1. 오프닝 / 인트로 영상

Video Player를 이용해서 적용한다.

Render Mode를 이용하여 Camera에 main Camera를 적용한다.

Loop를 해제하여 영상이 반복으로 재생되지 않도록 한다.

스크립트 작성 -> VideoController

오프닝 영상이 끝나면 인트로 영상이 재생되며, 인트로 영상이 끝나면 다음 씬으로 넘어간다.

영상 재생 중 어떠한 입력이 있을 경우 우측 하단에 Skip 텍스트가 3초간 출력되도록 하며,

이 시간 안에 추가적인 입력이 있을 경우 영상을 스킵하는 기능을 추가한다.

1. 타이틀(메인메뉴)

마우스 커서

Project Settings -> Default Cursor에서 변경이 가능하다.

게임시작 -> 프로필 선택 -> 다음 씬

Json을 활용해서 저장 및 불러오기

DataManager 스크립트를 작성

DontDestroyOnLoad를 사용하여 오브젝트가 파괴되지 않고 유지되도록 한다.

저장할 정보 -> 프로필 이름, 플레이어의 정보(마지막 저장 위치, 재화, 아이템 등)

데이터를 Json으로 변환 -> JsonUtillity.ToJson() -> (저장) Using System.IO를 사용하면 외부폴더에 접근할 수 있게 된다.

(불러오기) JsonUtillity.FromJson<>() -> Json형식으로 저장된 파일을 가져와 Data형식으로 변환

1. PlayScene

플레이어 이동

Input.GetKey 코드를 사용하여 입력을 받아 플레이어 캐릭터가 이동한다.

플레이어 공격을 Boolean 형식으로 판단하여 방향을 결정했다.

Ex) Input.GetKey(Keycode.righrtarrow) -> leftslah = false / rightslash = true 일 경우 오른쪽으로 공격

이러한 방식으로 방향을 결정하니 코드가 복잡해지고 키를 동시에 눌렀을 경우 버그가 발생하는 등의

여러가지 문제점들이 발생했다.

Boolean 형식으로 방향을 판단하지말고 Transform을 활용하여 플레이어의 위치를 판단하여 공격을 적용해보면 좀 더 간단하게

적용할 수 있을 것 같다. => 이 부분에 대해서는 많은 공부가 필요함

플레이어 점프

GetKeyDown과 GetKeyUp의 GravityScale의 값을 다르게 줘서 LongJump기능을 주었다.

JumpCount 변수를 추가해서 더블점프 구현이 가능할 것으로 생각된다.

대쉬

대쉬키를 눌렀을 경우 ex) Input.GetKeyDown(Keycode.C) 플레이어가 앞쪽으로 일정시간 빠르게 이동(speed, time)

이때(대쉬하는 동안; time) GravityScale 값을 0으로 설정하여 공중에서 대쉬도중 아래로 떨어지는 경우를 방지한다.

Animation 관련 -> 구체적인 계획을 가지고 애니메이션을 적용 해야할 필요성을 느꼈다.

Idle 애니메이션 만들고 -> Run 애니메이션 만들고 -> … 등 하나씩 추가하니까 중간중간 animator를 수정하기를 반복하다 보니

복잡해지고 적용이 잘 되지 않았다.

해서 애니메이션을 만들 때 어떠한 애니메이션들을 만들지, 어떠한 경우에 작동할지 등을 구체적으로 계획한 후에 애니메이션을 만들고 animator를 적용해야하는 것을 느꼈다.

SpriteFlip

이미지를 좌우 또는 상하 반전이 필요할 때가 있어 SpriteRenderer의 Flip을 활용하여 반전을 주었더니 콜라이더는 반전이 안되는 등의 문제점들이 발생하여 SpriteRenderer의 flip을 사용하지 않고 Scale값을 음수로 주어서 flip을 주었다.

Collider

BoxCollider를 사용하여 바닥을 구분함 -> isGrounded

가끔 바닥이 뚫리는 현상이 발생함 -> Boxcollider의 두께가 너무 얇고 플레이어가 점프 후 착지할 때의 속도가 너무 빠른 경우

충돌처리가 잘 되지 않는 것으로 판단된다.

바닥이 아닌 Collider에서의 부자연스러움 ex) 벽, 플랫폼 옆면 등 에서 미끄러지지 않고 끼는 현상

-> 마찰력을 0으로 만들어 줌으로서 해결이 가능하다.

Platform Effect2D 컴포넌트를 추가하여 캐릭터 Collider의 Used By Effector를 체크하고, Platform Effect2D의

Use One Way를 체크 해제 및 Use Side Friction을 체크 해제한다.

Monster

기본적으로 몬스터에 Collider를 붙여서 행동을 한다.

Ex) 굼벵이 -> 앞쪽으로 이동 / collider 범위를 벗어나면 flip

맹독충 -> collider 범위내에 player가 감지되면 player를 향해 이동 및 공격

Loading Scene

Scene 전환이 자연스럽게 하기 위해 씬 전환이 이루어 질 때 Loading Scene을 Load한 후 다음 Scene으로 넘어간다.

Coroutine함수를 활용하여 약 3s 정도 노출된 이후 다음 Scene으로 전환

보완할 점

\* 계획을 더 구체적으로 세울 것

계획서 작성시에는 전체적인 내용을 적더라도 실제로 프로그래밍 하기 전에 좀 더 구체적인 계획을 세운 후 개발을 시작한다.

예를들어 계획서 작성시에는 플레이어의 이동을 구현한다. 라고 작성하더라도 개발을 시작하기전에 어떤식으로 이동을 구현할지

또, 애니메이션은 어떤게 필요한지, 등 좀 더 구체적인 계획을 세운 후 개발을 시작한다.

\* 개발 순서

꼭 있어야 하는 부분(기능 위주) 먼저 개발을 할 필요가 있다.

예를 들어 배경은 검은색 화면만 있어도 플레이어 캐릭터와 적이 있으면 게임은 돌아가지만 캐릭터와 적이 없고 배경만 있으면

게임은 돌아가지 않는다.

이번 프로젝트중 가장 잘못되었던 부분이 이 부분인데 처음부터 TitleScene 완벽하게 구현하고 그 다음 맵 구현하고 플레이어

만들고 하는 방식으로 하니까 개발이 제대로 이루어지고 있다는 성취감도 들지 않을뿐더러 마음만 조급해지는 것을 느꼈다.

\* Resource 정리

이번 프로젝트에서 resource가 너무 많아 힘들었다. 그래서 구현할 때 필요한 부분을 찾아 개발하고 다시 필요한 resource를

찾아 개발하고 하는 방식으로 하다보니 resource 찾는데 쓸데없이 시간을 많이 사용하였다.

다음 개발을 진행할 경우 시간이 오래 걸리더라도 개발 전에 사용할 resource를 정리한 후 개발을 시작하자

또한 resource를 정리하다 내가 어떤것들을 구현할지 또, 어떻게 구현할지 등을 생각할 수 있을 것이다.

\* 개발 계획 지키기

프로젝트를 진행하면서 개발 계획이 하나도 지켜지지 않았다.

계획을 세우면서 이걸 구현하는데 시간이 얼마나 걸릴지 잘 몰라 엉성하게 계획한것도 있지만, 그럼에도 너무 지켜지지 않았다.

계속해서 지켜지지 않다보니 나중에는 아예 신경도 안쓰고 하다가 결국에는 포기하는 경우도 생겼다.

다음 개발부터는 위에 작성했듯이 좀 더 구체적인 계획을 세우고 개발계획을 최대한 지키면서 개발을 할 필요를 느꼈다.

\* 다른 사람의 코드를 적극 활용

혼자서 공부한다는 생각으로 구글검색도 해보고 이해안되면 유튜브도 보면서 개발을 했는데 이게 나쁘지는 않지만 하나하나 전부

이런 방식으로 하기에는 시간이 너무 많이 걸리며 또한, 이해 안되는 부분에 대해서는