기본 조건

1. 인터페이스 / 상속 / 가상 함수 / 프로퍼티
2. 컬렉션
3. GDI+ 라이브러리
4. 델리게이트 / 이벤트

추가 조건

1. 쓰레드
2. 싱글톤 / 팩토리 패턴
3. 파일처리
4. 네트워크

Unit.DLL 생성

부모클래스 Unit 생성 후 상속을 통해 파생클래스들 생성

EnemyUnit / PlayerUnit 폴더 안의 유닛 cs 전부 파생클래스로 생성함

Unit.cs 에 인터페이스 ICritical\_HIT 를 생성. 치명타 관련기능

EnemyUnit 폴더 안의 적들 유닛 파생클래스에 다중상속 해줌

Private로 은닉되어 있는 Unit 멤버데이터들 가상함수로 파생클래스에 넘겨주고

파생클래스에서 오버라이딩으로 받도록 디자인

UNITTYPE 열거체에 지정해준 유닛의 타입(플레이어,적군) 프로퍼티로 저장함

unitFactory에서 유닛을 인스턴스 하고 이 프로퍼티를 통해 유닛 타입을 지정해줌

BattleManager.cs 에서 Battle\_ATK 라는 델리게이트를 이용, 일반공격과 스킬공격에서

체력 설정함수 HIT에 매개변수로 각 공격력 설정하는 메서드가 등록된 PAtk , EAtk 전달

함으로 전투시 유닛들의 체력을 설정한다

Battle\_Attatch 클래스에서 컬렉션 ArrayList 사용

적 유닛과 플레이어 유닛을 받아와서 ArrayList 객체 callbacks안에 담고 배틀매칭 의도

Form1.cs 와 DeadEventHandler.cs 에서 유닛이 죽었을 경우의 이벤트를 다뤄 줌

유닛의 생존여부를 다룰 bool 타입 변수와 int형 체력 변수를 매개변수로 받아와서

체력이 0이하로 떨어질 경우, bool 타입을 바꿔주며 유닛의 죽음 전달하는 이벤트

UnitFactory.cs 에서 추상클래스 UnitFactory 생성. 팩토리 패턴을 이용하여

팩토리 객체들에서 유닛 인스턴스 부분과 유닛 타입을 지정해줌

게임의 확장성을 고려하여 유닛 객체가 여러 개 동시 생성될 경우를 대비 팩토리를 설계함

UnitXML을 이용, 유닛 도감을 펼칠 수 있도록 파일처리도 사용함

Form1.cs 에서 쓰레드와 GDI 라이브러리를 이용하여 폭발 이미지 애니메이션 생성

쓰레드를 통하여 공격버튼을 누른 후 이미지 생성에 텀을 줘서 좀 더 타격감 있는 느낌 줌

기타 여하 이미지들 GDI+ 라이브러리 Image 를 통하여 등록된 리소스를 불러오는 방식으로

이미지박스에 넣어 줌