20200604

UML에서 계획했던 TurnSystem, CardSystem, EnemySystem을 모두 완성했다.

모든 맵을 하드코딩할 수 없기 때문에 데이터를 읽어오는 방법을 찾아보고 있다.

CSV를 기본적인 데이터 파일로 사용하고 JSON을 일부 중요 데이터 파일로 사용 하기로 결정함

CSV를 사용 해야 하는 곳

Unit

* Hp, DrawCount

Card

* Name, Cost

State

* 없는듯?

Enemy

* States, MaxHp, atk

Deck

* Allys

EnemySystem

* Enemies, pos(?)

20200605

CSV를 읽는 프로그래밍을 먼저 하려고 했으나 어제 밤에 친구와 상의를 하면서 바뀐 UI들 때문에 이걸 먼저 수정하기로 했다. ㅂㄷㅂㄷ

유닛 위에 나오는 카드 간격 0.5(임시) : 상속후에는 0.35(임시) 실제 크기 0.7

유닛 위에 카드가 나오게 했고, 사용, 정렬 모두 다 되는 것을 확인

20200606

몬스터 3\*3 공간에 배치하는 것으로 변경 완료

카드에 사정거리 추가 완료(나 구조 ㅈㄴ 잘 만들었어 진짜 기분 겁나 좋아)

CSV(Comma-Separated Values) : 약자에서 볼 수 있듯이 comma(,)로 나뉘어 있는 값들 이란 뜻

CSV는 엑셀에서 여러 시트를 만든 것을 인식을 못한다.

* 즉, 한페이지 짜리를 여러 개 사용해야 한다

CSV 파서를 인터넷에서 찾아서 사용함. 사용 이해는 했으니 괜찮을 것 같음. 만들지 못한게 아쉽지만, 내용이 괜찮아 보임

20200607

친구가 그린 UI틀을 가져와서 게임에 적용시킴, 몬스터 이미지도 그럴싸한 걸로 적용(무단사용 ㅋㅋㅋ) 몬스터 이미지 때문인지 뭔가 그럴싸해 보임

몬스터 AI를 구체적으로 어떻게 설정할 지 고민

20200608

Hate시스템을 만들었다. 몬스터가 각각의 유닛에 대한 Hate수치를 가지고 있고 각각의 방법에 따라 적을 선택하여 공격한다.

체력바가 추가되었다. 최대치 만큼 미리 만들어 둔 UI를 활성화, 비활성화를 오가며 업데이트 하여 사용한다.

20200609

8일자 작업까지 UML 업데이트 완료, 깃허브 업로드

적 체력바 만듬, 몬스터 죽는 거 정상구현, 아군 죽으면 사라지는 것 대신에 영정사진 걸리고 카드 사용 봉인

20200610

몬스터, 아군 체력바 연동 제대로 안되던 것 수정, 아군이 모두 죽으면 몬스터가 더 이상 공격하지 않음(게임 오버)

20200613

구체적인 맵을 하나 만들어 보기로 결정

유닛 구조 개편, 상태이상 추가, 몬스터 구조 개편 필요

Enemy대신 State가 주체적인 판단을 하게 변경, 이제 State가 대상을 탐색하고 currentState를 변경함. Enemy는 이제 atk(공격력)을 가짐, 이 공격력에 의거해서 데미지를 입힐 것임. MaxHp, Hp, Atk를 가져오는 과정을 모두 CSV파일 로드로 대체함.

Resources.Load(path)로 가져온 프리펩의 변수 일부를 미리 설정하고 그것을 Instantiate 하려 했지만 공개된 정보가 아니면 복제가 되지 않았다. 그래서 해결책을 한참 찾았는데 awake 함수는 active일 때만 작동하기 때문에 가져온 프리펩을 Active false로 만들고 Instantiate 한 후 값을 할당 했다.

Unit 구조를 개편했다. 이제 각각의 스크립트만 만들고 추가적인 작업을 하지 않아도 자동으로 해당하는 체력, 공격력, 드로우 수, 카드들을 읽어온다. CSV파일 구조를 일부 변경했다. 이제 턴 종료시에 모든 카드를 버린다.

Card 구조를 개편했다. 이제 각각의 Use만 만들면 돌아간다. CSV 파일 구조를 일부 변경해서 사용한다. 내일은 각각의 카드 제작과 상태이상 구현을 목표로 하고있다.

20200614

Unit에 atk를 추가하고 slash, shot 구현.

갑자기 떠오른 패시브 아이템들은 대체 어떻게 구현을 해야 할까 고민을 했는데 해결책이 안 나와서 친구들과 얘기했더니 떠올랐다. 아이템을 획득하면 그 아이템에 해당하는 컴포넌트가 추가되고 그 컴포넌트에서 아이템의 효과를 자동으로 발동하게 하는 방식으로 만드는 것이 좋은 방법 같다.

20200615

상태이상 시스템을 어떻게 구현할 지 정했음. hate처럼 만들되 Enter, Exit, TurnUpdate 등의 State와 같은 함수를 만들어서 적용시킬 것. 또한 Unit이나 Enemy의 기본 스텟을 전투 중에만 사용하는 복사 스텟을 만들어서 그 수치를 변경하기로 결정함. 이젠 고민을 더 하지 않고 만들어보고 문제가 있으면 고치려고 함.

상태이상 구현완료, Condition클래스를 사용함. 이제 Unit을 상속받아서 Undead와 Enemy가 작동함. CaveSpider가 사용하는 Slow 클래스도 제작완료, 제대로 작동함을 확인. 같은 상태이상이 중첩되면 중첩된 만큼 지속시간이 늘어남.

각 상태이상을 얻을 때 해당하는 아이콘이 자동으로 붙게 설정, Slow의 카드 드로우 수가 복구 되지 않는 문제 해결, foreach구문 도중에 변수가 지워졌을 때 생기는 오류를 goto문을 이용하여 수정함. 일단 지금 문제는 다 해결한 듯하다. 내일은 일어나서 바뀐 것들을 정리하고 UML에 적용시킨 후에 추가적인 카드들을 제작할 예정이다.

20200616

어제 만든 내용들 UML 정리 완료. 깃허브 업로드 후 각종 카드 만들 것

Tount 상태이상 추가, Roar카드 제작, Ready 상태이상 추가, Ready카드 제작.

Ready 과정에서 상당히 맘에 들지 않는 코딩을 함. 해결책을 못 찾아서 일단 그냥 사용함. 개인적으로 이런 하드코딩에 가까운 방법은 사용하고 싶지 않음

Heal카드, DefensePosition카드 제작 완료. 이제 카드를 사용할 때 거는 조건에 따라 적에게 사용할지 아군에게 사용할지 정할 수 있음.

이제 SupportShot카드만 만들면 기본세트 끝! 이건 내일 마저 만들고 정리 후 업로드 할 예정

20200617

SupportShot 구현. Condition을 상속받는 CountCondition, CountdownCondition으로 구분했고 이후로도 AttackedCondition, HitedCondition과 같은 이름으로 구분 지어서 만들 예정이다. 각 Condition의 특징은 이름에서 알 수 있듯이 어떤 것을 추가로 가지고 있거나 어떤 때에 발동하는 지에 대해 알 수 있다. 이제 기본적인 세팅이 끝났으니 앞으로는 전체적인 게임의 흐름을 만들고 남은 전투부분을 채울 생각이다.