배경생성하기 스프라이트 폴더에서 드래그 드롭

배경에서 어떤레이어를 앞쪽에 보여줄건지는 sortinglayer에서 설정가능

배경에 스크롤링 오브젝트 스크립트 생성

자주 사용했던 트랜스폼 transform.Translate 이다.

배경과 시작타일을 왼쪽으로 계속 이동시켜준다.

public class ScrollingObject : MonoBehaviour {

public float speed = 10f; // 이동 속도

private void Update() {

// 게임 오브젝트를 왼쪽으로 일정 속도로 평행 이동하는 처리

if (!GameManager.instance.isGameover)

{

transform.position = transform.position + transform.right \* -speed \* Time.deltaTime;

or

transform.Translate(Vector3.left \*speed\*Time.deltaTime)

or

transform.Translate(new Vector3(-speed,0,0)\*Time.deltaTime)

}

}

}

백그라운드 루프문

Awake는 start처럼 초기 1회시작되는 메서드로 스타트보다 1프레임 빠르게 시작

먼저 사용할 박스컬린더의 컴포넌트를 가져온후 가로길이를 측정한다.

왼쪽으로 배경의 가록 길이만큼 이동했을때 리포지션을을 불러온다

리포지션 offset에 현재 가로의 2배가 되는 값을 설정해논다

현지위치에서 오른쪽 방향으로 그 길이만큼 이동

transform.position은 Vector3이고 offset은 Vector2 이므로 앞에 형변환을 위해 자료형식을 정해준다.

public class BackgroundLoop : MonoBehaviour {

private float width; // 배경의 가로 길이

private void Awake() {

// 가로 길이를 측정하는 처리

BoxCollider2D backgroundCollider = GetComponent<BoxCollider2D>();

width = backgroundCollider.size.x;

}

private void Update() {

// 현재 위치가 원점에서 왼쪽으로 width 이상 이동했을때 위치를 리셋

if (transform.position.x <= -width)

{

Reposition();

}

}

// 위치를 리셋하는 메서드

private void Reposition() {

Vector2 offset = new Vector2(width \* 2f, 0);

transform.position = (Vector2)transform.position + offset;

}

}

고정픽셀 캔버스 컴포넌트의 ui스케일모드의 기본설정인고정 필셀크기는 화면의

크기와 상관없이 고정된 값임

그렇게 될경우 스크린 크기에 따라 값이 변경될수 있으므로 켄버스 속성에서 scale모드를 scale with Screen size로 변경

Score 점수판같은 경우에는 활성화 시켜서 항상 보이게 하지만 게임오버와 재시작 버튼은 부모자식 관계로 만들어 게임오버 됬을때만 보이도록

비활성화 시키자

스태택 변수:

여러 오브젝트들이 해당 변수 하나만을 공유하는것

싱글톤 패턴으로 만들어야한다

게임매니저 오브젝트의 경우 언제든지 접근이 가능하고 프로그램에 단하나만 존재해야함

그래서 싱글턴패턴으로 구현하는편

게임매니저 클래스

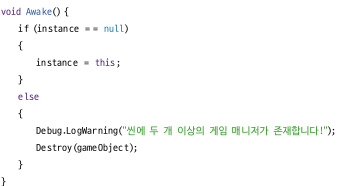
싱글턴 할당할 전역변수, 게임오버 상태, 점수출력 유아이, 게임오버 게임 오브젝트

등이 있고 score 을 텍스트로 설정한것과 gameoverUI를 차이는 gamveoverUI는 setattribute를 사용해야하므로 게임 오브젝트로

score는 값만 바꾸면 되므로 텍스트로 설정한게 아닐까 생각해본다.

Start와 역할은 비슷하지만 한 프레임 일찍시작되는 awake에서 인스턴스 확인 및 선언

Public static GameManager instance;



Instance는 싱글턴이될 GameManager 오브젝트가 저장될 변수이고 이미 만들어져있는게 없다면

이 오브젝트를 게임매니저로 이미 다른게 존재한다면 지금 오브젝트를 파괴한다

SceneManager.GetActiveScene()은 현재 활성화된 씬의 정보를 가져오는 메소드

SceneManager.LoadScene은 활성화 시킬 씬을 지정하는 메소드

OnEnable(): 한번 실행되고 마는 start와 awake와는 달리 컴포넌트가 활성화 될 때 자동으로 한번 실행되는 메소드. 개별 컴포넌트를 끄고 켜지 않고 게임 오브젝트를 끄고 켜는 방식으로 상태를 리셋할 때 주로 이용된다.

발판 생성기 : 프리팹으로 무한반복하는 제일 쉬운방법은 필요할때마다 생성하는것이지만

메모리 사용량을 조절하기위해 미리 만들어놓고 불러오는 방식이 있고 오브젝트 폴링 방식이라고 한다.

게임시작시 로딩이 오래걸리는 것도 이런것들 때문에\

instantiate()나 Destroy()메서드는 메모리 성능을 많이 요구함

변수로는 사용할 원본 프리팹, 생성 수량, 어떤위치로 이동할것인지에 대한 정보,배치 시간등에 관한 변수가 있다.

업데이트에는

게임오버시에는 동작하지 않도록 하고 배치시간과 배치주기시간이 현재시간보다 작다면 현재시점을 배치시간으로 하고 다시 배치주기, ypos를 랜덤으로 바꾼다음

발판 오브젝트는 OnEnable을 가지고 있으므로 플랫폼 오브젝트를 껏다가 다시 켜준다.

현재 순번의 발판을 오른쪽에 재배치하고 순번값을 증가시켜준다음

마지막 순번에 도달했다면 순번을 리셋한다.

이렇게 작성도니 스크립트를 spawner안에 넣도록 하자