(명일방주식) 타워디펜스 기획 2021.01.05

0.)

게임 흐름

게임 시작 -> 계정 로그인 -> 메인 씬

* 1. 메인 씬

1. 게임의 필수 요소 정립
   1. 인게임

아군 타워(지상 배치 – 적을 저지할 수 있음/2층 배치 – 적을 저지하지 못하고 지원하는 타워)

적(저지 수를 가지고 있어 아군 타워를 돌파하여 목적지까지 도달하는 것을 목표로 함/드론 또한 존재하고 이 드론은 지형을 무시하며 전진함)

라이프 - 스테이지 별로 라이프 수치를 가지고 있음. 1도 깎이지 않아야 완벽한 승리. 이외에는 클리어한다면 보통 클리어로 처리

시작/도착 포인트 – 적이 시작 포인트에서 출발하여 도착 포인트까지 도달하면 스테이지에 존재하는 라이프를 깎고 0이 되면 패배

보스전 – 특별한 형태의 적이 등장하여 일반 적들과 함께 등장하고 스킬을 사용함

* 1. 타워 필수 요소/필요한 변수들

적 (명칭 “적 타워 -> 적” 변경)

상태(기절/사망)

자신이 현재 공격하고 있는 대상 – (사거리 내에 있는 대상을 공격 / 가장 늦게 배치 된 타워를 우선 공격)

능력치 – 체력 / 방어력 / 마법저항력 / 공격력 / 공격 속도 / 공격 타입(마법/물리/힐) / 저지 수(적의 경우에는 돌파가능한 수치) / 이동 속도

버프&디버프 – 최종 캐릭터 상태를 결정 지을 때 마지막에 합산되는 부분

리젠 시간

투사체 속도 - 0이라면 즉발식이거나 근접공격

공격 타입 – 원거리거나 근거리거나

공격 범위 – 범위냐 단일이냐

사정거리 – 좀 더 고민해보자(어떻게 값을 받아 설정해줄까/콜라이더 모양을 짜서 넣어줄까)

스킬 –슬롯을 만들어 넣자(정보를 투사체에 담아 전달 할 예정)(대상에 따라 다른 개수를 가질 것)

가하고 있는 버프 리스트

가하고 있는 디버프 리스트

당하고 있는 버프 리스트

당하고 있는 디버프 리스트

저지 당하고 있는지 판단하는 변수

현재 자신의 위치 배열 값 (스테이지를 이루는 배열의 값)

자신이 바라보고 있는 방향(추후 리소스를 추가하게 되면 좌 우 Flip을 하기 위해 사용)

길 찾기를 위한 포인트를 담고 있는 리스트(스테이지의 2차원 배열의 위치를 줌/포인트에 도달할 때마다 해당 포인트를 지워가며 다음 포인트로 이동함/보통 같은 타입의 적은 같은 경로로 움직이게 됨)

아군 타워

레벨

경험치

상태(기절/사망)

자신이 현재 공격하고 있는 대상 – (사거리 내에 있는 대상을 공격 / 대상이 죽거나 사거리에서 이탈한다면 다시 탐색 / 원거리 유닛의 경우 주기적으로 도착 포인트와 가까운 대상 순서로 공격함)

능력치 – 체력 / 방어력 / 마법저항력 / 공격력 / 공격 속도 / 공격 타입(마법/물리/힐) / 저지 수(적의 경우에는 돌파가능한 수치) / 레어도

버프&디버프 – 최종 캐릭터 상태를 결정 지을 때 마지막에 합산되는 부분

코스트

리젠 시간 – 게임 시작 시에는 0

투사체 속도 - 0이라면 즉발식이거나 근접공격

공격 타입 – 원거리거나 근거리거나

공격 범위 – 범위냐 단일이냐

사정거리 – 좀 더 고민해보자(어떻게 값을 받아 설정해줄까/콜라이더 모양을 짜서 넣어줄까)

스킬 – 3개 슬롯을 만들어 넣자(정보를 투사체에 담아 전달 할 예정)

직업 – 명일방주와 동일하게 가보자(이것에 따라 배치 가능 위치 정해짐)

가하고 있는 버프 리스트

가하고 있는 디버프 리스트

당하고 있는 버프 리스트

당하고 있는 디버프 리스트

저지 중인 적 리스트

현재 자신의 위치 배열 값 (스테이지를 이루는 배열의 값)

자신이 바라보고 있는 방향(추후 리소스를 추가하게 되면 좌 우 Flip을 하기 위해 사용)

투사체

투사체를 생성한 타워의 ID(접근하여 데미지 계산에 필요한 정보들을 얻어낼 수 있음)

공격 대상(공격 대상의 정보에 접근하여 범위 공격 또한 계산할 수 있음)

* 1. 게임 기술적 로직

게임의 맵의 구성은 길 찾기와 기믹 구현, 타워 배치를 위해 2차원 배열로 구현함.

기본적인 공격은 모두 투사체를 구현하여 사용할 것임. 투사체는 근접 공격에도 적용되며 즉시 타격하는 방식으로 구현. 원거리 투사체는 투사체 속도를 부여하여 적에게 공격

버프와 디버프를 가지고 있는 타워나 적의 경우 각자 자신의 리스트를 가지고 있어 효과를 받는 대상들을 가지고 있게 된다. 처음 버프 상태에 돌입할 때 대상의 정보를 가지게 되고 가진 상태에서는 모든 데미지 판정 때 해당 버프, 디버프를 적용하여 계산한다. 지속 시간이 끝날 때 해당 리스트에서 제외하고 해당 대상들에게 접근하여 효과 리스트에서 해당 효과를 없앤다. 되면 효과가 끝나게 된다. 이런 효과들을 UI로 표시할 때는 스테이지 상에 존재하는 타워들을 순회하며 리스트에서 꺼내와 대상에게 표시해주면 된다.

효과를 받는 대상들은 버프 디버프 리스트를 가지고 있어 자신이 어떤 효과를 받고 있는지 알고있도록 한다. 데미지 계산 시에 사용한다.

투사체

공격 함수에서 호출 할 때 인자를 넘겨주어 공격 대상을 찾아가도록 함/전달 받은 위치에 닿게 되면 스스로 소멸하며 데미지 계산에 들어감) 스크립트로 작성하여 투사체 프리팹에 붙혀 넣어 작동하게 함

투사체는 각 타워 별로 다를 것이기에 이는 서로 다르게 제작하여 해당 타워가 공격할 시 인스턴싱하여 사용

타워에게서 전달 받는 인자

공격 대상이 되는 적의 ID(이를 통해 위치, 방어력 등의 수치에 접근할 수 있음)

타워의 상태 변수들(공격력/버프 상태/방어구 관통력/투사체 속도/공격 타입/범위)

최종 데미지 판정 및 적의 체력 차감은 이 투사체에서 진행 됨

* 공격을 가할 시 과정

공격 타입

자신의 공격력 \* 공격력 버프 항목(자신이 가진 리스트에서 찾아냄 – 20210107 현재로써는 %로 적용함)

방어구 관통력 (공격 대상의 물리/마법 방어력에 접근하여 적용되는 수치를 계산함 – 현재 기획상으로는 상대 방어력을 %로 깎아냄)

* 최종 데미지 계산

[(공격력 \* 버프효과) – (대상의 방어력\*(100-방어구 관통력)/100)] #이 계산의 값이 0 이하라면 1로 적용

위의 값을 공격하는 대상의 체력에서 차감

범위 공격의 경우공격 받는 대상을 찾아내어 대상마다 적용해줌

위의 값과 과정들은 투사체에 담겨져 사용됨

* 저지 과정

타워는 “저지 수” 를 가지고 있음

자신이 바라보고 있는 방향의 앞으로 한 칸과 자신이 서있는 칸의 적의 “저지 수” 를 합산하여 계산함

합산 한 저지 수가 자신이 가진 저지 수 보다 낮을 경우 모든 적의 이동이 차단되며 이는 아군 타워가 상대에게 저지 당함의 변수를 작동시켜 이동 불가 상태로 만든다

저지 수를 넘길 경우 저지 수가 가장 큰 대상을 찾아내고 그 대상이 아군 저지 수와 같을 경우엔 그보다 낮은 대상을 모두 움직일 수 있도록 변경하고 모두 같은 저지 수라면 먼저 온 순서대로 보냄 / 가장 큰 대상이 있고 아군 저지 수 보다 낮을 경우 저지 수가 낮은 먼저 온 대상부터 보낸다(먼저 왔다의 정의는 적 스폰 시 고유 ID를 부여하여 판단)

* 적의 이동

적의 이동은 A\* 알고리즘을 통해 구현한다(스테이지를 배열로 구현한 이유)

배열 탐색을 통해 특정 포인트까지 이동하게 된다

필요 함수 목록

Public float GetHp(); - 대상의 HP 수치를 반환

Public float SetHp(); - 대상의 HP 수치 설정

Public float GetArmor(); - 대상의 방어력 반환

Public float GetArmorType(int Type); - 대상의 방어 타입 반환(공격자의 공격 타입에 따라 불러올 리턴값 변화)

private DamageCalculate(); - 데미지를 객체 내에서 계산함

Public float GetDamage(float CalculatedDamage); – 계산된 데미지를 대상의 체력에서 차감

Public void MakeAttack(int EnemyID, int MyID); - 프리팹을 만들어 투사체 생성

Public void Initiate(); – 대상을 초기화

Public void Stun(); - 대상을 기절상태로 만듦

Public void Death(); - 대상을 사망상태로 만듦

1. 게임 계정

게임 실행 시 로딩 씬을 띄우고 Load 함수를 이용해 소유하고 있는 아군 타워와 리소스를 받아온다

정보 저장은 json 파일로 저장

게임 시작 시 계정을 등록하고 이는 json 파일로 등록되며 각 계정별로 상태를 저장한다

타워 별로 고유 ID를 가지고 있으며 이는 GameManager가 관리하여 json 파일에 넣어 저장

계정 -> 가지고 있는 타워 리스트 -> 타워 구조체에 담겨진 정보들을 json 파일로 모두 갖고 있음(레벨만 가지고 있고 나머지는 미리 내부에 정보를 채워둔 구조체에서 가져와 레벨 별 증가 수치를 합산하여 능력치를 체크)