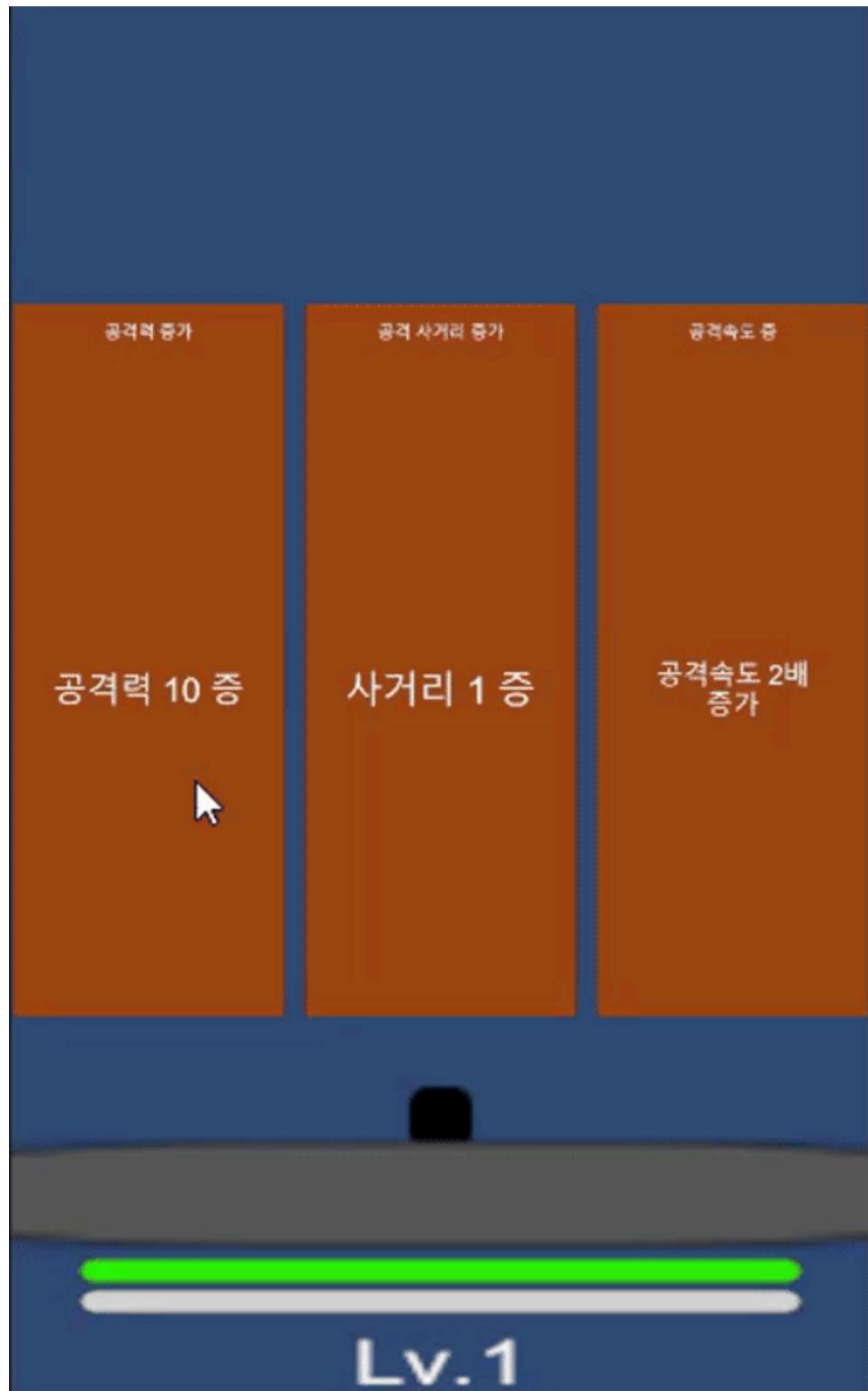


Introduction

Noname Defense Prototype는 웨이브 방어 게임을 목표로 한 프로토타입입니다. 화면 하단의 플레이어가 포탑처럼 포탄을 발사하고, 상단 격자에는 몬스터가 배치되어 성벽을 향해 전진합니다.



주요 특징

- 플레이어는 좌우로 움직이며 마우스/터치 포인터로 격자 셀을 지정해 공을 발사합니다. 셀이 비면 자동으로 사거리 내 가장 가까운 적을 다시 조준합니다.
- 몬스터는 격자 최상단에서 스폰되어 `enemyRowAdvanceInterval`마다 한 행씩 내려오고, 성벽에 도달하면 피해를 준 뒤 사라집니다.
- `attackRange`가 0보다 크면 성벽으로부터 몇 행 떨어진 위치까지 원거리 공격을 수행합니다. 0이면 충돌 공격만 유효합니다.
- 드롭을 수거하면 경험치·골드·체력·증강을 획득합니다. 게임 오버가 발생하면 남은 드롭은 연출 없이 제거되고 골드 보상만 즉시 지급됩니다(증강 드롭은 소멸).
- 레벨업 시 3개의 증강 후보가 제시되며, 선택 결과가 즉시 `PlayerEntity`에 적용됩니다.

키 피처 & 이니셔티브

- **Clean Architecture:** Core / Application / Infrastructure / Presentation 4계층으로 분리해 의존성을 최소화했습니다.
- **Definition Importer:** Excel → JSON → ScriptableObject 파이프라인을 제공해 데이터 자산 관리 비용을 낮췄습니다.

문서 길잡이

- 빠른 시작: [Getting Started](#)
- 계층별 설명: [Layers](#)
- 도구 개요: [Tools](#)
- API Reference: `api/`, `api-editor/` 폴더를 통해 DocFX가 생성한 문서를 확인할 수 있습니다.

Getting Started

프로젝트 구조를 빠르게 이해하기 위한 요약입니다.

- **왜 계층을 나눴나요?**

- Unity/Unreal 등 실시간 엔진에서 핵심 로직을 테스트하기 어렵기 때문에 Core/Application 계층을 분리해 재사용성을 확보했습니다.
- 콘텐츠 팀이 Core 로직을 직접 수정하지 않도록 책임을 분리하고, Definition Importer로 데이터 자산을 주입할 수 있게 했습니다.

계층 개요

계층	핵심 책임	주요 요소	참고
Core	게임 규칙·도메인 모델	<code>PlayerEntity</code> , <code>EnemyEntity</code> , <code>DefenseGameSettings</code>	상위 계층에서 직접 사용
Application	UseCase·서비스·포트	<code>StartGameUseCase</code> , <code>DefenseSimulationService</code> , <code>IGameInputReader</code>	Core 모델 조작 + 추상화
Infrastructure	Adapter, I/O 구현	<code>DefenseInputAdapter</code> , <code>InMemoryGameStateRepository</code> , Definition Importer	Application 포트 구현
Presentation	ViewModel·View	<code>GameViewModel</code> , <code>DefenseGameBootstrapper</code> , Player/Enemy/Fortress View	씬/FX/UI 연결

격자 기반 플레이 흐름

- `DefenseGameSettings`는 `spawnOriginX`와 `spawnColumnSpacing`으로 열을, `firstRowY`와 `rowSpacing`으로 행을 정의하고 숨겨진 -1행에서 몬스터 대기열을 채웁니다.
- `enemyRowAdvanceInterval`마다 모든 몬스터가 한 행씩 내려오며, 행 개수에 상관없이 계속 진행하다가 플레이어 위치나 성벽 위치의 트리거와 충돌하면 제거됩니다(성벽을 맞추면 피해가 발생).
- 플레이어는 좌우 이동 후 포인터 클릭으로 조준 셀(row/column)을 잠그고, 해당 셀이 비면 자동으로 사거리 내 가장 가까운 적을 다시 조준합니다.
- `attackRange`가 0보다 크면 성벽 기준으로 몇 행 위까지 공격 가능한지 계산해 원거리 공격을 수행합니다.
- 드롭을 수거하면 경험치·골드·체력·증강을 획득하며, 게임 오버 시 남아 있는 드롭은 연출 없이 제거되고 골드 보상만 즉시 지급됩니다(증강 드롭은 소멸).
- 플레이어나 성벽에 충돌해 피해를 준 뒤 제거된 몬스터는 드롭 아이템을 생성하지 않습니다.

Layers

각 계층의 책임과 주요 요소를 정리했습니다. 세부 설명은 하위 문서를 참고하세요.

- [Core](#)
- [Application](#)
- [Infrastructure](#)
- [Presentation](#)

Core Layer

Unity 구현과 분리된 게임 규칙·도메인 모델을 정의합니다.

Entities

- `PlayerEntity`: 위치, 이동/공격 스탯, 경험치·레벨·체력·력·골드, 증강 적용 효과를 관리합니다.
- `FortressEntity`: 성벽의 최대/현재 체력과 피해·회복 로직을 캡슐화합니다.
- `EnemyEntity`: 겨자 행/열 좌표, 체력/공격력/사거리/쿨다운, 드롭 테이블을 보관합니다.
- `ProjectileEntity`: 투사체 위치·속도·폭발 반경·목표 좌표를 추적합니다.
- `ResourceDropEntity`: 드롭 ID/종류/지연 시간을 기록해 수집 시점을 계산합니다.
- `GameState`: 플레이어·성벽·적·투사체·드롭 컬렉션을 묶는 루트 상태이며, 플레이어가 조준 중인 겨자 셀(row/column)도 함께 보관합니다.

Primitives & Enums

- `Float2`: Core 전용 경량 2D 벡터.
- `ProjectileFaction`, `ResourceDropType` 등 게임 진행에 필요한 상수들을 모았습니다.

Value Objects

- `DefenseGameSettings`: 플레이어/성벽 기본값과 겨자 스폰 파라미터를 정의하며, `GetCellWorldPosition`/`TryGetCellIndices`로 행·열 ↔ 월드 좌표 변환을 제공합니다. 행 개수에 관계 없이 계속 내려오는 적을 위한 트리거 위치(플레이어/성벽) 판단 기준도 제공합니다.
- `EnemyDefinition`, `EnemySpawnEntry`: ScriptableObject 기반 적 스펙/가중치 테이블.
- `GameplayAbilityDefinition`, `GameplayEffectDefinition`: 증강(Modifier) 집합을 정의합니다.

상위 계층(Application/Infrastructure/Presentation)은 이 모델을 직접 사용해 UseCase, Adapter, View를 구성합니다.

Application Layer

핵심 방어 규칙을 실행하는 UseCase·서비스·포트 계층입니다. Core 모델만 조작하며 Unity 구체 구현과는 분리되어 동작합니다.

Ports

- `IGameStateRepository`: `GameState`를 보관하고 상위 계층이 현재 상태를 주입/조회할 수 있도록 해주는 저장소 포트입니다.
- `IGameInputReader`: 최신 Input System 입력(좌우 이동, 격자 셀을 지정하는 포인터 클릭 등)을 읽어 도메인 친화적인 값으로 변환합니다.

UseCases

- `StartGameUseCase`: 플레이어·거점·게임 상태를 초기화하고 진행 상황을 리셋합니다.
- `MovePlayerUseCase`: 입력값과 `deltaTime`을 기반으로 플레이어 이동과 공격 쿨다운을 갱신합니다.

Services

- `DefenseSimulationService`
 - `DefenseGameSettings`가 정의한 격자(`spawnOriginX`, `spawnColumnSpacing`, `firstRowY`, `rowSpacing`, `gridRows`, `gridColumns`, `enemyRowAdvanceInterval`)를 바탕으로 -1행 대기열을 만들고 주기적으로 한 행씩 전진시킵니다. 행 개수와 무관하게 계속 내려오며, 플레이어 위치나 성벽 위치 트리 거에 닿으면 제거됩니다.
 - 플레이어 자동 공격은 `GameState`에 기록된 셀(row/column)을 우선 조준하고, 해당 셀이 비면 사거리 안에서 가장 가까운 적을 즉시 재조준합니다.
 - 플레이어나 성벽에 충돌해 피해를 입힌 몬스터는 드롭 아이템을 생성하지 않습니다.
 - 드롭 스폰/수거, 적 공격, 투사체 충돌, 레벨업 이벤트를 묶어 `SimulationStepResult`로 반환하여 Presentation 계층이 뷰를 갱신하도록 합니다.

Presentation/Infrastructure 계층은 위 포트를 주입받아 입력 처리와 화면 갱신을 담당합니다.

Infrastructure Layer

입력/저장소/툴 등 Unity 구성 요소가 Application/Core 계층과 상호작용하도록 중계합니다.

Input

- `DefenseInputAdapter`: 최신 Input System으로 좌우 이동과 폭격 포인터를 읽어 `IGameInputReader`에 전달.

Repositories & Tools

- `InMemoryGameStateRepository`: 단일 GameState 인스턴스를 보관.
- Definition Importer 툴: Excel `[enemyInfo]/[playerInfo]` → JSON → ScriptableObject 파이프라인.

Unity Helpers

- `Float2UnityExtensions`: `Float2`와 `UnityEngine.Vector2/Vector3` 간 변환.

이 계층에서 Application 포트를 구현하고, 외부 데이터(Excel, JSON, SO)를 연결합니다.

Presentation Layer

Unity 씬에서 GameState를 **GameViewModel** 이벤트로 바인딩하고 UI/뷰를 구성합니다.

주요 View

- **DefenseGameBootstrapper**: 모든 View/Prefab/Adapter를 주입하고 **GameViewModel**을 생성.
- **PlayerView / EnemyView / FortressView**: 위치·체력 등 상태 변화를 반영.
- **ProjectileView / ResourceDropView**: 이동·수집 애니메이션 처리.
- **AugmentSelectionView**: 어빌리티 선택 UI.
- **DefenseDebugPanel**: 개발용 치트/디버그 UI.

ViewModel

- **GameViewModel**: 입력을 읽고 **DefenseSimulationService**를 호출하여 턴 진행, 어빌리티 선택, 폭격 입력 등을 관리합니다.

Managers

- **UIFeedbackManager, FXManager, SoundManager**: UI/FX/Sound 이벤트를 중앙에서 처리(현재는 로그 기반 스텝).

씬에서는 **DefenseGameBootstrapper** 하나가 ViewModel/UseCase/Repository를 조립하고 Inspector 노출 필드에 View/Prefabs/Adapters를 연결합니다.

Tools

프로젝트에 포함된 편의 툴 목록입니다.

- [Definition Importer](#)

Excel의 [enemyInfo], [playerInfo] 섹션을 JSON/ScriptableObject로 변환하는 Unity Editor 창. 데이터 파이프라인과 ScriptableObject 자동 생성 흐름을 제공합니다.

필요한 툴이 추가되면 이 페이지에 함께 정리할 예정입니다.

Definition Importer Tool

Excel 시트를 기반으로 적/플레이어 정의를 JSON 및 ScriptableObject로 변환하는 Unity Editor 전용 창입니다.

열기

Unity 상단 메뉴에서 `Tools > Definition Importer`를 선택합니다.

Excel → JSON

1. Definition Type(Enemy/Player)을 고릅니다.
2. `[enemyInfo]`, `[playerInfo]` 섹션이 포함된 `.xlsx` 파일을 선택합니다.
3. Output JSON 경로를 지정한 뒤 `Convert` 버튼을 누르면 JSON 파일이 생성됩니다.

JSON → ScriptableObject

1. JSON Import Path에 변환된 파일을 지정합니다.
2. `Import JSON -> ScriptableObjects` 버튼을 누르면
 - `Assets/Resources/Enemies` 또는 `Assets/Resources/Players` 아래에 코드(ID) 기반 SO가 생성/갱신됩니다.
 - 기존 SO와 값이 달라진 항목은 경고 로그로 알려줍니다.

컬럼 가이드

- **EnemyInfo**
 - `enemyCode`
 - `maxHealth`
 - `attackDamage`
 - `attackRange` : 성벽으로부터 몇 행 떨어진 곳까지 공격 가능한지(행 단위)
 - `attackCooldown`
 - `dropTable`
- **PlayerInfo**
 - `playerCode`, `moveSpeed`, `attackDamage`, `attackRange`, `attackCooldown`, `maxHealth`, `luck`

주의 사항

- 드롭 리스트는 `(type:Gold;amount:5;probability:1)` 형태로 작성합니다.
- 행의 첫 문자가 ;인 경우 주석으로 간주되어 무시됩니다.
- `DefinitionImporterWindow.cs`에 XML 주석이 포함되어 있으므로 DocFX API 문서에서도 동일 정보를 확인할 수 있습니다.

TODO List

새로운 격자 기반 플레이 구조를 적용한 뒤 추가로 챙겨야 할 작업들을 정리했습니다. 진행 상황에 따라 체크 상태를 업데이트해 주세요.

Gameplay & Balancing

- [] 웨이브 데이터 자산화: 스테이지별 열/행 점유율과 이동 속도 곡선을 ScriptableObject/테이블로 분리
- [] Enemy View 프리팹 가중치 튜닝: 실제 드랍률/사거리 변화에 맞춰 시각 피드백 재검토
- [] 셀 타겟팅 튜토리얼 추가: 조준 셀을 고정/해제하는 UX 가이드와 입력 장치별 안내

Systems & Infrastructure

- [] DocFX 정적 사이트 자동 배포: GitHub Actions 또는 CI 스크립트로 `_site` 생성/배포 파이프라인 구성
- [] Definition Importer 개선: Excel 증강/웨이브 시트 추가, 검증 에러를 UIElements ValidationSummary로 출력
- [] 리소스 드롭 풀링: 게임 오버 후 즉시 제거하는 경로와 객체 풀을 통합해 GC 부담 최소화

Documentation

- [] Layers 문서에 테스트/QA 전략 섹션 추가 (Play Mode vs. Edit Mode)
- [] Tools 문서에 GIF/이미지 자산 관리 지침 명시 (경로: `images/`)
- [] TODO 페이지 최신화 자동화: 릴리스마다 완료/신규 항목 검토

Tracking

우선순위	항목	담당	상태
High	웨이브 데이터 자산화	TBD	진행 예정
Medium	DocFX 배포 자동화	TBD	진행 예정
Medium	Definition Importer 검증 UX	TBD	진행 예정
Low	문서 QA 섹션 확장	TBD	대기

필요 시 이 페이지를 반복적으로 갱신해 진행 상황을 공유해 주세요.