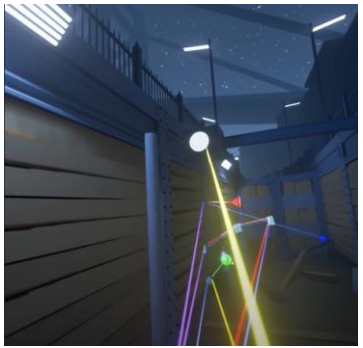
Light Repair Team

LRT

기획(레퍼런스)



[Trailer 영상]

작업할 내용

laser

목표물

거울

분배기

합치기

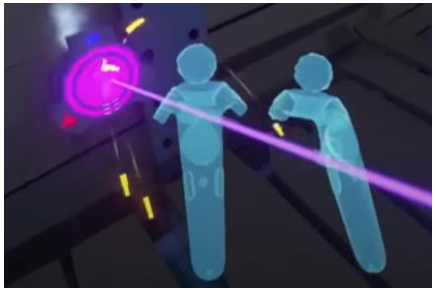
플레이어

맵

작업할 내용(Player) - 김성호

Output

Input
- 닿았을 시 이펙트



PlayerControl
- 카메라회전
- 모션 작용
- 플레이어 움직임

작업할 내용(Player) - 김성호

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

// Player를 움직이고 싶다
Unity 스크립트|참조 0개
public class PlayerMove : MonoBehaviour
{

    public float speed = 5;

    Unity 메시지|참조 0개
    void Start()
    {

        Cursor.visible = false; //커서를 화면에서 안보이게
        Cursor.lockState = CursorLockMode.Locked; //커서를 마우스 화면 중앙에 고정
    }

    Unity 메시지|참조 0개
    void Update()
    {

        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");

        Vector3 dir = new Vector3(h, 0, v);
        dir = Camera.main.transform.TransformDirection(dir);
        dir.Normalize();
        dir.y = 0;

        transform.position += dir * speed * Time.deltaTime;
    }
}
```

1) Player Move + Cam 오브젝트

FPS 기반의 이동 툴과 1인칭 시점 Cam 회전을 이용하여 움직이는 기능 구현

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

// Player의 시점에서 Cam을 회전시키고 싶다
Unity 스크립트|참조 0개
public class CamRotate : MonoBehaviour
{
    // rx, ry 축값
    float rx, ry;
    // 회전속도
    public float rotspeed = 200;

    Unity 메시지|참조 0개
    void Start()
    {

    }

    Unity 메시지|참조 0개
    void Update()
    {
        float mx = Input.GetAxis("Mouse X");
        float my = Input.GetAxis("Mouse Y");

        rx += my * rotspeed * Time.deltaTime;
        ry += mx * rotspeed * Time.deltaTime;

        rx = Mathf.Clamp(rx, -70, 70);
        transform.eulerAngles = new Vector3(-rx, ry, 0);
    }
}
```

작업할 내용(Player) - 김성호

```
void Update()
{
    // 만약 왼쪽 컨트롤러의 telefort 버튼을 누르면
    if (teleport.GetStateDown(SteamVR_Input_Sources.LeftHand))
    {
        // 조준점이 보이고
        maker.SetActive(true);
        // lr를 켜고싶다
        lr.enabled = true;

    }

    // Ray를 이용해서 왼쪽 컨트롤러의 앞방향으로 바라보고싶다
    Ray ray = new Ray(hand.position, hand.forward);
    lr.SetPosition(0, ray.origin);
    RaycastHit hitInfo;
    bool isRayCast = Physics.Raycast(ray, out hitInfo);
    if (isRayCast)
    {
        lr.SetPosition(1, hitInfo.point);
        maker.transform.position = hitInfo.point + hitInfo.normal * 0.1f;
        maker.transform.localScale = makerOriginScale * kAdjust * hitInfo.distance;
        maker.transform.forward = hitInfo.normal;
    }
    else
    {
        // 허공
        Vector3 pos = ray.origin + ray.direction * 100;
        lr.SetPosition(1, pos);
        maker.transform.position = pos;
        maker.transform.localScale = makerOriginScale * kAdjust * 100;
        maker.transform.forward = ray.origin;
    }

    // 그렇지 않고 왼쪽 컨트롤러의 teleport 버튼을 떼면
    if (teleport.GetStateUp(SteamVR_Input_Sources.LeftHand))
    {
        // 조준점을 안보이게 하고싶다
        maker.SetActive(false);
        // lr를 끄고싶다
        lr.enabled = false;

        if (isRayCast)
        {
            // 이때 Ray로 바라본곳에 Floor가 있다면
            int hitlayer = hitInfo.transform.gameObject.layer;
            if (hitlayer == LayerMask.NameToLayer("Floor"))
            {
                // 그곳으로 이동하고 싶다
                transform.position = hitInfo.point;
                // tower 같은 곳으로 이동할때 사용함
                //transform.position = hitInfo.transform.position;
            }
```

2) Teleport 오브젝트

Raycast 기능을 이용하여 컨트롤러 앞방향으로 바라봄

1. 버튼을 누르면 Teleport로 이동할 시점을 보이게 하여 원하는 시점으로 이동
2. 버튼을 떼면 Teleport 시점이 보이지 않는 기능을 구형

작업할 내용(Player) - 김성호

```
참조 1개
internal void 놔줘()
{
    grabObject = null;
}

참조 1개
private void Throw()
{
    if (grabObject != null)
    {
        //grabObject.놓다(pose);
        //grabObject = null;
        grabObject.transform.parent = null;
        grabObject = null;
    }
}

참조 1개
private void Catch()
{
    Collider[] cols = Physics.OverlapSphere(transform.position, 100f, LayerMask.GetMask("Item"));
    if (cols.Length > 0)
    {
        for (int i = 0; i < cols.Length; i++)
        {
            grabObject = cols[i].gameObject;
            if (grabObject != null)
            {
                //grabObject.잡다(transform.position, transform);
                // 만약 다른손이 잡고있던 물체였다면 다른손에게 "놔줘" 라고 요청하고싶다
                if (grabObject.transform.parent != null)
                {
                    grabObject.transform.parent = null;
                }
                grabObject.transform.position = transform.position;
                grabObject.transform.localPosition += new Vector3(0, 0, 0.3f);
                grabObject.transform.parent = gameObject.transform;
                break;
            }
        }
    }
}
```

3) Grip, Grabbable 오브젝트

Teleport와 같은 Raycast 기능을 이용하였고
Grabbable 스크립트와 연동시켜 작동됨

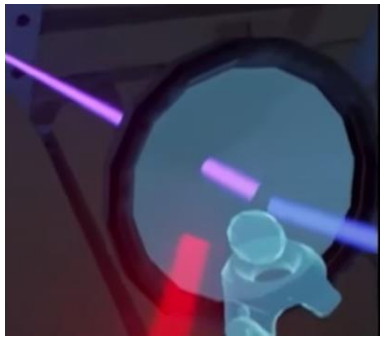
1.아이템이 있는 쪽으로 컨트롤러를 가까이 가져가면 아이템을 집을수 있고
2. 버튼을 떼면 놓은 상태 그대로 아이템이 놓여져 있는 기능을 구현하였음

```
참조 0개
internal void 잡다(Vector3 pos, Transform parent)
{
    // 만약 다른손이 잡고있던 물체였다면 다른손에게 "놔줘" 라고 요청하고싶다
    if (transform.parent != null)
    {
        transform.parent.GetComponent<Grip>().놔줘();
        transform.parent = null;
    }
    transform.position = pos;
    transform.parent = parent;
    GetComponent<Rigidbody>().isKinematic = true;

}

참조 0개
internal void 놓다(Valve.VR.SteamVR_Behaviour_Pose pose)
{
    transform.parent = null;
}
```

작업할 내용(Item) - 송민우

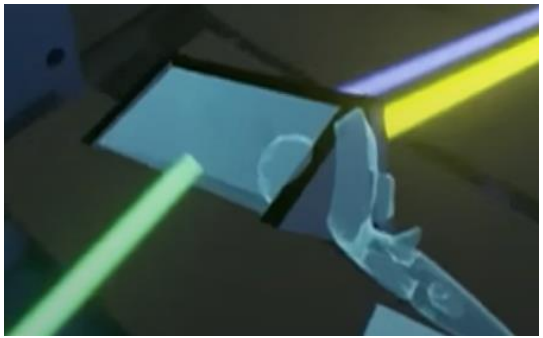
반사경

- 빛 반사구현



합성기

- 여러 빛 입력 구현
- 빛 색깔 조합
- 빛 출력



분배기

- 빛 색깔 분해
- 여러 빛 출력

작업한 내용(Item) - 송민우

```
public class LaserHit {
    3 references
    public RaycastHit raycastHit;
    5 references
    public LaserInput hitLaserInput;
    4 references
    public float width;
    8 references
    public LaserColor color;
    2 references
    public Vector3 inputDir;
    1 reference
    public LaserHit(RaycastHit hit, LaserInput hitLaserInput, Vector3 inputDir, LaserColor color, float width) {
        this.raycastHit = hit;
        this.hitLaserInput = hitLaserInput;
        this.inputDir = inputDir;
        this.color = color;
        this.width = width;
    }
}
```

유니티 **Ray를 모방**하여
Laser, LaserHit, LaserInput 구현.

```
public interface LaserInput {
    1 reference
    void OnLaserInput(LaserHit hit);
    3 references
    void OnLaserInputEnd(LaserHit hit);
}
```

```
public class Laser {
    2 references
    public static void Shoot(LineRenderer lineRenderer, Vector3 startPos, Vector3 dir, float width, LaserColor color, ref LaserHit prevLaserHit) ···
    1 reference
    public static Color GetColor(LaserColor lc) {···
}
```

작업한 내용(Item) - 송민우

```
public class Mirror : MonoBehaviour, LaserInput
{
    5 references
    private LineRenderer lr;
    5 references
    private LaserHit prevLaserHit;
    0 references
    void Start()
    {
        lr = GetComponent<LineRenderer>();
        lr.material = MaterialManager.Instance.laserMaterial;
    }

    1 reference
    public void OnLaserInput(LaserHit hit) {
        lr.enabled = true;
        Vector3 dir = Vector3.Reflect(hit.inputDir, hit.raycastHit.normal);
        Laser.Shoot(lr, hit.raycastHit.point, dir, hit.width, hit.color, ref prevLaserHit);
    }

    3 references
    public void OnLaserInputEnd(LaserHit hit) {
        lr.enabled = false;
        if (prevLaserHit != null) {
            prevLaserHit.hitLaserInput.OnLaserInputEnd(prevLaserHit);
            prevLaserHit = null;
        }
    }
}
```

거울 오브젝트

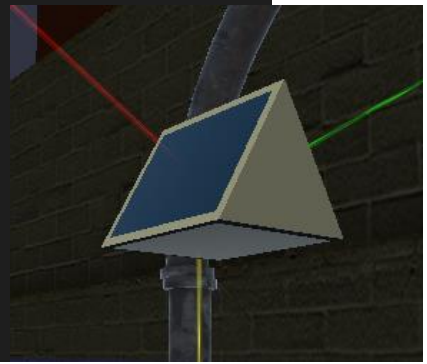
LaserInput을 상속

빛이 들어올 때는
들어온 빛의 반사벡터로 Laser.Shoot()

빛이 들어오지 않을 때는
LineLenderer를 끔

작업한 내용(Item) - 송민우

```
public void OnLaserInput(LaserHit hit)
{
    switch (hit.color)
    {
        case LaserColor.YELLOW:
            DividerOn(hit, LaserColor.RED, LaserColor.GREEN);
            break;
        case LaserColor.PURPLE:
            DividerOn(hit, LaserColor.RED, LaserColor.BLUE);
            break;
        case LaserColor.CYAN:
            DividerOn(hit, LaserColor.BLUE, LaserColor.GREEN);
            break;
        default:
            DivderOff();
            break;
    }
}

2 references
public void OnLaserInputEnd(LaserHit hit)
{
    DivderOff();
}
```



빛 분배기 오브젝트

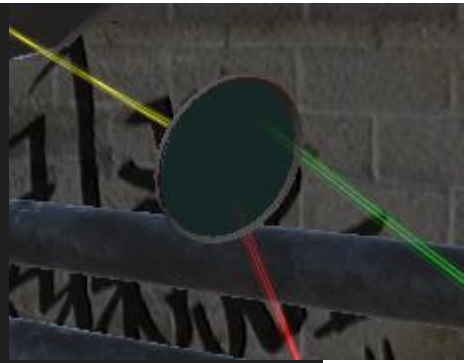
- **노랑, 보라, 하늘색 빛을 각각의 원색 2개로 분리시킴**
- **그 외의 색깔이 들어오면 꺼짐**

구현한 방법

- **LaserInput을 상속**
- **빛이 들어올 때는 들어온 빛의 분해 가능한 색이면 분리, 아니면 끔**
- **빛이 들어오지 않을 때는 분배기를 끔**

작업한 내용(Item) - 송민우

```
public enum LaserColor {
    /*
        NONE    = 000
        RED     = 001
        GREEN   = 010
        BLUE    = 100
        YELLOW  = (R + G)     = 011
        PURPLE  = (R + B)     = 101
        CYAN    = (G + B)     = 110
        WHITE   = (R + G + B) = 111
    */
    0 references
    NONE = 0,
    7 references
    RED = 1 << 0,
    6 references
    GREEN = 1 << 1,
    6 references
    BLUE = 1 << 2,
    2 references
    YELLOW = LaserColor.RED + LaserColor.GREEN,
    2 references
    PURPLE = LaserColor.RED + LaserColor.BLUE,
    2 references
    CYAN = LaserColor.GREEN + LaserColor.BLUE,
    1 reference
    WHITE = LaserColor.RED + LaserColor.GREEN + LaserColor.BLUE
}
```



빛 합성기 오브젝트

- **3원색 2개를 받아 조합색 1개로 합침**
- **3원색 외의 색은 그대로 내보냄**
- **빛이 없으면 꺼짐**

구현한 방법

- **빛 합성을 쉽게 계산하기 위해 LaserColor를 비트마스크로 정의**

작업한 내용(Item) - 송민우

```
void Update()
{
    if (colorMask != 0) {
        output.gameObject.SetActive(true);
        output.laserColor = (LaserColor)colorMask;
    } else {
        output.gameObject.SetActive(false);
    }
}

1 reference
public void OnLaserInput(LaserHit hit) {
    colorMask |= (int)hit.color;
}

3 references
public void OnLaserInputEnd(LaserHit hit) {
    colorMask ^= (int)hit.color;
}
```

빛 합성기 오브젝트

- 3원색 2개를 받아 조합색 1개로 합침
- 3원색 외의 색은 그대로 내보냄
- 빛이 없으면 꺼짐

구현한 방법

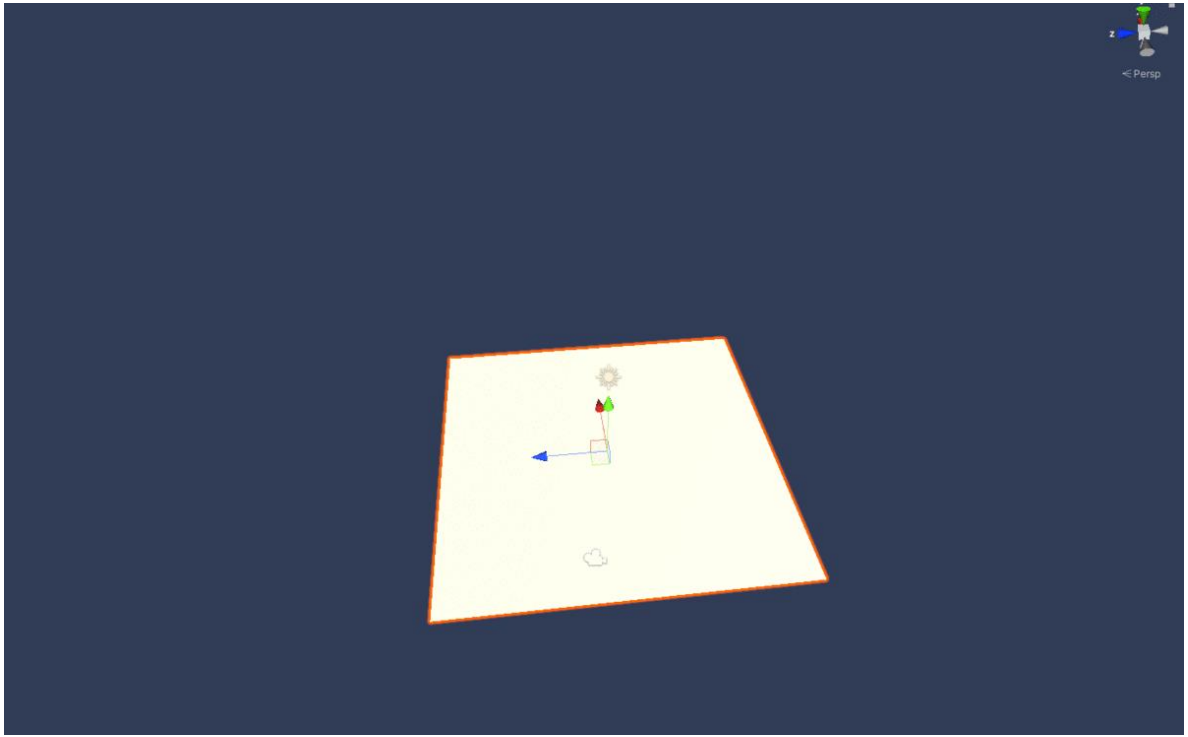
- 빛이 들어오고 있으면 colorMask에 OR 연산
- 빛이 나가면 colorMask에 XOR 연산
- 연산한 값이 그대로 LaserColor가 됨

작업할 내용(Level Design) - 이승재, 하미연

아이템 이미지 구현
- 분배기
- 반사기
- 합성기
- 레이져 Input
- 레이져 Output

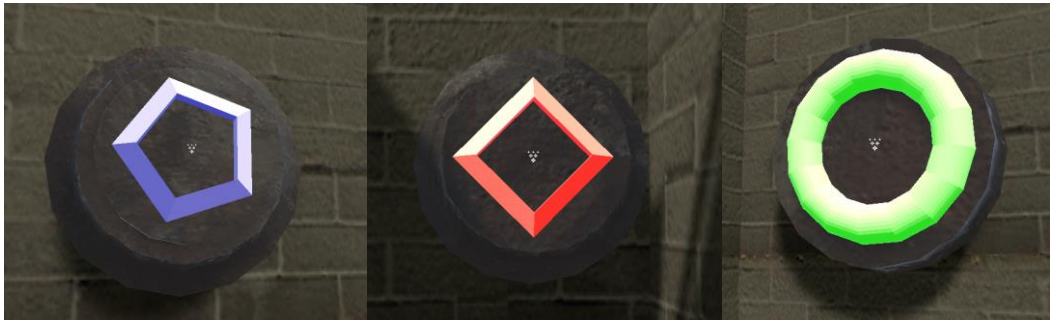
맵 구현
- 스테이지 제작
- 맵 제작
- 스테이지 완료시 건물 불들어오기
- 메인 화면
- 종료 화면
- UI

맵 제작- 이승재, 하미연



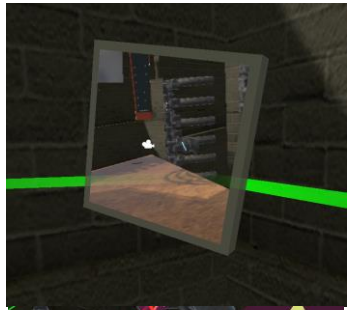
게임 오브젝트 제작- 이승재, 하미연

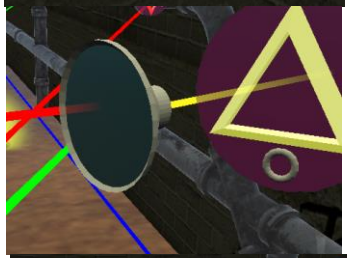
Lasor 발사기

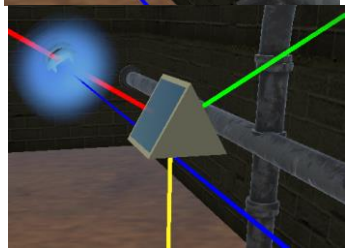


Lasor 도착지

거울 Item



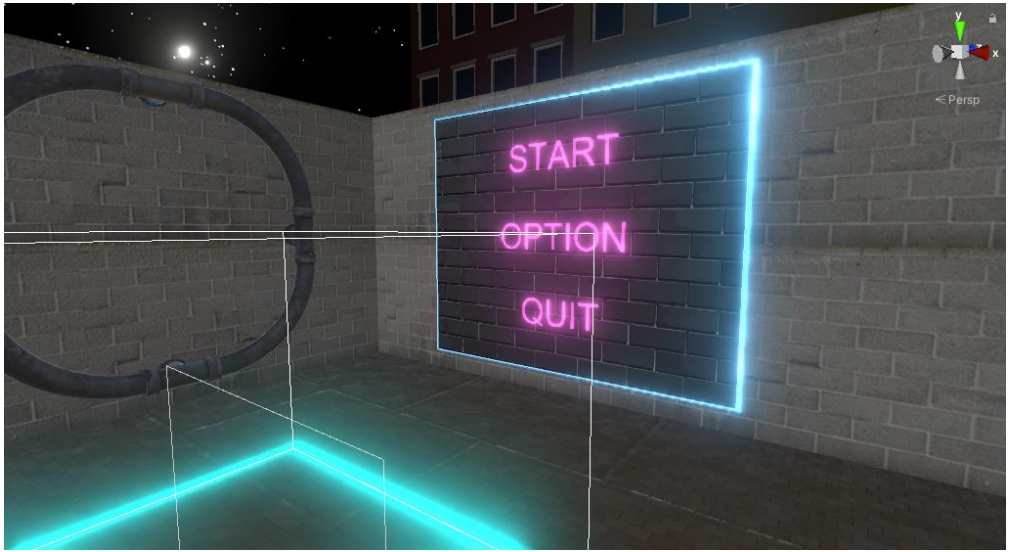




벽 구현- 이승재, 하미연

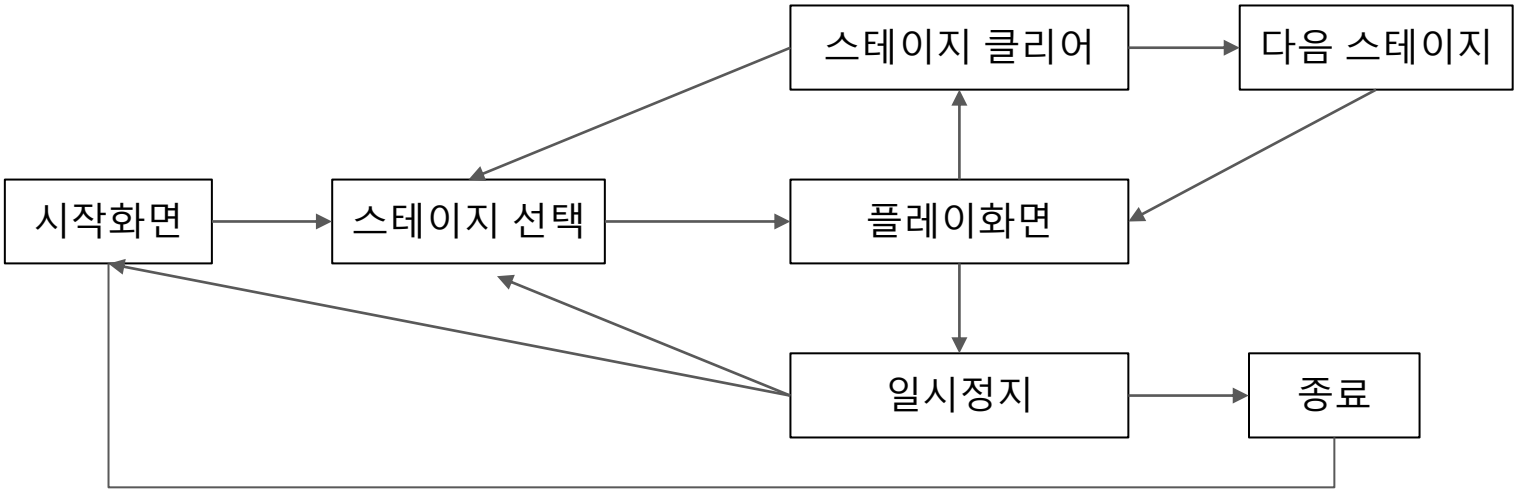
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class Wall : MonoBehaviour, StageEnv {
    public float downSpeed = 0.1f;
    public GameObject dustParticle;
    public int finishCount=50;
    int count;


    private void Start() {
        dustParticle.SetActive(false);
    }

    public void OnStageClear() {
        dustParticle.SetActive(true);

        Invoke("DownWall", 0.1f);



    }


    void DownWall() {
        if (count == finishCount) {
            CancelInvoke("DownWall");
            gameObject.SetActive(false);
            dustParticle.SetActive(false);
            count = 0;
        }
        gameObject.transform.position += new Vector3(0, -downSpeed, 0);

        count++;
        Invoke("DownWall", 0.1f);


    }


}
```

시작페이지 제작 - 이승재, 하미연




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

0 Unity 스크립트 | 참조 0개
public class SceneChanger : MonoBehaviour
{
    public Image bgBorder;
    참조 0개
    public void PlayGame()
    {
        FadeOutBorder();
        Fader.instance.FadeOut(0.7f, () => {
            Invoke("NextScene", 0.5f);
        });
    }

    참조 1개
    private void FadeOutBorder()
    {
        StartCoroutine(IeFadeOutBorder());
    }

    참조 1개
    private IEnumerator IeFadeOutBorder()
    {
        float t = 0;
        while (t < 1)
        {
            t = Time.deltaTime;

            Material mat = bgBorder.material;
            Material fadeOutMat = new Material(Shader.Find("Standard"));
            fadeOutMat.SetColor("_EmissionColor", mat.color * (1 - t) / 1.3f);
            bgBorder.material = fadeOutMat;

            yield return new WaitForEndOfFrame();
        }
    }

    참조 0개
    public void QuitGame()
    {
        Application.Quit();
    }



    참조 0개
    void NextScene()
    {
        SceneManager.LoadScene("Background");
    }
}
```

화면 플로우



GamePlay 영상

땡큐