게임프로그래밍
유니티 게임(계단 오르기)

2020875029 신성우

초기 게임(단순 계단 오르기)

추가 내용

- 게임 메인/클리어/종료 화면
- 자동으로 계단 무한 생성
- 떨어지면 바로 게임 종료(데드존)
- 자동으로 코인 및 방해물 생성
- 코인 획득 시 스코어 증가
- 스코어 높아질 수록 속도 높아짐
- 방해물에 부딪히면 게임 종료
- BGM

메인 화면

클리어 화면

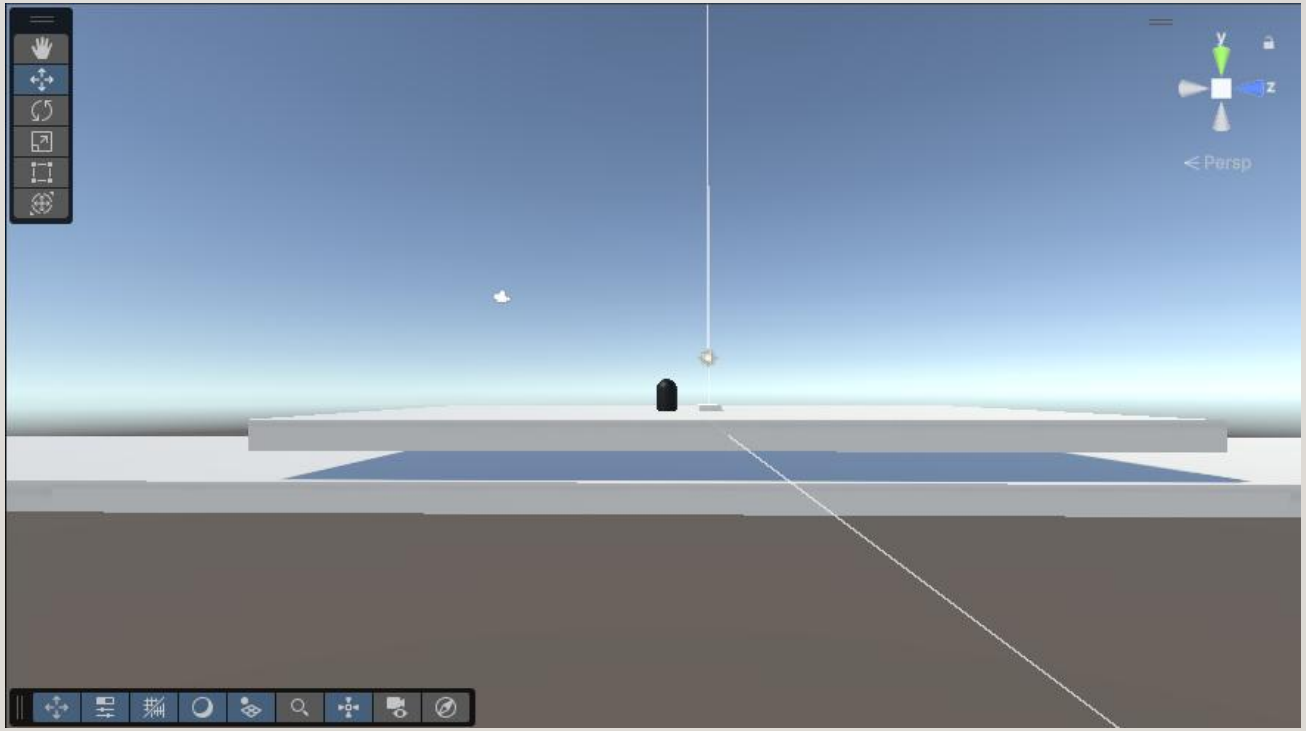
종료 화면

자동으로 계단 무한 생성

```
using System.Collections.Generic;
using UnityEngine;

public class InfiniteStairManager : MonoBehaviour
{
    public GameObject stairPrefab;      // 계단 프리팹
    public Transform player;            // 플레이어 Transform

    public int initialStairCount = 30;  // 처음에 몇 칸 생성할지
    public float stepHeight = 0.3f;     // 한 계단 올라갈 때 Y 증가량
    public float stepDepth = 1f;        // 한 계단 앞으로 갈 때 Z 증가량

    public float spawnDistanceAhead = 20f;   // 플레이어 앞쪽 몇 m까지 계단을 유지할지
    public float deleteDistanceBehind = 10f; // 플레이어 뒤쪽 얼마 이상 멀어지면 삭제할지

    private Queue<GameObject> stairs = new Queue<GameObject>();
    private Vector3 nextSpawnPos;

    public GameObject coinPrefab;     // 코인 프리팹
    [Range(0f, 1f)]
    public float coinSpawnChance = 0.3f;  // 계단 하나당 코인 생성 확률 (0~1)

    public GameObject obstaclePrefab;     // 장애물 프리팹
    [Range(0f, 1f)]
    public float obstacleSpawnChance = 0.2f;  // 계단 하나당 장애물 생성 확률 (0~1)

    void Start()
{
    if (stairPrefab == null || player == null)
    {
        Debug.LogError("InfiniteStairManager: stairPrefab 또는 player가 비어 있습니다!");
        enabled = false;
        return;
    }

    // 프리팹의 Collider 기준으로 자동 간격 맞추기
    var col = stairPrefab.GetComponent<Collider>();
    if (col != null)
    {
        var size = col.bounds.size;   // 실제 월드 크기
        stepHeight = size.y;          // 계단 높이
        stepDepth = size.z;           // 계단 깊이 (앞뒤)
        // 필요하면 약간 여유를 주고 싶으면 stepDepth = size.z * 1.05f; 이런 식으로
    }

    nextSpawnPos = transform.position;

    for (int i = 0; i < initialStairCount; i++)
    {
        SpawnOneStep();
    }
```

```
    void Update()
    {
        // 1) 플레이어 앞쪽에 계단이 충분한지 확인하면서 계속 생성
        while (nextSpawnPos.z - player.position.z < spawnDistanceAhead)
        {
            SpawnOneStep();
        }

        // 2) 뒤에 너무 멀어진 계단은 삭제
        if (stairs.Count > 0)
        {
            GameObject firstStep = stairs.Peek();
            if (player.position.z - firstStep.transform.position.z > deleteDistanceBehind)
            {
                Destroy(firstStep);
                stairs.Dequeue();
            }
        }
    }

    void SpawnOneStep()
{
    GameObject step = Instantiate(stairPrefab, nextSpawnPos, Quaternion.identity);
    stairs.Enqueue(step);

    // 계단 위에 코인 생성 (확률적으로)
    if (coinPrefab != null && Random.value < coinSpawnChance)
    {
        // 계단 위 살짝 위로 띄운 위치 계산
        Vector3 coinPos = nextSpawnPos;
        coinPos.y += 1.0f; // 계단 윗면 위로 약간 올리기 (필요에 따라 조절)

        Instantiate(coinPrefab, coinPos, Quaternion.identity);
    }

    // 장애물 생성 (확률)
    if (obstaclePrefab != null && Random.value < obstacleSpawnChance)
    {
        Vector3 obstaclePos = nextSpawnPos;
        // 계단 윗면 가운데에 놓되, 플레이어가 지나가는 높이 정도로 올리기
        obstaclePos.y += 0.5f; // 계단 높이에 맞게 필요하면 조정

        Instantiate(obstaclePrefab, obstaclePos, Quaternion.identity);
    }

    // 다음 계단 위치
    nextSpawnPos.y += stepHeight;
    nextSpawnPos.z += stepDepth;
}

}
```

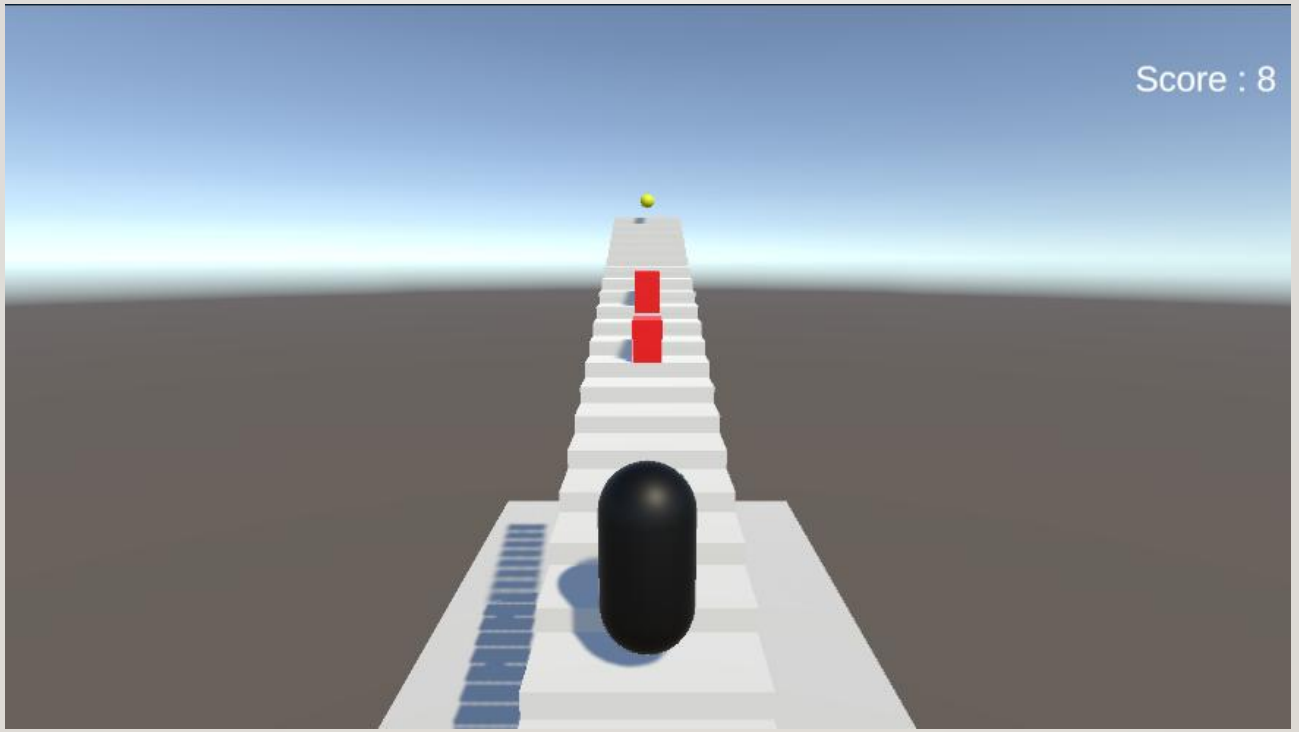
데드존



자동으로 코인 및 방해물 생성

```
{
    GameObject step = Instantiate(stairPrefab, nextSpawnPos, Quaternion.identity);
    stairs.Enqueue(step);

    // 계단 위에 코인 생성 (확률적으로)
    if (coinPrefab != null && Random.value < coinSpawnChance)
    {
        // 계단 위 살짝 위로 띄운 위치 계산
        Vector3 coinPos = nextSpawnPos;
        coinPos.y += 1.0f; // 계단 윗면 위로 약간 올리기 (필요에 따라 조절)

        Instantiate(coinPrefab, coinPos, Quaternion.identity);
    }

    // 장애물 생성 (확률)
    if (obstaclePrefab != null && Random.value < obstacleSpawnChance)
    {
        Vector3 obstaclePos = nextSpawnPos;
        // 계단 윗면 가운데에 놓되, 플레이어가 지나가는 높이 정도로 올리기
        obstaclePos.y += 0.5f; // 계단 높이에 맞게 필요하면 조정

        Instantiate(obstaclePrefab, obstaclePos, Quaternion.identity);
    }

    // 다음 계단 위치
    nextSpawnPos.y += stepHeight;
    nextSpawnPos.z += stepDepth;
}
```

코인 획득 시 스코어 증가



스코어 증가 시 속도 증가

```
using UnityEngine;

public class PlayerMove : MonoBehaviour
{
    [Header("기본 속도 설정")]
    public float baseForwardSpeed = 5f;   // 기본 앞으로 속도
    public float sideSpeed = 4f;          // 좌우 이동 속도

    [Header("난이도(속도 증가) 설정")]
    public float speedPerScore = 0.1f;    // 점수 1점당 속도 증가량
    public float maxForwardSpeed = 15f;   // 최대 앞으로 속도

    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        Vector3 velocity = rb.velocity;

        // 좌우 입력
        float h = Input.GetAxisRaw("Horizontal");
        velocity.x = h * sideSpeed;

        // 점수에 따라 앞으로 속도 계산
        float targetSpeed = baseForwardSpeed;

        // ScoreManager가 존재하면 점수 반영
        if (ScoreManager.Instance != null)
        {
            int score = ScoreManager.Instance.score;
            targetSpeed += score * speedPerScore;
        }

        // 최대 속도 제한
        if (targetSpeed > maxForwardSpeed)
        {
            targetSpeed = maxForwardSpeed;
        }

        velocity.z = targetSpeed;   // 앞으로 이동 속도 적용

        rb.velocity = velocity;
    }
}
```

BGM

출처

- GPT
- 제미나이

점수 : 10점

- 초기버전의 게임에서 여러가지 기능을 추가하였고 게임이 제대로 동작하므로 10점이라고 생각합니다.