\*\* 스크립트 설명

\Move\_Actor\_F

우선 move\_actor는 움직이는 것들의 기본 토대가 된다.

player\_ctrl은 플레이어가 가지는 것들의 관리값이고

player\_move는 플레이어 입력 부분. (키보드라고 생각하면 됨)

\Map\_background

Map에 있는 오브젝트들의 클래스 (딱히 상속x)

Ex) 산, 구름, 지형 등

이 맵 오브젝트들의 움직임은 Map\_Ctrl스크립트를 통해 관리되며

Hierarchy창에 MapCtrl에 달려있음

[ 추가사항 ]

모델링이 평평한 모양이 아니라서 collider를 붙이면 붕 뜨는 부분이 존재함.

Wall\_Collider\_Left, Right 로 기차의 콜리더 부분 따로 만들었음 ,

Mesh Renderer를 꺼놓은 상태. Trigger로 체크할 것

**<<주의사항>>**

* String 을 함부로 쓰지말자

C# 문자열은 Immutable 즉 한번 문자열이 설정되면, 다시 변경할 수 없다.

Ex) 문자열 변수 s가 있을 때, s = “C#’ ; 이라고 한 후 다시 s = ‘F#’ 이라고 실행하면, .NET 시스템은 새로운 string 객체를 생성하여 ‘F#’ 이라는 데이터로 초기화한 후 이를 변수명 S에 할당함. 즉 변수 s는 내부적으로는 전혀 다른 메모리를 갖는 객체를 가리키는 것.

그래서 System.Text.StringBuilder 클래스를 이용하자.

String 클래스는 Immutable이라서 문자열 갱신을 많이 하는 프로그램엔 적당하지 않다. 반면 mutable타입인 stringbuilder는 별도 메모리를 생성, 소멸하지 않고 일정한 버퍼를 갖고 문자열 갱신을 효율적으로 처리한다.

Int I = 9;

StringBuilder sb = new StringBuilder();

Sb.Append(i.ToString());

String s = sb.ToString();

* For? Foreach?

Foreach는 for, while 등 다른 루프 문장보다 내부적으로 최적화 되어있으므로 가능하면 foreach를 쓰자.

아 맞다

그 각 object들은 train 안에 넣어놓고

인스턴스될 위치를 미리 정해놓기\

그리고 미리 소파랑 상자 만들어놓고 오브젝트 풀링으로 하기

그리고 또 능력치 관리할 그거 머더라 검색해보기

**플레이어 -> HP**

**전체 플레이어 (공통/ 공유) -> 코인, 아이템 개수**

**승객 -> 질병수치, 배고픔 수치**

**몬스터 -> HP, 공격력(등장하는 애들마다 공격력이 다름)**

**기차 -> 스탯 ( 속도, ㅇㅇ, ㅇㅇ), HP**

기차는 한 칸마다 관리할건가?

기차를 한 칸마다 관리해야 (칸 하나 프리팹 -> 스크립트달아놓기) 붙이기도 떼기도 편하지

그리고 전체 기차 관리 스크립트 하나 더 놓고

왜냐면 만약에 3번 칸이 hp가 0됐으면 4,5,6까지 달려 있었을 때 걔네까지 다 떨궈야 되니까