1. **씬 전환**

* 유니티 구성요소 - 스크립트, 에디터, 카메라, 씬 등
* 폴더 창에서 Create – Scene

\* 최초 유니티에서는 SampleScene이 존재.

\* 새 씬마다 화면 전환 (ex. 게임에서 스테이지, 맵 등 전환)

* UI - Legacy - Button 을 새 씬의 Canvas 밑에 만들어줌
* 게임 오브젝트 생성하여 SceneChanger 라고 명명.
* Assets - Scripts 폴더에 SceneChange 스크립트 생성.
* 해당 스크립트에 public void OnClickSceneChange() 함수 생성
* using UnityEngine.SceneManagement 네임스페이스 추가
* 함수에 SceneManager.LoadScene("SampleScene") 호출구문 추가
* SceneChanger 오브젝트에 SceneChange 스크립트 추가
* 만든 버튼의 On Click 기능에 +, SceneChanger 오브젝트 추가하고 OnClickSceneChange 함수 지정
* 버튼 누르면 씬 전환됨.

1. **DontDestroy**

* 폴더에 DontDestroy 스크립트 생성
* 스크립트에 Awake 함수 작성, DontDestroyOnLoad(this); 구문 작성.

\* this는 나 자신을 가리킴. (해당 스크립트를 가진 오브젝트)

* DontDestroyObj 라는 이름의 게임 오브젝트 생성
* 해당 오브젝트에 DontDestroy 스크립트 달기
* 씬 저장 후 플레이하면 DontDestroyOnLoad라는 씬이 하이어라키에 생긴걸 확인 가능.
* 버튼을 눌러 씬 전환을 해도 DontDestroyOnLoad라는 씬과 하위 오브젝트는 사라지지 않음.
* 하이어라키에서 DontDestroyObj 하위에 임의의 오브젝트를 달아줘서 테스트 해보면 상속구조에 따라 하위 오브젝트도 다 남아있는 걸 확인 가능

\*빌드에 포함되어야 함.

-> File - Build Settings - Scenes In Build에서 Add Open Scenes를 눌러서 다른 씬들도 빌드에 추가하고 저장해줘야 함.

1. **애니메이션 설명**
   * OpenGameArt.Org에서 Hero, Animal 검색해서 다운로드

\*상업적 이용 등 자유이용 가능한 저작권 : CC0

* + 폴더 창 Resources 폴더 안에 character, monster 폴더 생성
  + 해당 스프라이트 각각 폴더로 이동
  + 스프라이트 선택하여 Inspector 창에서 Sprite Mode를 Multiple로 설정.
  + Sprite Editor를 눌러 Slice 클릭
  + Type에서 Automatic으로 Slice 했을때 같은 크기로 잘 잘리면 그대로 쓰되, 잘 안잘리면 Grid By Cell Size로 바꾸어서 균일한 규격값을 입력하여 Slice. 이후 Sprite Editor 창에서 Apply를 클릭하면 폴더 창의 Sprite가 Slice된 것을 확인할 수 있음
  + Slice 된 Sprite 에서 원하는(달리는) 이미지만 선택하여 하이어라키 창으로 드래그 앤 드롭.
  + 애니메이션 생성 대화상자가 팝업되며, character\_run 등 적당한 이름으로 명명하여 캐릭터 스프라이트와 같은 경로에 저장.
  + 하이어라키에 캐릭터 오브젝트가 생성되고, 애니메이션과 에니메이션 컨트롤러가 붙어있음을 확인 가능
  + 해당 캐릭터의 크기를 scale 값 등으로 적당하게 조절하여 맵 위로 이동. (몬스터도 마찬가지)

\*몬스터의 방향이 마주보지 않는 이미지라면 Flip 옵션의 X축을 체크.

* + 캔버스 생성하고 Order in Layer 값 1로 수정. 해당 캔버스 하위에 캐릭터와 몬스터 오브젝트를 달아줌.
  + 캐릭터와 몬스터 오브젝트의 Order in Layer 값을 1로 수정.
  + 기존 캔버스에 백그라운드 스크롤 오브젝트를 달아주고, 백그라운드 스크롤 오브젝트 하위에 배경이미지를 모두 달아줌.
  + 기존 캔버스의 Order in Layer 값이 0인지 확인.
  + 두 캔버스의 Render Mode를 Screen Space - Camera로 변경하고 Render Camera에 하이어라키의 Main Camera를 끌어다 붙여줌.
  + 플레이모드에서 애니메이션과 배경 스크롤 재생 확인.
  + 폴더 창에서 전투 이미지를 선택하여 우클릭 - Create - Animation을 선택하여 character\_attack으로 저장.
  + 캐릭터의 애니메이션 컨트롤러를 더블 클릭하여 켜지는 Animator 창에 character\_attack 애니메이션을 드래그 앤 드롭하여 추가.

1. **충돌, RigidBody 포지션 변경 등**
   * 몬스터를 화면 밖에 두면 안보임
   * 폴더 창에서 Monster 스크립트를 생성
   * 배경은 여러장이라 localPosition을 설정했지만 캐릭터는 혼자니까 본인만 움직이면 됨.
   * Monster 스크립트에 public float speed 선언.
   * Start에서 speed = 0.5f 로 설정.
   * Update에서 this.transform.Translate(new Vector3(-1 \* Time.deltaTime \* speed, 0)); 구문 작성

\* transform은 변경시킨다는 의미. Translate는 매개변수 값만큼 변경한다는 의미.

* + monster 오브젝트(rat)에 monster 스크립트를 달아줌
  + 물리엔진을 추가해주기 위해 character, monster 오브젝트에 Add Component로 RigidBody를 추가
  + Use Gravity는 중력의 영향을 받는 옵션. 우리 게임에서는 꺼주도록 함.
  + RigidBody 적용만으로는 충돌 체크를 할 수 없음(그냥 서로 지나감)
  + 각 오브젝트에 Box Collider 컴포넌트를 추가해줌.
  + 각 오브젝트에 초록색 상자가 생김. Box Collider에서 Size를 적당하게 조절.
  + Is Trigger는 조건 체크를 위한 옵션. 체크하도록 함.
  + Character 스크립트를 생성.
  + public Animator anim 변수 선언, void OnTriggerEnter(Collider other) 함수 작성.
  + 함수 내부에

if (other.transform.tag == "enemy")

{

Debug.Log("어쩌구 start");

anim.Play("character\_attack");

}

구문 작성

* 몬스터 오브젝트의 Inspector 상단의 Tag 항목에서 Tag를 추가하여 enemy로 명명.
* 캐릭터 오브젝트에도 player라는 Tag를 추가
* 캐릭터 오브젝트에 Character 스크립트 달아주기
* 캐릭터 스크립트 컴포넌트의 Anim 요소에 Character 오브젝트를 달아줌. 플레이 시 충돌 확인 가능.

\* 캐릭터와 몬스터 오브젝트의 Z축 값이 동일해야함. 0으로 수정.

* monster 스크립트의 OnTriggerEnter 함수에 tag는 player, 조건문 내부에 speed = 0f; 작성. 몬스터 충돌 시 정지 확인 가능.

1. **객체지향, 싱글톤**

* Java, C#, C++ 등 객체를 기반으로 하는 언어
* Class와 같은 하나하나의 인스턴스를 객체라고 함
* 객체는 각각 자신의 역할만 존재.
* if, 캐릭터가 몬스터를 때렸을 때 몬스터의 hp를 어디서 깎는지?

-> 캐릭터 객체와 몬스터 객체 각각에서는 알 수가 없음.

* so, GameManager와 같은 객체가 존재함. but, GameManager는 1개만 존재해야함
* 변수에는 지역(local), 전역(global), 정적(static)변수가 존재.
* 단 하나만 존재해야하는 static 변수를 이용하여 GameManager와 같은 객체를 다룰 것.

- 코드로 테스트

monster 스크립트에서 character 스크립트로 접근 가능? 불가.

GameManager 스크립트를 생성해준다.

GameManager 스크립트 자체를 static으로 만들어주기 위해, 하이어라키에 GameManager 오브젝트를 생성하고 오브젝트에 해당 스크립트를 달아줌.

- 싱글톤(Singleton) 코드로 설명

GameManager 스크립트에 #region Singleton ~ #endregion 코드 작성

원하는 과정을 return해줄 수 있는 get함수.

class GameManager라는 객체로 만들어주고나서 이 클래스 자체(GameManager) 변수 \_instance를 static으로 선언함(더 이상 만들 수 없게)

시작하는 함수 Awake에서 \_instance = this 로 넣어줌으로써 더 이상의 GameManager 오브젝트가 생기지 못하게 함.

외부에서 접근하기위한 변수가 public static GameManager instance임.

\*get은 얻어오는 것, set은 세팅해주는 것.

if 구문은 만약에 GameManager가 없을 경우 만들어주기 위한 예외처리 구문임.

- 어떻게 쓰는지 알기 위해 코드 작성

GameManager 스크립트에 아래 코드 작성.

public void TestSingleton()

{

Debug.Log("TestSingleton");

}

character나 monster 스크립트에 GameManager의 TestSingleton 함수 호출 구문 작성

-> 플레이 시 TestSingleton 로그가 찍힘.

- GameManager에서 캐릭터와 몬스터를 컨트롤하기 위한 코드 작성

public Character character 선언

public Monster monster 선언

GameManager 오브젝트에 Character와 Monster 항목이 생김. 각각에 맞는 오브젝트를 달아줌.

로그를 찍는 TestSingletonCharacter, TestSingletonMonster 함수를 각각 캐릭터, 몬스터 스크립트에 작성.

GameManager 스크립트의 Start함수에 character.TestSingletonCharacter();, monster.TestSingletonMonster(); 호출구문 작성.

\*캐릭터와 몬스터의 1대1 형식인 우리 방치형 게임에서는 캐릭터에 몬스터 정보를 같이 넣거나 그 반대로도 가능하다고 생각할 수 있지만, 볼륨이 큰 RPG에서는 그게 불가능하기 때문에 기본적으로 게임매니저를 사용해야함.

Monster 스크립트의 트리거 코드를 삭제.

게임매니저 스크립트에 CrashCharacterToMonster 함수 작성.

캐릭터 스크립트의 OnTriggerEnter 함수에 게임매니저의 CrashCharacterToMonster 함수 호출구문 작성.

캐릭터 스크립트의 다양한 애니메이션 동작 관리를 위해 PlayerState라는 public enum 구문을 작성 (none, run, attack 등)

캐릭터 스크립트에 Public void PlayAnimation 함수 작성, 매개변수에 PlayerState state 작성.

함수 내부에 state에 따른 switch case 구문 작성. 각 조건 실행구문에 anim.Play 구문 작성.

게임매니저 스크립트의 CrashCharacterToMonster 함수에 character.PlayAnimation(Character.PlayerState.attack); 호출 구문 작성.

몬스터 스크립트에 public void SetMonsterSpeed(float setSpeed) 함수 작성.

함수 내부에 speed = setSpeed; 구문 작성.

게임매니저 스크립트의 CrashCharacterToMonster 함수에 monster.SetMonsterSpeed(0f); 구문 작성.

- 실습) 몬스터와 만났을 때에 배경이 계속 흘러가는 부분 수정해보기!

게임매니저에 백그라운드 스크롤 변수 선언하고

백그라운드 스크롤에 속도 조절 함수 선언해서 충돌 함수에서 값 넘겨서 호출하기.

1. **과제**

- 해상도 720 1280으로 맞춰오기

- 캐릭터 공격 10번하고나면 배경도 흘러가고 캐릭터도 달리고 몬스터도 흘러가게 작업해오기