, 진행, 퇴장 시점을 명확히 구분한 함수로 구성되어 있으며, 이를 통해 상태 전환과 AI의 행동을 명확하게 함으로 기능을 구현하였습니다.

[Idle]

- 드래곤이 공격 대기 중인 상태로, 주기적으로 주변 적의 위치를 감지하여 사거리 내 진입 여부를 판단하고, 다음 공격을 위한 스킬 쿨타임을 관리합니다.

[Skill]

- 감지된 적군에게 객체 타입에 맞는 공격을 가합니다.

[Move To Point]

- 마우스 입력에 의해 캐릭터 선택되었을 때 상태로 설정되며 입력에 따라 목적지로 이동합니다.

[Looping Move]

- 씬에서 설정된 목적지로 이동하고, 도착했다면 다음 목적지로 이동하게 하는 알고리즘으로 구성되어 있습니다.

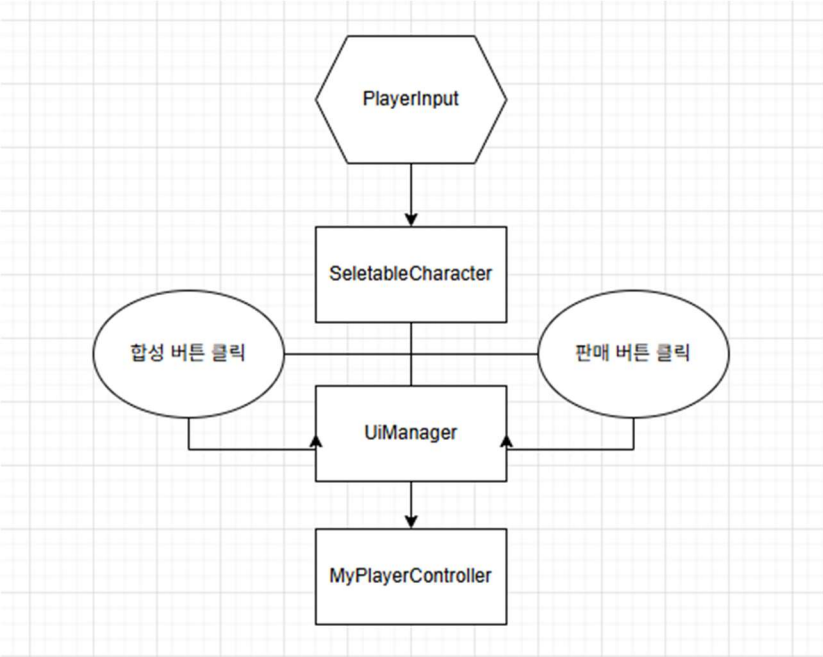
[Stun]

- 일시적으로 이동 및 행동이 제한되는 상태로, 드래곤의 군중 제어 스킬에 피격되었을 시 해당 상태로 진입합니다

[Dead]

- 체력이 0이 되었을 때 진입하는 상태로, 드래곤의 적군 감지 알고리즘에서 제외되며 Dead 애니메이션 재생이 모두 끝났다면 씬에서 사라집니다.

2. 유닛 선택 활성화와 이를 통한 합성 및 판매 등의 상점 시스템



[SelectableCharacter]

- 유닛의 선택 가능 여부를 관리하며, 클릭 또는 드래그 박스를 통해 선택된 유닛을 리스트에 등록합니다.

- 선택된 유닛은 외곽선 효과를 적용해 시각적 피드백을 제공하며, 드래그 박스를 사용할 경우 OverlapBox를 이용해 다중 선택이 가능합니다.

[UiManager]

- 게임 플레이 중 UI에 표시되는 모든 인스펙터를 관리하고 있으며 이 정보를 다른 객체에게 알리며 이벤트를 발생시켜 커맨드 패턴에서의 커맨더와 유사한 역할을 합니다.

- 합성 및 판매 버튼이 클릭되었다면 MyPlayerController 에 해당 이벤트가 실행되었음을 알립니다.

[MyPlayerController]

- 유닛 생성, 합성, 판매 등의 핵심 기능을 담당하며, 유닛들의 타입과 레벨을 비교하여 조건 충족 시 상위 유닛을 생성하며, 판매 시에는 유닛 정보를 바탕으로 보상 알고리즘을 실행합니다.

회고록

[잘 된 점]

1. **FSM(State Machine)의 효과적인 활용**

- 캐릭터의 행동 상태를 유한 상태 기계로 구조화함으로써, 복잡한 전투 상황 속에서도 캐릭터의 로직이 명확하고 안정적으로 동작할 수 있었습니다. 각 상태(대기, 이동, 공격, 스턴, 사망 등)는 독립적이면서도 유기적으로 전환되어 전체 시스템의 유지보수성과 확장성을 높이는 데 큰 역할을 했습니다.

2. **드래곤 공격 알고리즘의 설계 및 구현 성과**

- 드래곤의 공격 방식은 단일 대상, 광역 공격, 스턴 효과 등 다양한 전략을 우선순위 타겟을 향해 처리하도록 구성하였으며, 이는 전투의 전략성과 몰입도를 동시에 높이는 데 효과적이었습니다. 특히 스턴 공격의 대상 선정을 위한 우선순위 알고리즘은 기술적 완성도와 실효성 모두에서 만족스러운 결과를 보였습니다.

[개선할 점]

1. **메모리 사용 및 성능 최적화 부족**

- 개발 후반으로 갈수록 사용하는 오브젝트 수가 많아지면서 메모리 사용량이 증가했고, 프레임 저하 현상도 일부 발생하였습니다. 특히 다수의 유닛을 동시에 이동시킬 때 프레임 저하가 발생했습니다. 오브젝트 풀링과 같은 최적화 기법을 고려하긴 했으나, 구조 설계 초기 단계에서 이를 충분히 반영하지 못한 점이 아쉬움으로 남았습니다.

2. **유닛 선택 시스템의 재구성**

- 초기에 외부 코드 일부를 참고하여 유닛 선택 기능을 구현했지만, 프로젝트 특성에 맞지 않는 구조적 한계와 여러 버그가 발견되었습니다. 결국 시스템을 처음부터 다시 설계하게 되었습니다. 이를 통해 단순한 코드 인용이 아니라 구조와 의도까지 충분히 이해한 뒤 활용하는 개발자의 자세가 중요하다는 점을 배우게 되었습니다.