

UNITY 2D GAME

# COOKIE RUNNER

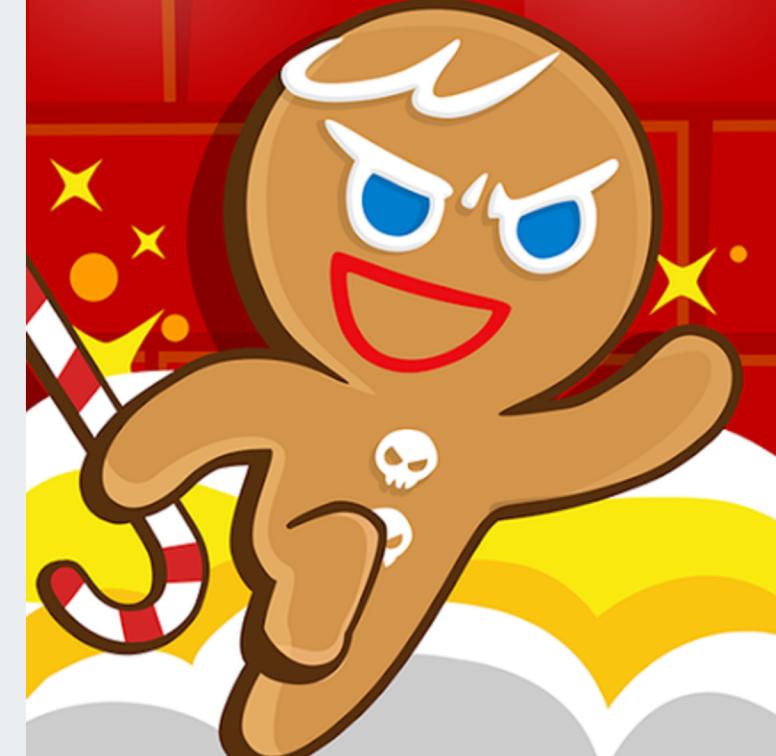
쿠키런 - 오븐브레이크 모작



TEAM 마개조(9조)

팀장 임지환

팀원 김영재 이건우 이유진 정성욱



01

## INTRODUCTION

게임 소개

02

## DEMONSTRATION

게임 시연

03

## CORE FEATURES

핵심 기능

04

## TROUBLESHOOTING

트러블 슈팅

05

## CONCLUSION

마무리

# INTRODUCTION

장르

장애물을 피하며 멀리 달리는 러닝 장르 게임

언어

UNITY, C#

지원 플랫폼

PC (UNITY 에디터) 및 ANDROID

주요 특징

1. 자동 달리기와 점프/슬라이드 입력 지원
2. 지속적인 맵 생성으로 무한 달리기 구현
3. 장애물 순환을 통한 게임 난이도 유지
4. UGS 연동
  - 플레이어 인증 시스템 구축
  - 데이터 저장/불러오기 지원
  - 랭킹 시스템을 통한 경쟁 요소 추가



**Unity**



# DEMONSTRATION

## 게임 시연

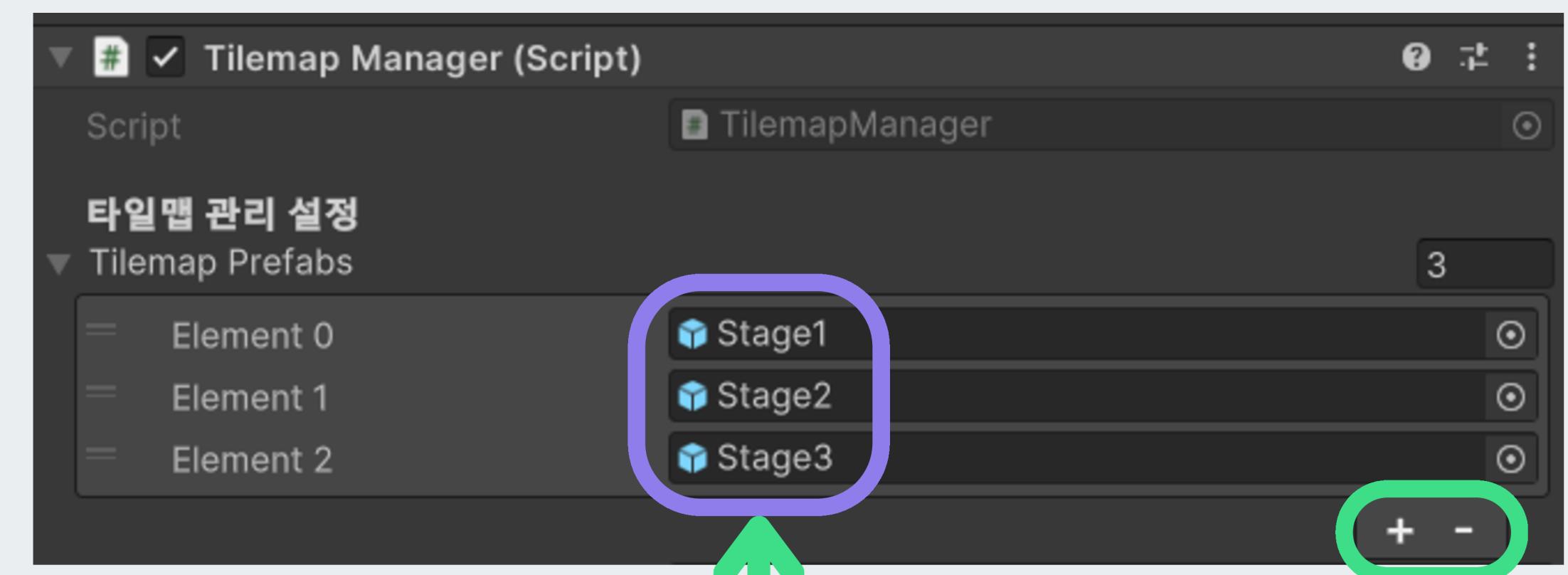
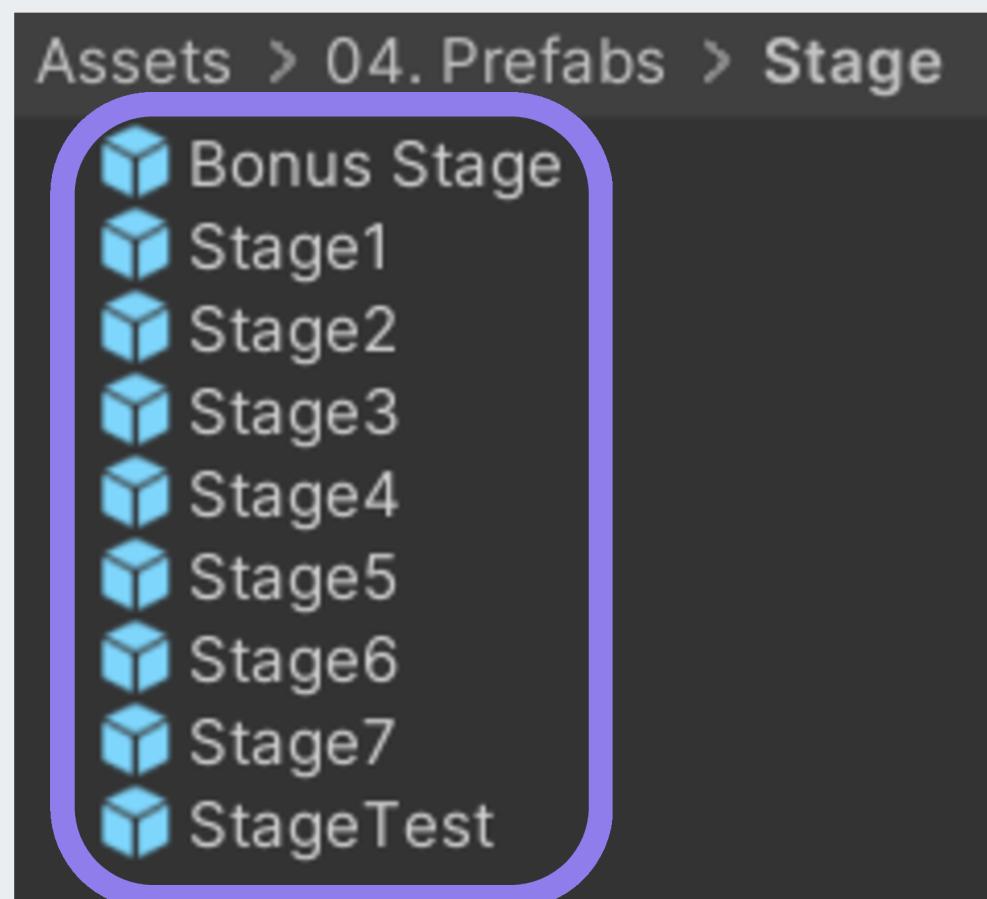
Game Play Video

주요 기능, 관련 내용을 중심으로 영상을 제작하여 여기 부착

1. 플레이어의 이동과 점프/슬라이드 시연
2. 맵 자동 생성 및 연결 기능
3. 장애물과의 충돌 시 체력 감소
4. 아이템 획득 및 효과 적용
5. UGS 연동 시연: 신기록 → 랭킹 확인

# CORE FEATURES

## 1. 맵 반복 생성



1. 원하는 맵 갯수 설정

2. 배치하고 싶은 맵 순서대로 배치

## CORE FEATURES

1. 맵 반복 생성



이전 맵 삭제

새로운 맵 생성

Stage Prev

Stage Next

Current Stage

# CORE FEATURES

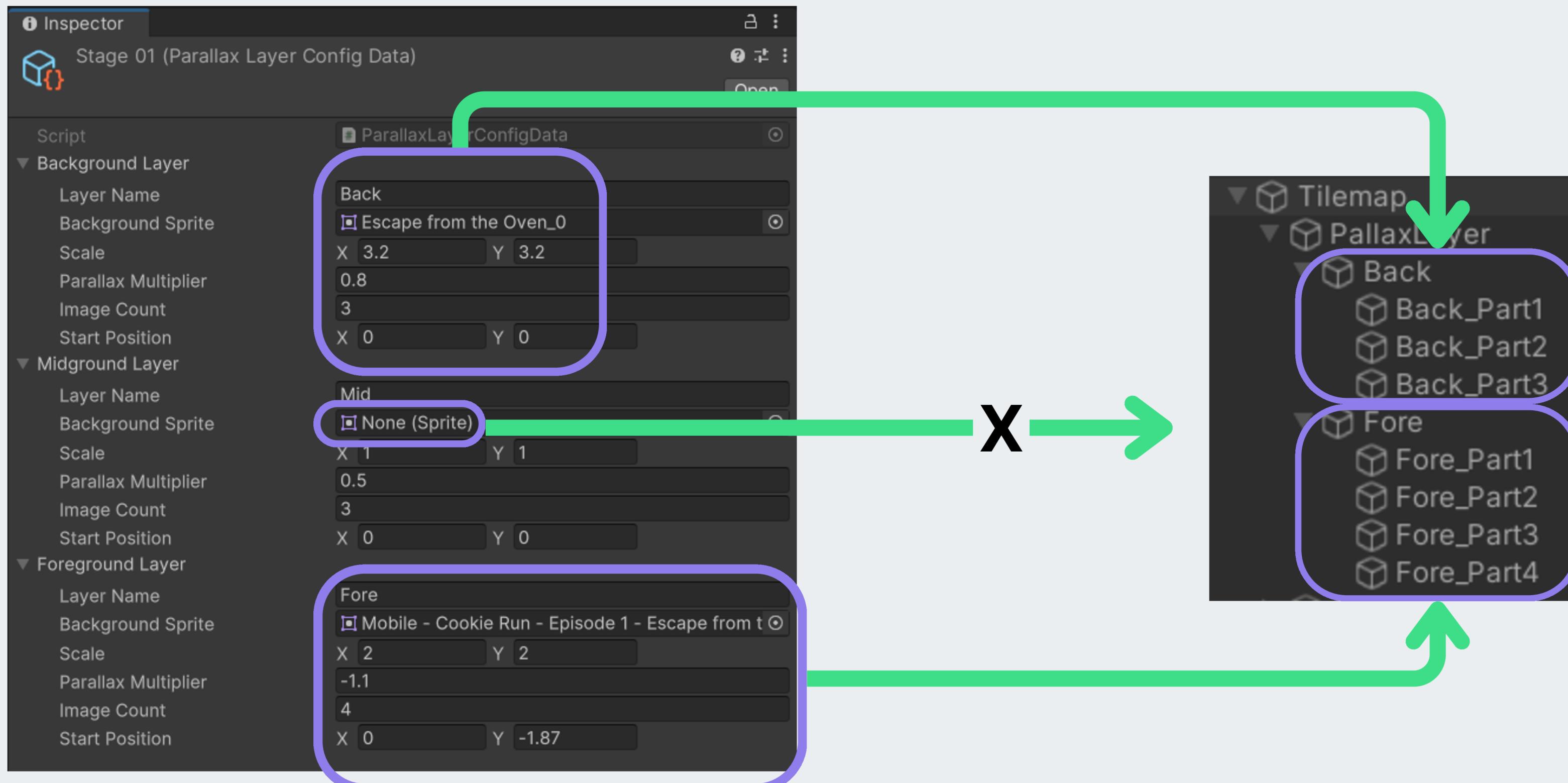
## 2. Parallax 배경 효과

배경의 여러 레이어가 플레이어의 움직임에 따라 서로 다른 속도로 이동하도록 만들어 깊이감과 입체감을 주는 기법



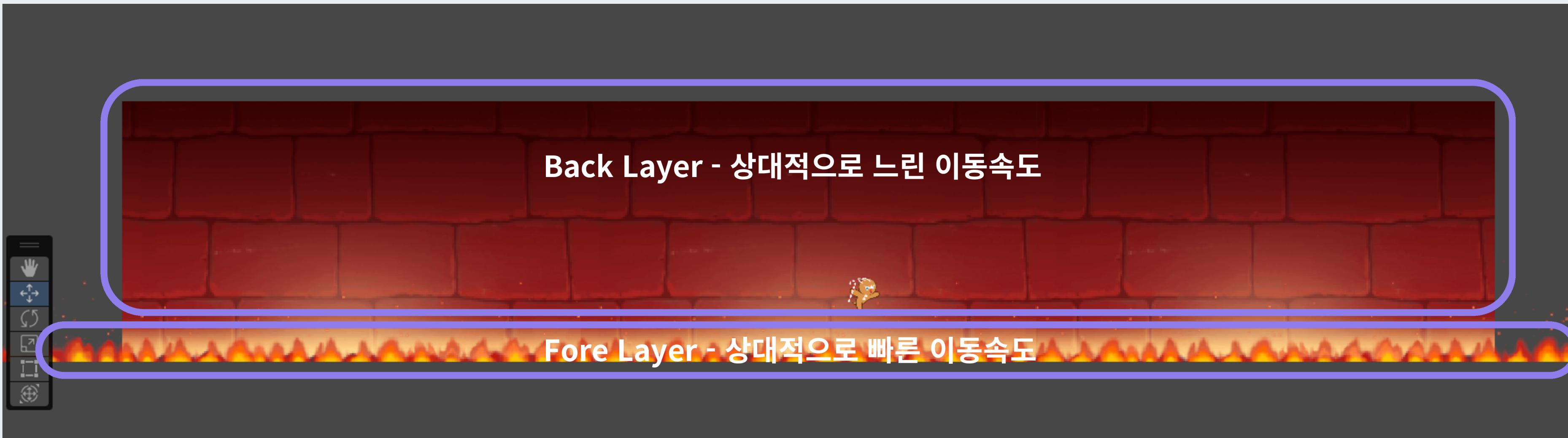
# CORE FEATURES

## 2. Parallax 배경 효과



# CORE FEATURES

## 2. Parallax 배경 효과



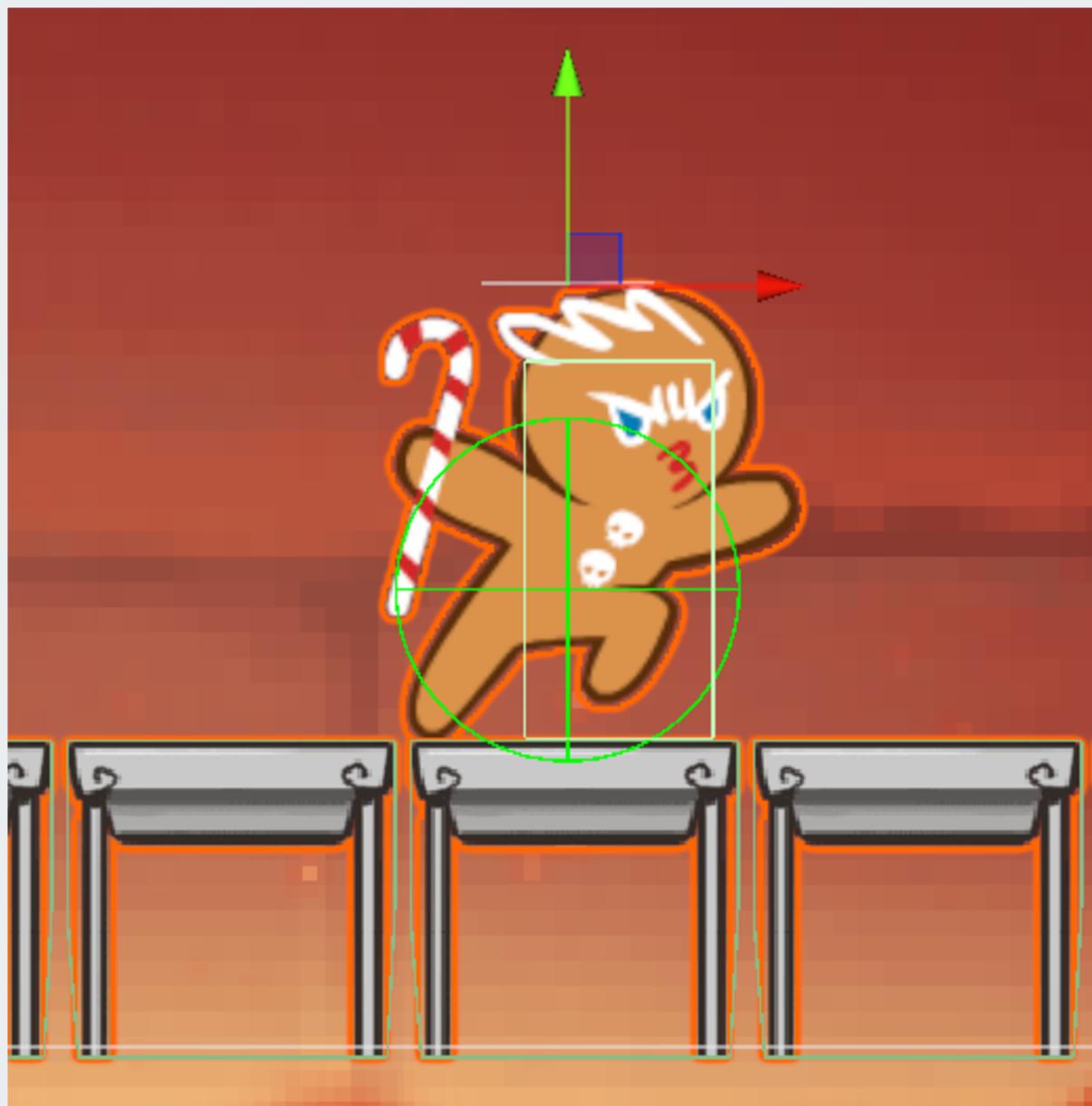
상대적으로 뒤에 있는 레이어의 움직이는 속도를 늦춰 깊이감을 부여

각 레이어는 vector 이미지를 사용하여 플레이어의 위치에 따라 지나간 이미지가 앞으로 움직임이며 반복됨

# CORE FEATURES

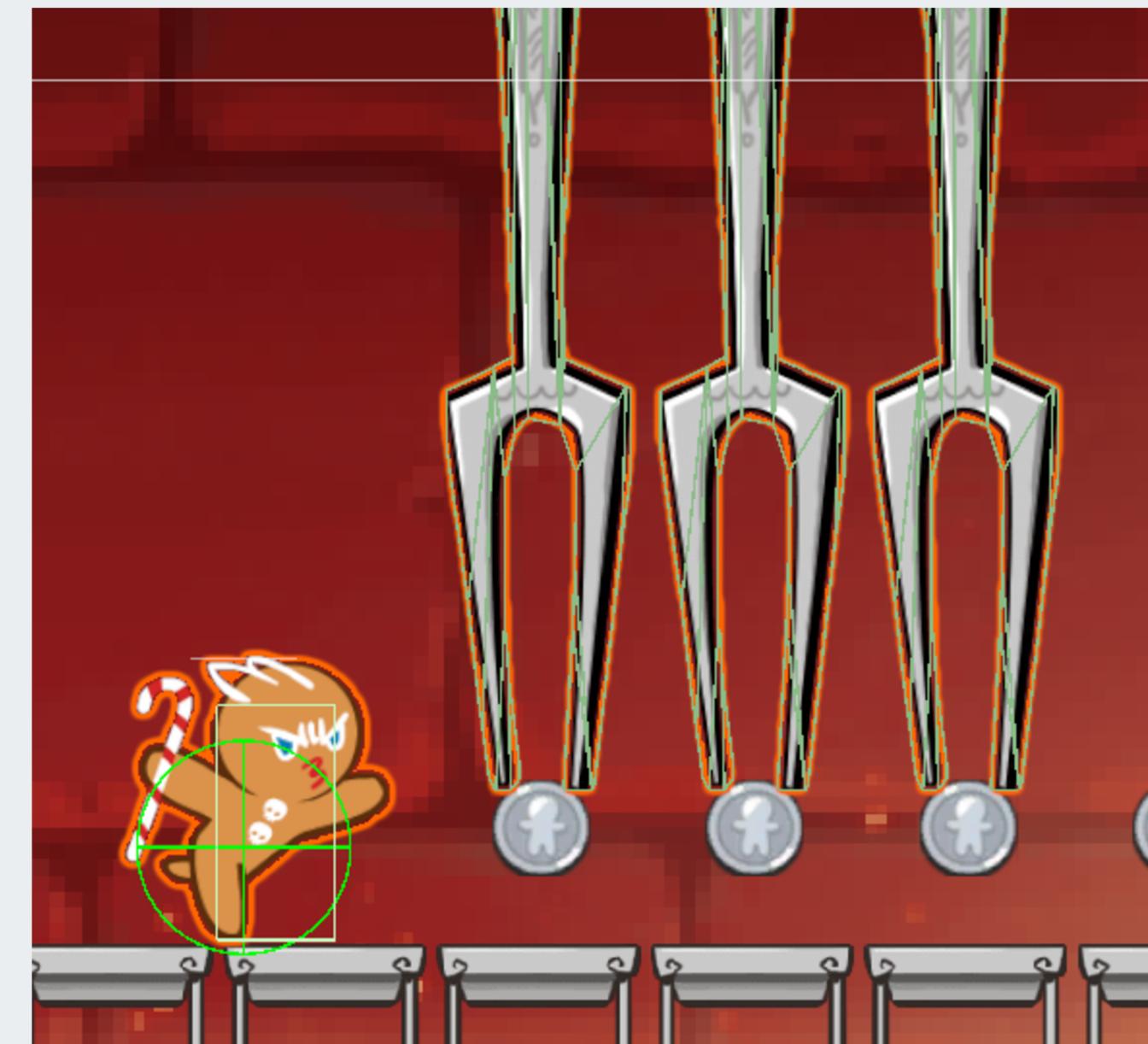
## 3. 장애물 충돌 처리

플레이어



2가지 충돌체 구성 - 장애물 충돌체, 바닥 충돌체

장애물



보다 정밀한 충돌을 위한 polygon Collider 2d 사용

# CORE FEATURES

## 3. 장애물 충돌 처리

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.collider.CompareTag("ObsColider"))
    {
        PlayerController playerController = collision.collider.GetComponentInParent<PlayerController>();
        PlayerMovement playerMovement = collision.collider.GetComponentInParent<PlayerMovement>();

        Collider2D collider = GetComponent<Collider2D>();

        if (playerController != null)
        {
            if (playerMovement.isItemInvincible == false)
            {
                playerController.OnHit(data.damage);
            }
            else
            {
                isKicked = true;
                rb.bodyType = RigidbodyType2D.Dynamic;
                rb.AddForce((Vector2)collision.gameObject.transform.position * data.power);
                collider.isTrigger = true;
                Destroy(collider);
            }
        }
    }
}
```

충돌체가 장애물 tag인 경우만 처리

무적이 아닐 경우 Hit!

무적일 경우 특정 방향으로 힘 추가

# CORE FEATURES

## 4. UGS 연동

 CookieRuner  
2025년 2월 25일에 생성됨

[프로젝트 이전](#) [프로젝트 아카이브](#)

**프로젝트 세부 정보**

프로젝트 이름	CookieRuner	
프로젝트 아이콘		
프로젝트 ID	7eb14b05-b802-434d-b15a-58c828f9d852	
Google 라이선스 키 인앱 구매 설정	설정되지 않음	
COPPA 설정	지정하지 않음	
프로젝트 가시성	모든 조직 구성원	

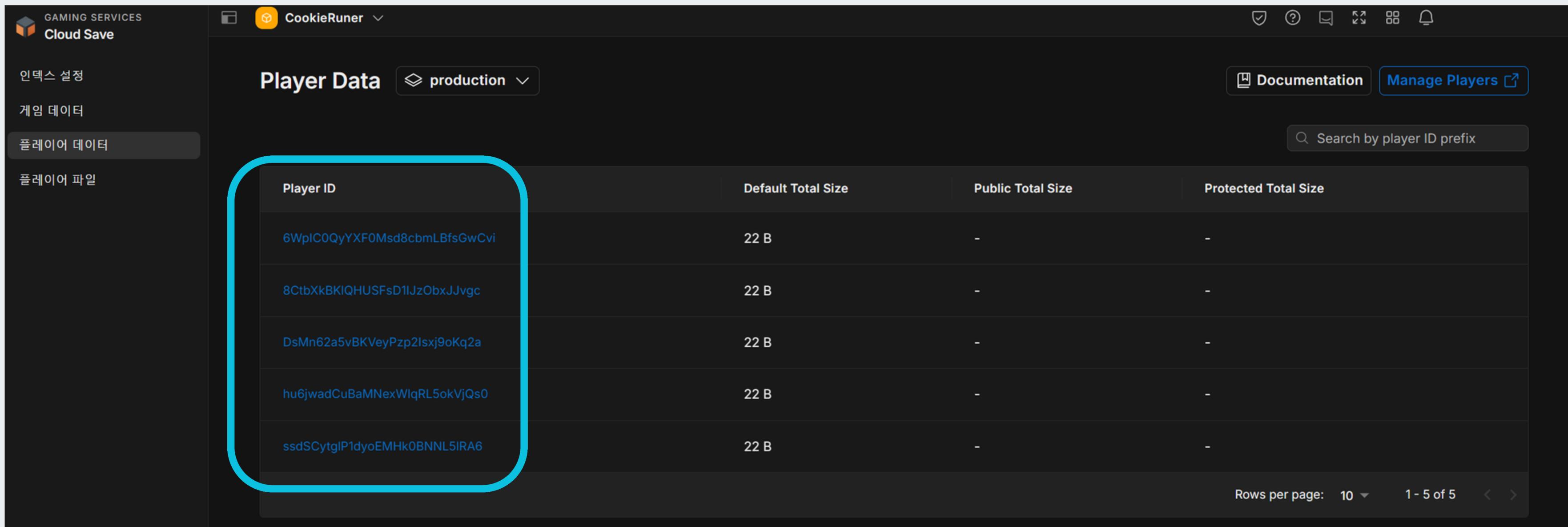
**서비스** [프로젝트 구성원](#) [개발 환경](#)

활성  
지난 30일 동안 프로젝트에서 사용한 서비스입니다.

GAMING SERVICES Cloud Save	클라우드에 플레이어 저장 데이터 저장	
GAMING SERVICES Leaderboards	게임에 우호적인 경쟁 추가	
GAMING SERVICES Player Authentication	의명, 플랫폼별 또는 커스텀 플레이어 인증	

# CORE FEATURES

## 4. UGS 연동



The screenshot shows the AWS Cloud Save Player Data interface for the game "CookieRun". The left sidebar has tabs for "Cloud Save", "Index Settings", "Game Data", and "Player Data" (which is selected). The main area shows a table with the following data:

Player ID	Default Total Size	Public Total Size	Protected Total Size
6WpIC0QyYXF0Msd8cbmLBfsGwCvi	22 B	-	-
8CtbXkBKIQHUSFsD1IJzObxJJvgc	22 B	-	-
DsMn62a5vBKVeyPzp2lsxj9oKq2a	22 B	-	-
hu6jwadCuBaMNexWIqRL5okVjQs0	22 B	-	-
ssdSCytglP1dyoEMHk0BNNL5IRA6	22 B	-	-

A blue rounded rectangle highlights the first row's "Player ID" column. The bottom right corner of the interface shows pagination: "Rows per page: 10" and "1 - 5 of 5".

Player Authentication을 활용한 인증 토큰 발급

# CORE FEATURES

## 4. UGS 연동

```
Unity 메시지 | 참조 0개
public async void Start()
{
    // 유니티 서비스 초기화
    await UnityServices.InitializeAsync();
    Debug.Log("유니티 서비스 초기화 완료.");

    // 익명 로그인 (로그인되어 있지 않다면)
    if (!AuthenticationService.Instance.IsSignedIn)
    {
        try
        {
            await AuthenticationService.Instance.SignInAnonymouslyAsync();
            Debug.Log("로그인 성공, 플레이어 ID: " + AuthenticationService.Instance.PlayerId);
        }
        catch (Exception ex)
        {
            Debug.LogError("로그인 실패: " + ex.Message);
            return;
        }
    }

    // Cloud Save에서 플레이어 데이터 불러오기
    bool loaded = await LoadPlayerDataWithNewtonsoftAsync();
    if (!loaded)
    {
        // 저장된 데이터가 없으면 기본값 할당
        playerDataSO.highScore = 0;
        Debug.Log("저장된 플레이어 데이터가 없습니다. 기본 데이터 생성.");
    }
    else
    {
        Debug.Log($"불러온 데이터 - 최고 점수: {playerDataSO.highScore}");
    }
}
```

```
public async Task<bool> LoadPlayerDataWithNewtonsoftAsync()
```

System.Threading의 스레드 풀에서 block 상태인 스레드를 Task로 부여 받아 비동기적으로 처리. 이로 인해 메인 스레드는 blocking 되지 않아 응답성이 존재. 코드 흐름은 blocking 된 상태

# CORE FEATURES

## 4. UGS 연동

### 플레이어 데이터 관리

The screenshot displays a dark-themed database interface with three nested document structures:

- Root Document (Default Access):** Contains fields for **Default**, **Public**, and **Protected**. It includes a search bar and a red-bordered "Delete Default Data" button.
- Child Document (Default Access):** Contains fields for **Default**, **Public**, and **Protected**. It includes a search bar and a red-bordered "Delete Default Data" button.
- Grandchild Document (Default Access):** Contains fields for **Default**, **Public**, and **Protected**. It includes a search bar and a red-bordered "Delete Default Data" button.

Each document structure has a "Default Data" section and a "PlayerData" section. The "PlayerData" section contains a table with columns: **Key**, **Value**, **Created**, **Last Saved**, **Write Lock**, and **Size**.

Key	Value	Created	Last Saved	Write Lock	Size
PlayerData	"{"highScore":1916}"	Feb 26, 2025, 10:59 AM GMT+9	Feb 26, 2025, 4:59 PM GMT+9	034553ee8fd24f3e63b1ba...	22 B

At the bottom right of the interface, there is a page navigation indicator showing "1 - 1 of 1".

# CORE FEATURES

## 4. UGS 연동

플레이어 랭킹 관리 - 높은 점수 순으로 정렬, 버킷(일정한 플레이어 모음), 티어(일정 점수 단위로 분류) 없음

The screenshot shows a game's configuration interface for a leaderboard. On the left, there's a sidebar with various settings like Details, Name, ID, Date created, Date modified, Last reset, and Archived versions. The main area is titled "Configuration" and contains fields for Sort order, Update strategy, Reset schedule, Buckets, and Tiers. A blue rounded rectangle highlights the "Sort order" field, which is set to "Highest to lowest".

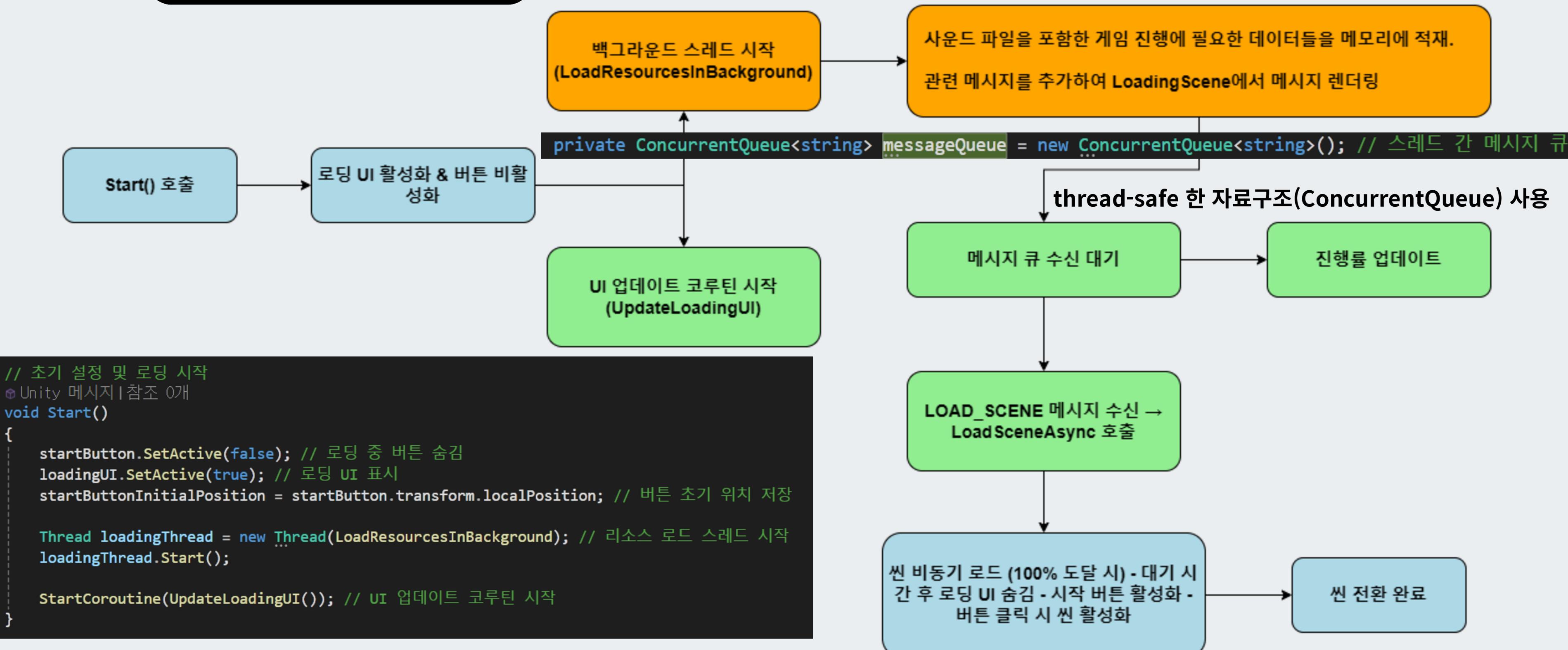
Details	Name	Ranking	edit
These are the details of your leaderboard.			
ID		Ranking	
Date created	February 25th, 2025, at 02:05 UTC		
Date modified			
Last reset			
Archived versions			

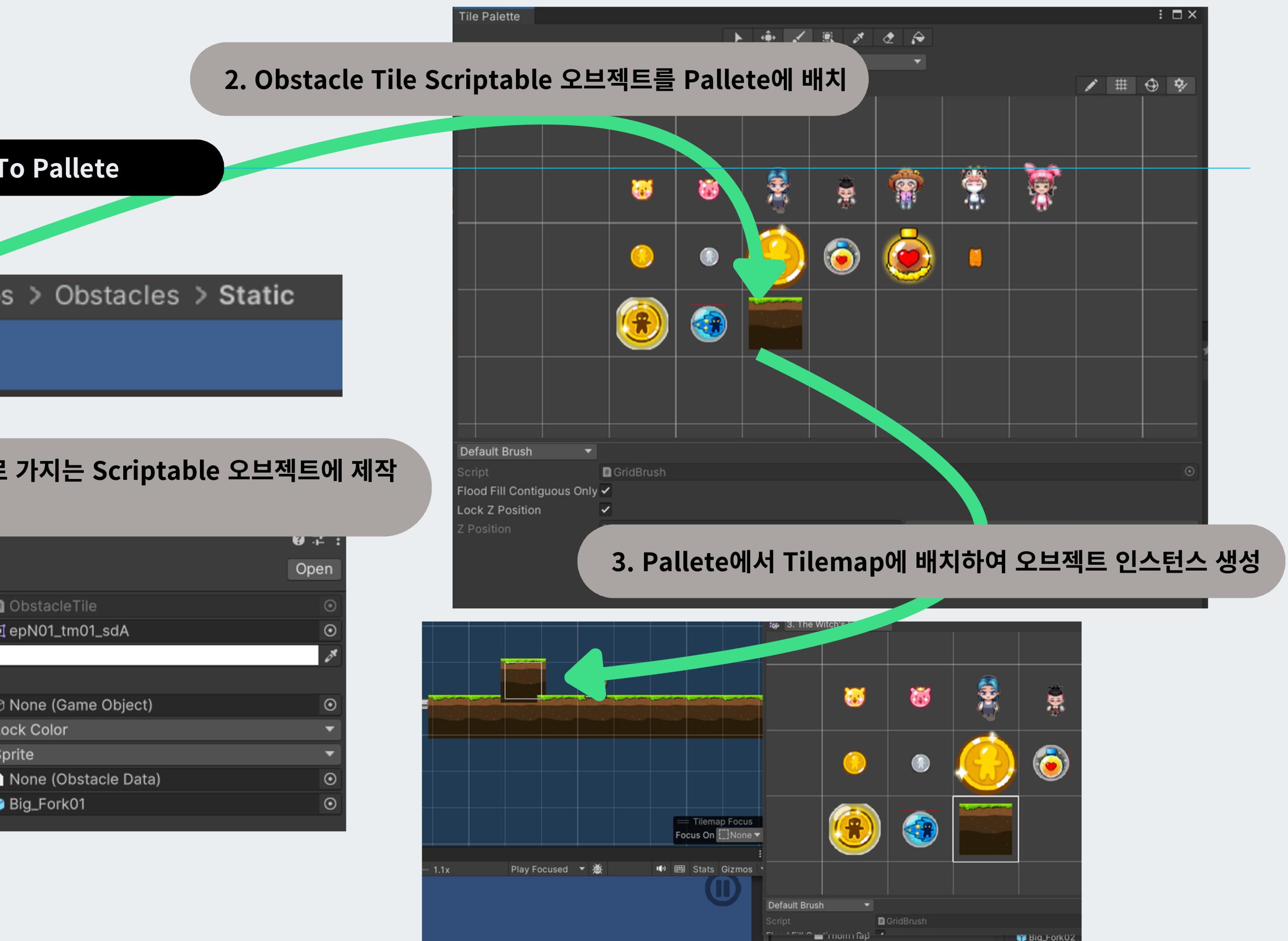
Configuration	Sort order	edit
These are the configurations of your leaderboard.	Highest to lowest	
Update strategy	Best score	
Reset schedule	None	
Buckets	None	
Tiers	None	

# CORE FEATURES

## 5. Loading Scene



# CORE FEATURES



# CORE FEATURES

## 6. Prefab To Pallete

```
public class ObstacleTile : Tile
{
    public ObstacleData obstacleData;

    // 타일맵에 배치할 오브젝트 프리팹 참조
    public GameObject obstaclePrefab;

    참조 0개
    public override bool StartUp(Vector3Int position, ITilemap tilemap, GameObject go)
    {
        // 에디터 모드에서 실행 방지 및 obstacleData와 obstaclePrefab 미할당 시 처리 중단
#ifndef UNITY_EDITOR
        if (!Application.isPlaying)
        {
            return base.StartUp(position, tilemap, go);
        }
#endif
        // obstaclePrefab이 없고, obstacleData도 없다면 경고 후 종료
        if (obstaclePrefab == null && obstacleData == null)
        {
            Debug.LogWarning($"[ObstacleTile] obstacleData와 obstaclePrefab 둘 다 할당되지 않음");
            return base.StartUp(position, tilemap, go);
        }

        // Tilemap 캐스팅 및 유효성 검사
        Tilemap actualTilemap = tilemap.GetComponent<Tilemap>();
        if (actualTilemap == null)
        {
            Debug.LogError($"[ObstacleTile] Tilemap 컴포넌트를 찾을 수 없습니다.");
            return base.StartUp(position, tilemap, go);
        }
    }
}
```

Tile이 Tilemap에 배치될때 호출되는 StartUp 함수  
를 오버라이딩하여 설정해둔 Prefab을 대신 배치

```
GameObject obstacleObject = null;
// 설정해둔 프리팹이 있으면 Instantiate
if (obstaclePrefab != null)
{
    Debug.Log("프리팹이 할당되어 있습니다. 인스턴스화를 시작합니다.");
    obstacleObject = GameObject.Instantiate(obstaclePrefab);
    obstacleObject.name = obstaclePrefab.name;
    Debug.Log("프리팹 인스턴스화가 완료되었습니다. 생성된 오브젝트 이름: " + obstacleObject.name);

    // 인스턴스화된 오브젝트 혹은 하위 오브젝트에 Obstacle 컴포넌트가 있는지 검사
    Obstacle obstacleComponent = obstacleObject.GetComponentInChildren<Obstacle>();
    if (obstacleComponent == null)
    {
        Debug.LogError("오류: 인스턴스화된 프리팹에서 Obstacle 컴포넌트를 찾을 수 없습니다.");
    }
}
```

# TROUBLESHOOTING

## >>> 1. Github 유니코드, UTF-8 깨짐 문제

```
@@ -72,17 +72,6 @@  
72 72     switch (data.Type)  
73 73     {  
74 74         case ItemType.Score:  
75  
76  
77             // ScoreManager.Instance.AddScore(data.score);  
78             Debug.Log($"?맷넓?럾??: {data.score}");  
79  
80             ScoreManager.Instance.AddScore(data.score);  
81             Debug.Log($"?점?넓?.Lerp?: {data.score}");  
82  
83  
84             // ScoreManager.Instance.AddScore(data.score);  
85             Debug.Log($"?점?넓?.Lerp?: {data.score}");  
86             ScoreManager.Instance.AddScore(data.score);  
87             Debug.Log($"스코어 획득: {data.score}");  
88             75
```

### 문제 원인

- Git 인코딩 미설정으로 코드 파일 내 한글 및 특수문자 깨짐 현상 발생

### 해결 과정

- 해당 스크립트들을 이전 commit에서 가져와 복구
- Git 전역 설정에 UTF-8 인코딩 추가  
: git config --global i18n.commitEncoding utf-8

### 추가 설명

X

# TROUBLESHOOTING

## >>> 2. 게임 일시정지 후 씬 전환 시 애니메이션 멈춤

### 문제 원인

- Time.timeScale = 0 사용 후 씬 전환 시 애니메이션 재생 멈춤 현상 발생



### 해결 과정

- 씬 전환 직전에 Time.timeScale = 1로 초기화 처리

### 추가 설명

- TimeScale은 게임 내 모든 시간 기반 연산에 영향을 미침

가능하면 gif 이미지를 넣으면 좋을듯

# TROUBLESHOOTING

## »»» 3. 장애물 충돌체 설정 문제

### 문제 원인

- BoxCollider2D 사용 시 충돌 처리 오류 발생
- PolygonCollider2D 적용 시 Collider 형태가 잘못 생성되어 오각형 형태로 비정상 적용

충돌체 오각형 나오는 이미지 삽입

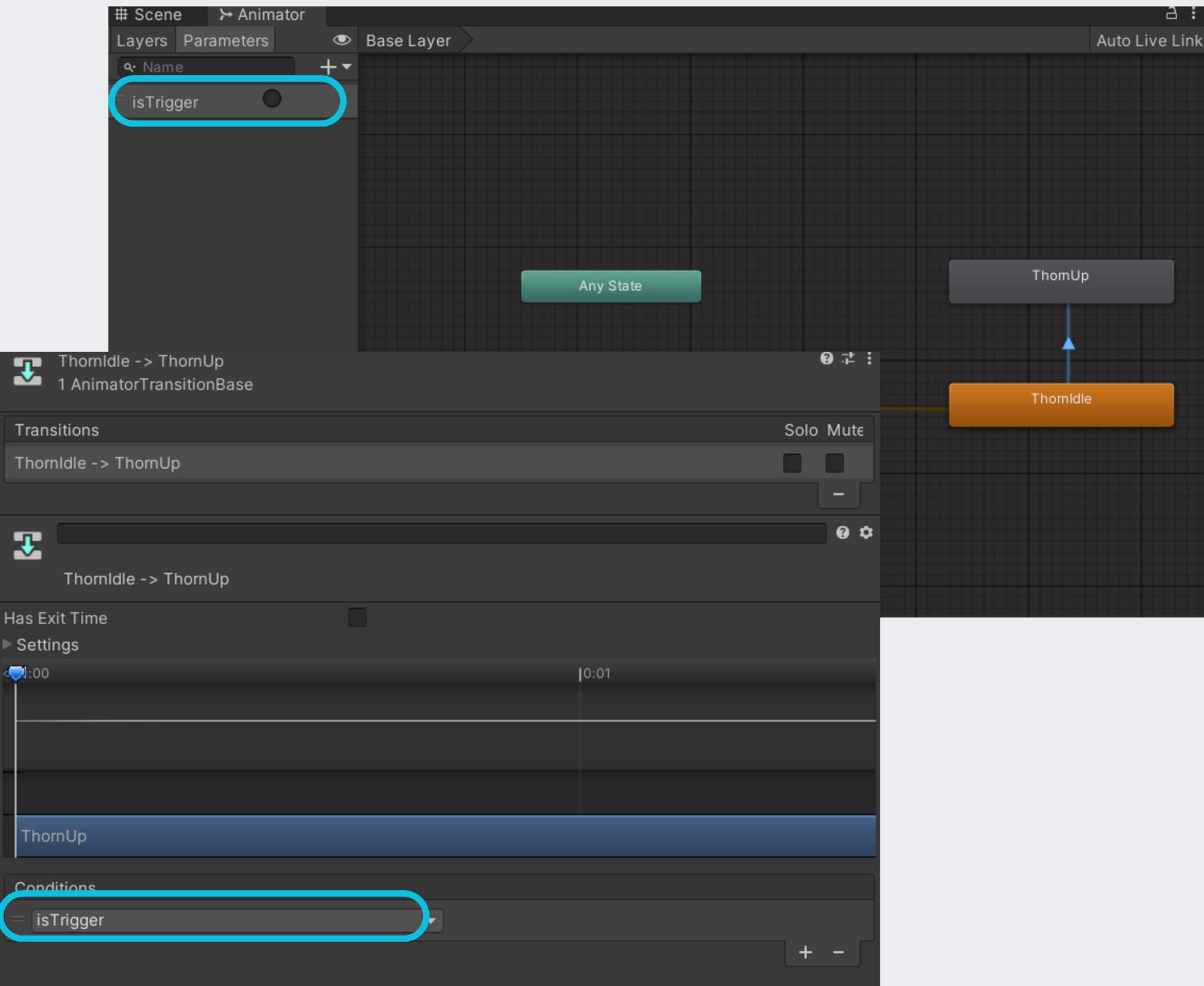
### 해결 과정

- 스프라이트 렌더러에 이미지 적용 이후 기존 Collider 제거 후 재생성시 정상 작동이 확인됨

정상적인 polygon 충돌체 나오는 이미지 삽입

# TROUBLESHOOTING

## >>> 4. 움직이는 장애물 생성 문제



### 문제 원인

- 밑에서 위로 올라오는 장애물이 작동하지 않는 현상 발생

### 해결 과정

- Animator에서 Condition 설정 누락 확인
- Animator에 적절한 변수 추가 후 Transition에 Condition 설정

# TROUBLESHOOTING

## >>> 5. 화면 비율 및 스케일 문제

### 문제 원인

- Canvas 스케일 모드를 Constant Pixel Size로 사용 시 해상도 변화에 따른 UI 깨짐 발생

### 해결 과정

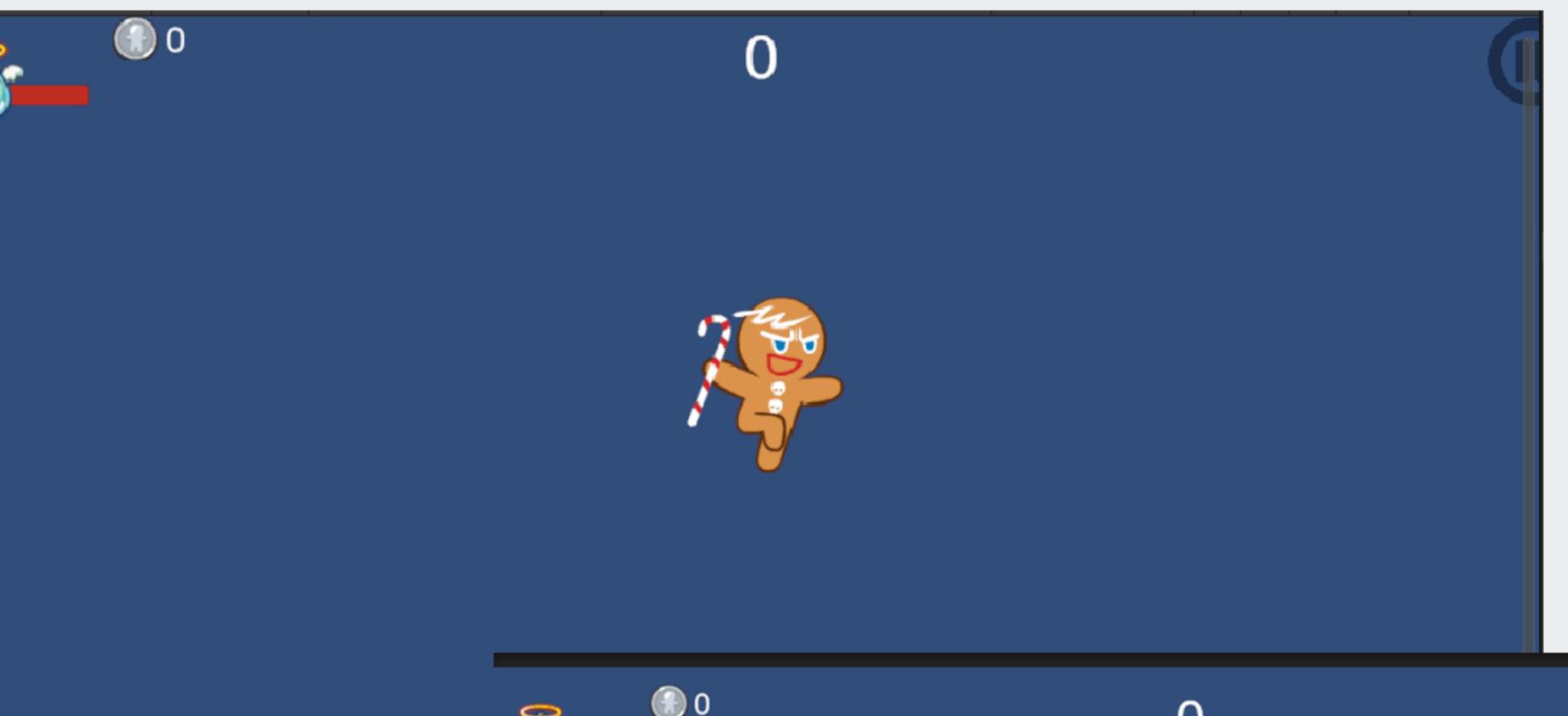
- Scale With Screen Size 옵션을 사용하니 다른 사람의 화면에서 도 동일한 비율로 UI가 나타나는 것을 확인

### 추가 설명

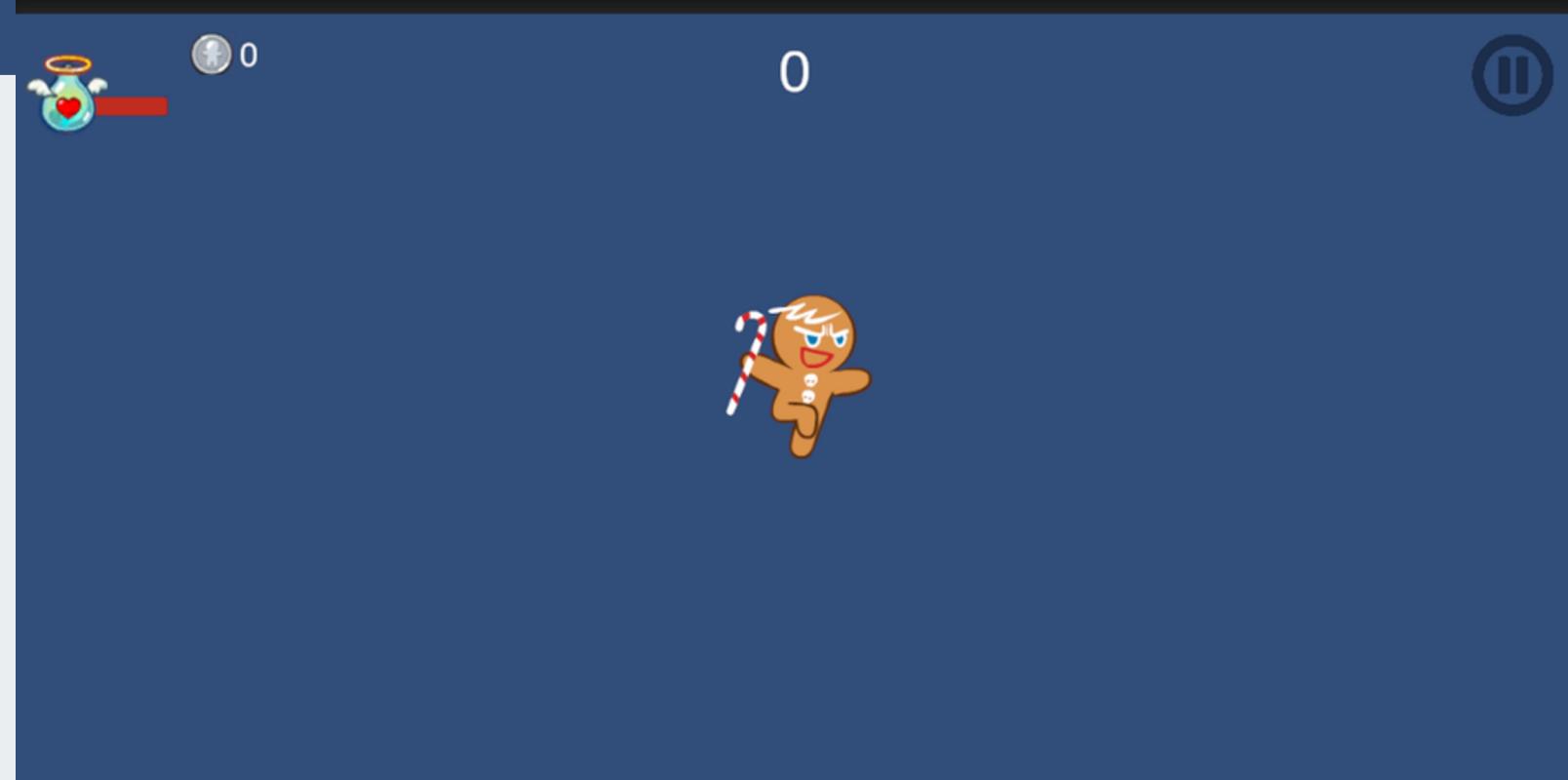
- Canvas의 Render Mode를 Screen Space - Overlay, Camera로 사용할 때, Constant Pixel Size 옵션은 모든 해상도에서 UI 요소의 크기가 픽셀 단위로 고정되기 때문에 해상도가 다르면 비율이 달라 보일 수 있음.

- Scale With Screen Size 옵션은 기준 해상도(Reference Resolution)를 설정하고 Screen Match Mode를 선택해 화면 크기에 따라 UI가 자동으로 조정되기 때문에, 다른 사람의 화면에서도 동일한 비율로 UI가 나타남

해결 전



해결 후



# TROUBLESHOOTING

## »» 6. 물리 연산 및 트리거 호출 관련 문제

### 문제 원인

- Update에서 Rigidbody를 사용한 물리 연산 및 트리거 호출 시도
- Unity의 함수 호출 주기(Execution Order)를 참고하지 않아 물리 처리와 애니메이션 전환 간 타이밍 불일치 문제 발생
- 트리거를 호출해도 애니메이션 전환이 무시되거나 지연됨

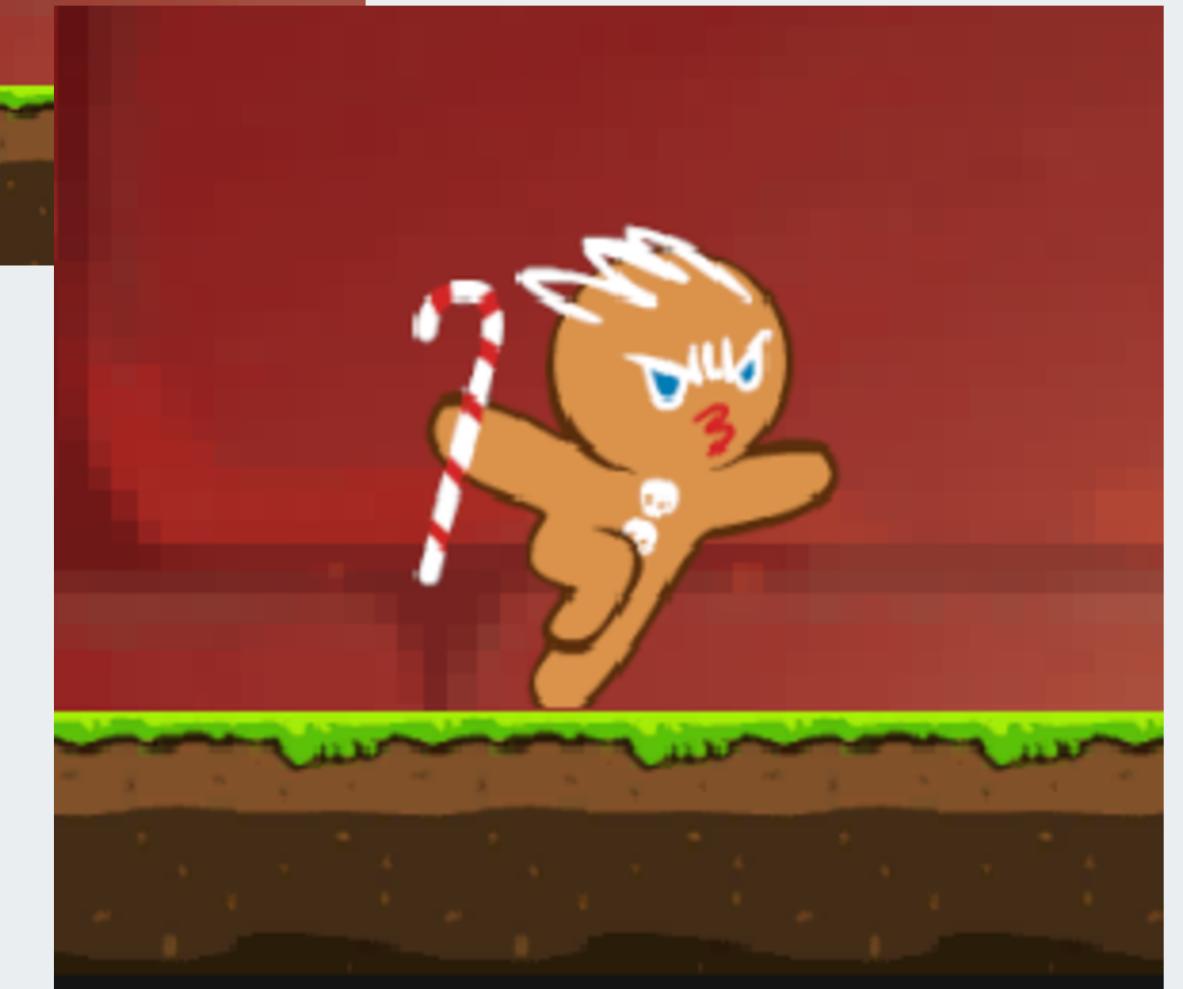
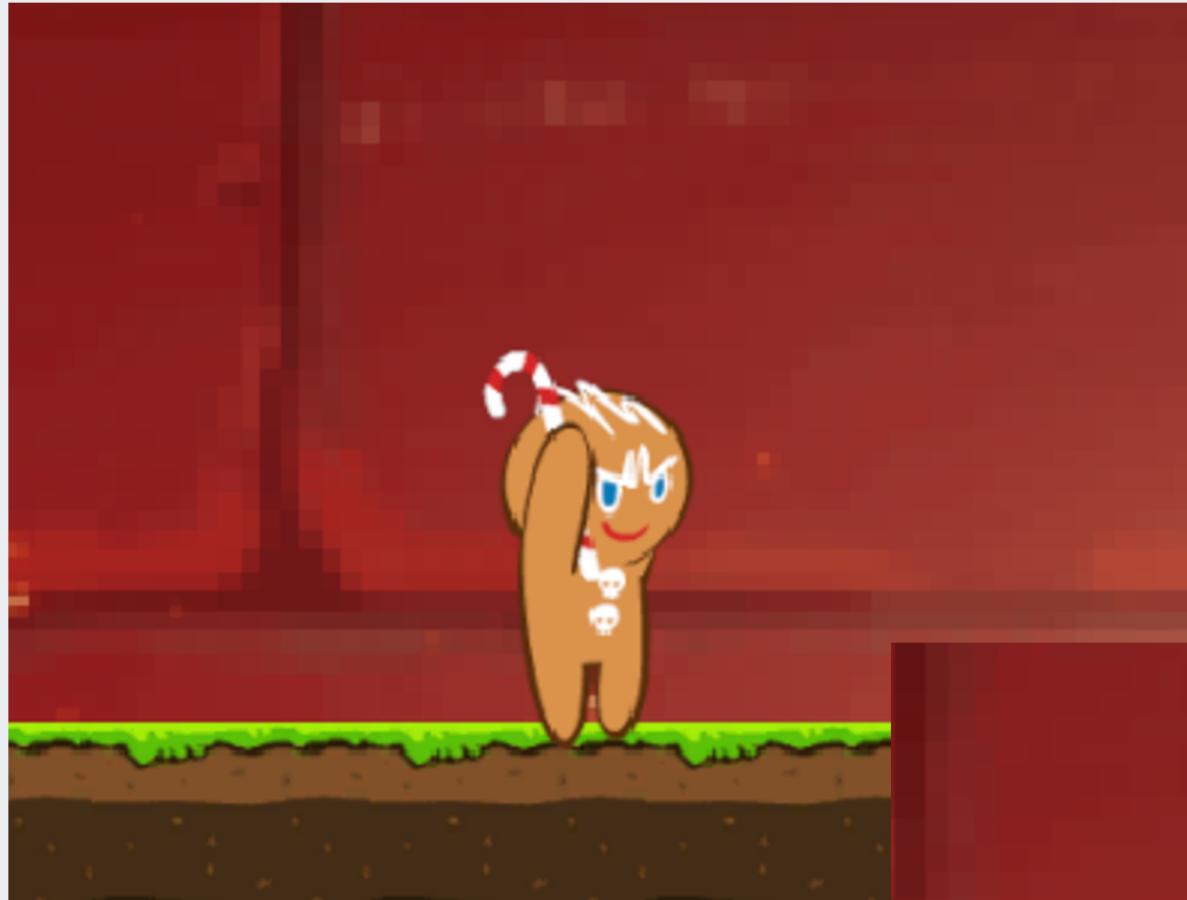
### 해결 과정

- 모든 물리 관련 코드와 트리거 호출 코드를 FixedUpdate로 이동

### 추가 설명

- FixedUpdate는 물리 연산 전용 메서드로, Rigidbody 관련 연산 및 트리거 호출 작업을 처리하도록 권장됨
- deltaTime이 달라지는 Update와 다르게 FixedUpdate는 일정한 호출 간격으로 인해 물리 처리 및 트리거 호출의 일관성을 확보함
- <https://docs.unity3d.com/kr/current/Manual/ExecutionOrder.html>

땅에 닿았고, Animator에서 trigger가 변함을 확인 했음에도 transition이 적용되지 않음



모든 강체 관련 코드를 FixedUpdate로 옮긴 후 올 바르게 작동됨을 확인

# THANK YOU

TEAM 마개조(9조)

팀장 임지환

팀원 김영재 이건우 이유진 정성욱

