PORTFOLIO

이재혁

E-mail : leehuk0622@gmail.com

Git : https://github.com/roomMaker

목차

1.연애는 처음이라서...

장르

AR 연애 시뮬레이션

개발기간

2022.09.29 ~ 2022.12.09

개발인원

8인 / 개발 4인 기획 4인

개발환경

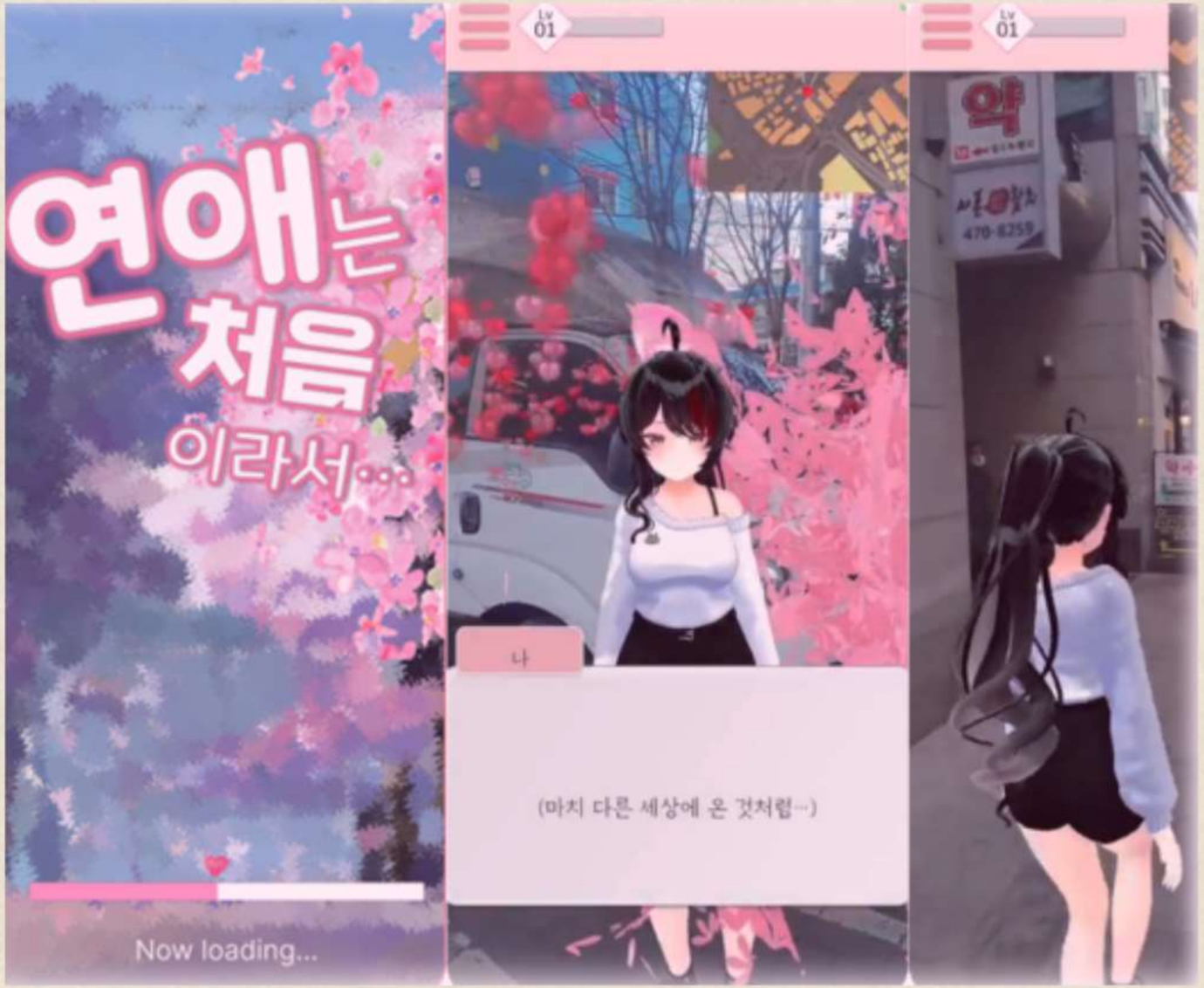
- Unity
- C#
- Google Maps API
- ARCore
- Git

담당업무

- Google Maps API 적용
- 스토리모드 이벤트 / 미니맵
- VPS 이벤트

링크

https://youtu.be/kLbO3MtDinI

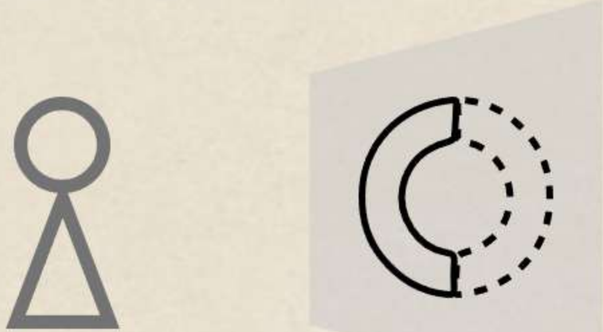


- 건물에 AR 오브젝트가 가려지는 효과

보이지 않는 가상의 벽에 큐브가 일부 가려지는 모습

벽을 뚫고 나오는 랍스터와 건물 안으로 사라지는 고래와 같은 연출에 사용

건물에 AR 오브젝트가 가려지는 효과를 내기 위해 Google Maps API 기술을 이용하여 플레이어의 휴대전화의 GPS 좌표를 기반으로 3D 형식의 맵을 로드한 뒤 맵의 건물오브젝트에 투명 머터리얼을 적용시켰습니다.



투명 벽

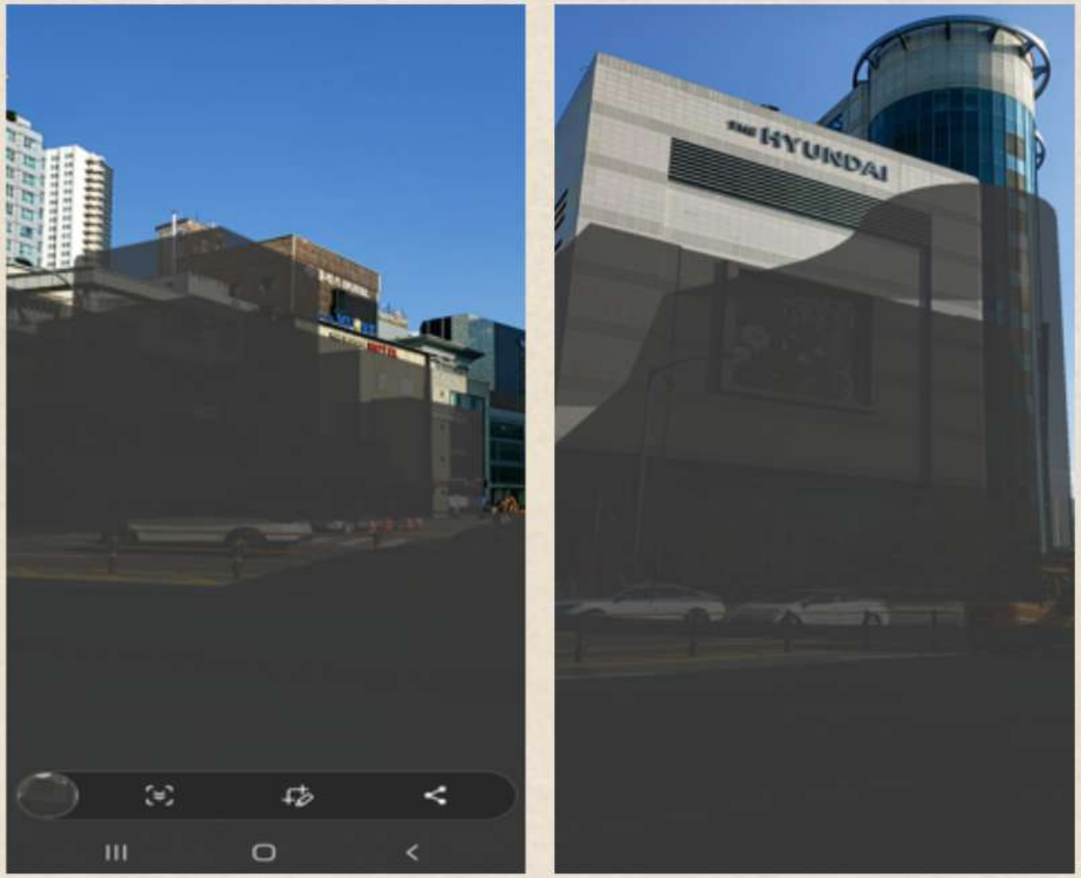
- 건물에 AR 오브젝트가 가려지는 효과

```
// 건물에 메쉬, 콜라이더, 네비판단태그를 넣는다.
public void SetBuildingsOption()
{
    _mesh = GetComponentsInChildren<MeshRenderer>();

    foreach (var renderer in _mesh)
    {
        _miniMapBuilding = renderer.gameObject;
        if (_miniMapBuilding.name[0] == 'E')
        {
            _miniMapBuilding.tag = "Building";
        }

        renderer.materials = _mat;
        renderer.gameObject.AddComponent<MeshCollider>();
        renderer.gameObject.AddComponent<NavMeshSourceTag>();
    }
}
```

Google Maps API를 통해 로드된 맵의 건물 오브젝트의 이름이 모두 E로 시작하는것을 바탕으로 맵이 로드된 후 로드된 맵 오브젝트들을 순환하며 건물 오브젝트에 메쉬, 콜라이더, 태그 등 필요한 요소들을 할당

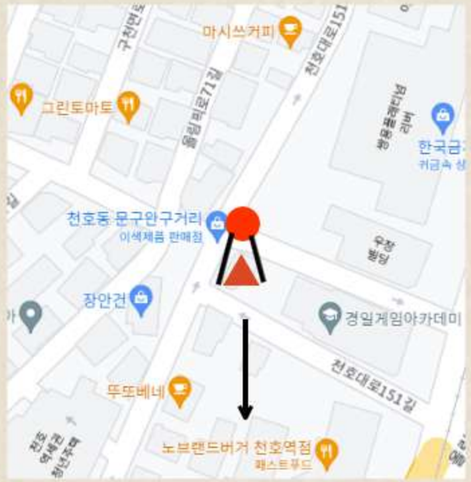
- Google Maps API 건물위치과 실제 건물위치의 동기화

Google Maps API로 생성된 3D 맵 오브젝트를 기반으로 실행되는
컨텐츠들을 위해 현실 건물의 위치와 맵 오브젝트의 건물 위치를 맞추어야 했습니다.
미리 지정된 실제 건물을 플레이어가 바라보며 버튼을 누르면
플레이어의 위치를 중심으로 한 뒤, 맵 오브젝트상의 지정된 건물 오브젝트의 방향과
플레이어가 실제 바라보고 있는 방향의 각도 차이를 이용하여
RotateAround() 함수로 맵 오브젝트 전체를 돌리는 방법으로 구현하였습니다.



Google Maps API로 로드 된 반투명한 회색 건물과 실제 건물의 위치가 일치하는 모습

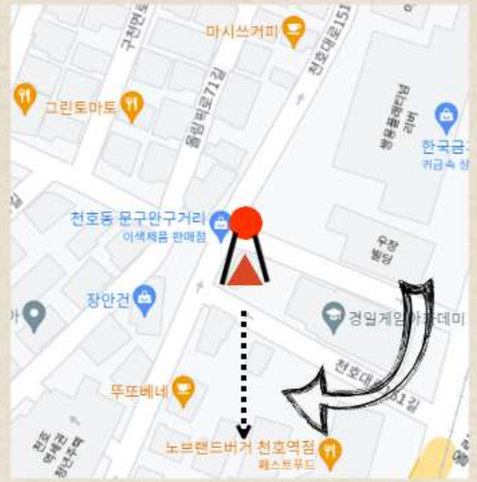
▲: 미리 지정된 건물



실제 지도

로드된 맵

플레이어가 실제 보고있는 방향인
점선 화살표(↓)와 플레이어로부터
지정된 건물로 향하는 방향(⋯▸)의
각도 차이 만큼 로드된 맵을 돌린다.
위의 경우 약 90도 차이가 나므로
로드된 맵을 시계방향으로 90도
돌려서 위치를 맞춘다.



로드된 맵을 시계방향으로 90도 회전시켜
실제 지도와 위치를 맞춘 모습

- Google Maps API 건물위치과 실제 건물위치의 동기화

```
/// <summary>
/// 등록된 건물이름과 스테이지 인덱스로 건물을 특정한 뒤 게임오브젝트에 저장, 반드시 맵이 로드된 후 실행되어야 함 (중요)
/// </summary>
private void SetDefaultBuilding()
{
    // 일반 맵
    foreach (var defaultBuilding in _map[0].GetComponentsInChildren<Transform>())
    {
        if (defaultBuilding.gameObject.name == GameManager.Instance.CurrentStageDefaultBuildingName)
        {
            _defaultBuilding[0] = defaultBuilding.gameObject;
            break;
        }
    }
    // 미니맵
    foreach (var miniMapDefaultBuilding in _map[1].GetComponentsInChildren<Transform>())
    {
        if (miniMapDefaultBuilding.gameObject.name == GameManager.Instance.CurrentStageDefaultBuildingName)
        {
            _defaultBuilding[1] = miniMapDefaultBuilding.gameObject;
            break;
        }
    }
}
```

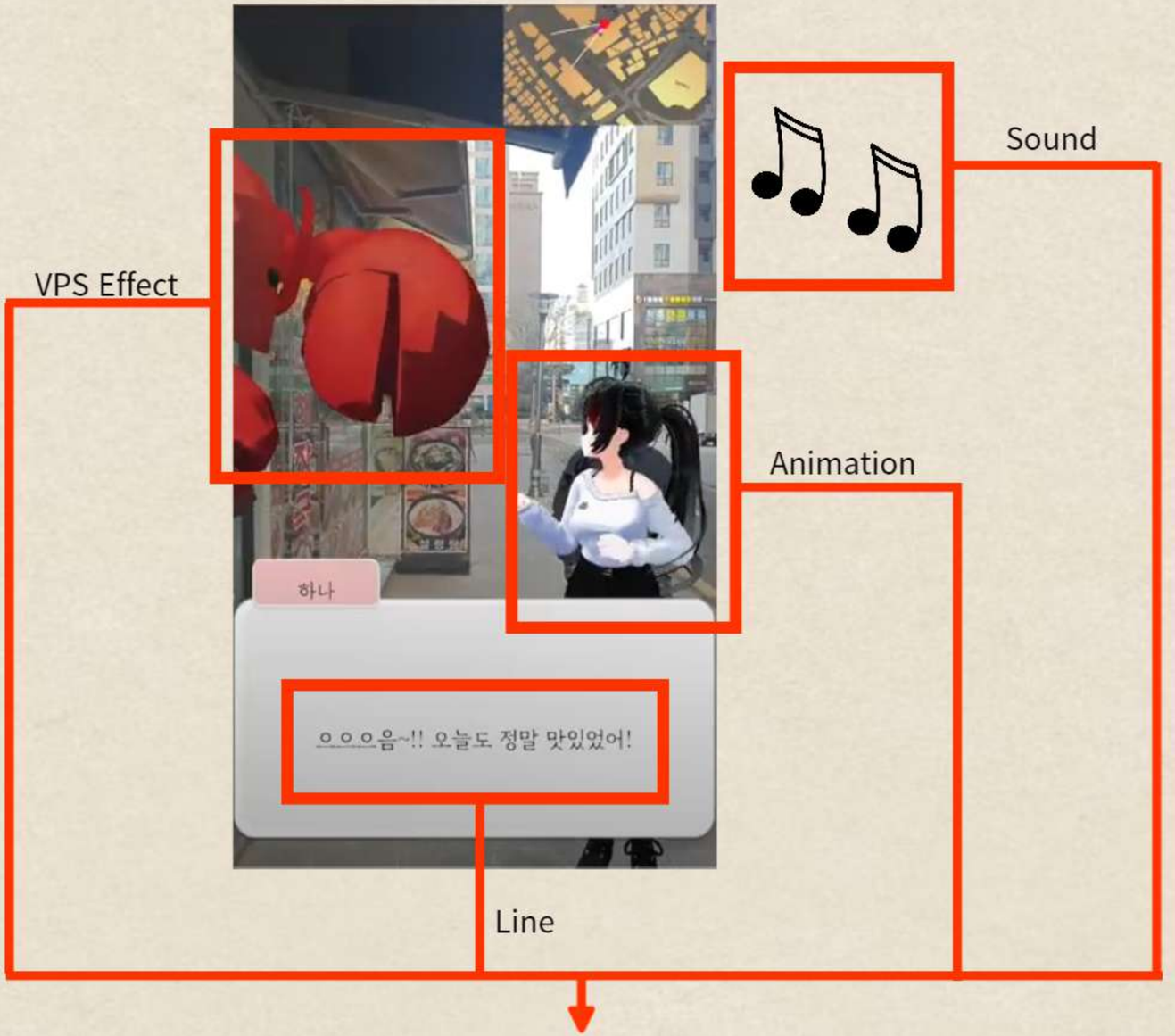
```
/// <summary>
/// 저장된 게임오브젝트 건물을 이용하여 맵을 회전시키는 함수
/// </summary>
public void RotateMap()
{
    SetDefaultBuilding();
    Transform mainTransform = Camera.main.transform;
    Transform tempTransform = _player.transform;


    Vector3 dir = _defaultBuilding[0].transform.position - mainTransform.position;
    dir.y = 0f;
    tempTransform.LookAt(_defaultBuilding[0].transform.position);
    _map[0].transform.RotateAround(
        mainTransform.position,
        Vector3.up,
        mainTransform.rotation.eulerAngles.y - tempTransform.rotation.eulerAngles.y
    );


    for (int i = 1; i < 4; i++)
    {
        _map[i].transform.rotation = _map[0].transform.rotation;
    }
}
```

싱글톤 패턴을 사용한 GameManager에 미리 저장해 놓은 건물의 이름과 로드된 맵의 건물 이름을 비교하여 스테이지별 DefaultBuilding을 세팅합니다.

AR 환경에서의 메인 카메라, 즉 현재 플레이어의 위치와 세팅된 DefaultBuilding의 위치를 이용하여 맵을 회전시킵니다.

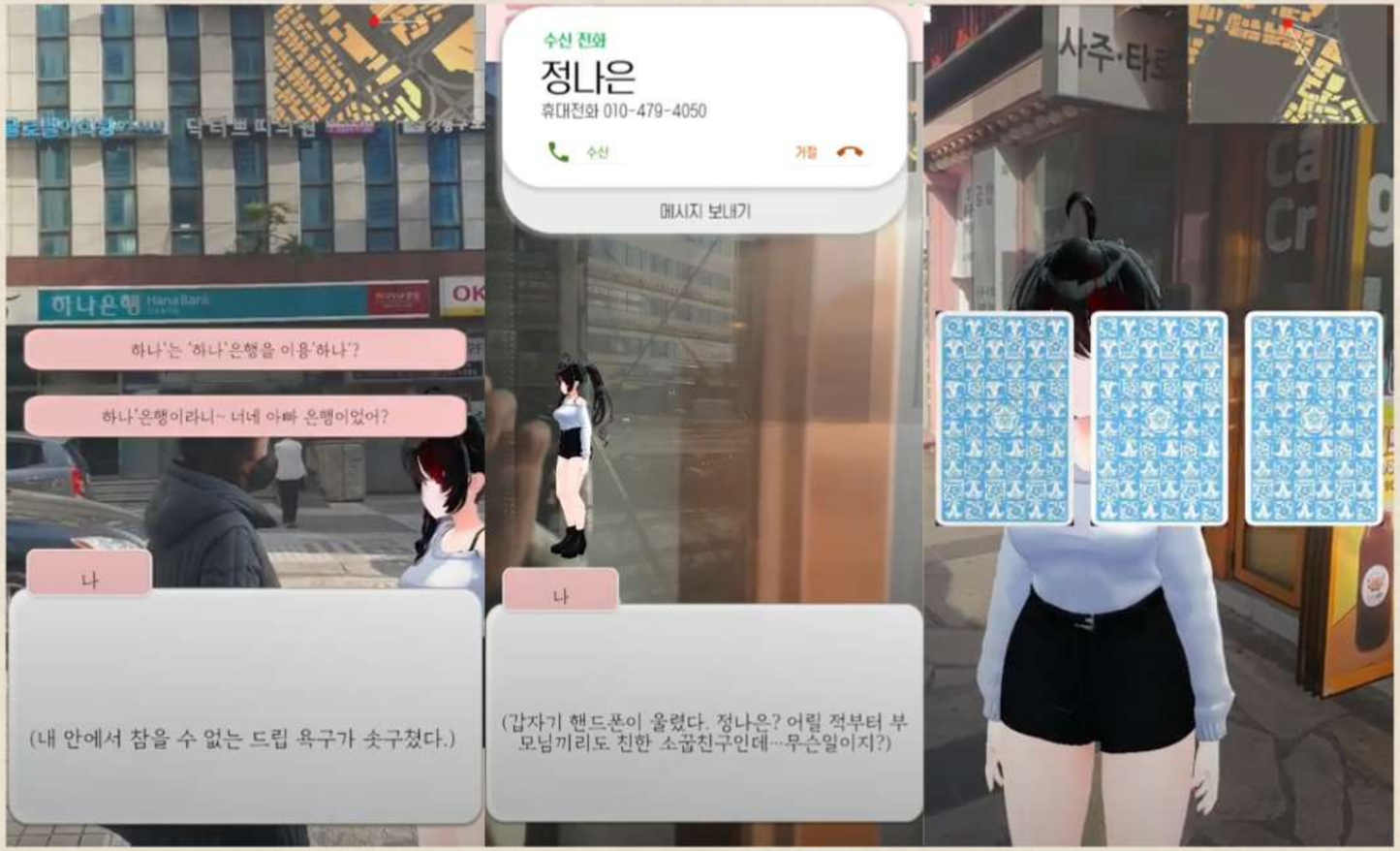
- VPS 이벤트 시스템

특정 구간 진입시 이벤트 발생



특정 구간에 진입했을 때 발생하는 대사 출력,
VPS Effect 활성화, 캐릭터 애니메이션, 사운드 출력과 같은
여러 이벤트를 한 번에 관리 할 수 있는 시스템입니다.
특정 구간에 진입했을 때 원하는 이벤트만 골라 실행할 수 있습니다.

- 그 외 제작 컨텐츠



선택지에 따라 다음 대사들이 변경되는 기능

엔딩 씬

2. 경일

장르

공포

개발기간

2022.08.16 ~ 2022.09.02

개발인원

2인

개발환경

- Unity
- C#
- Git

담당업무

- 상호작용
- 아이템 & 인벤토리
- 퍼즐

링크

https://youtu.be/6lko-AZmC7E

• 상호작용

상호작용

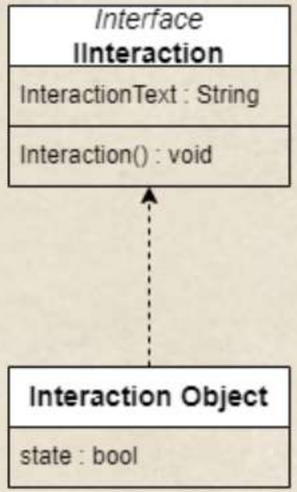
상호작용이 가능한 오브젝트는 IInteraction 인터페이스를 상속받아
플레이어가 바라보았을 때 띄워 줄 상호작용 문구를 InteractionText를 통해,
상호작용시 각각의 오브젝트가 해야하는 행동을 Interaction() 함수를 통해
각각 자신이 가지고 있게 하였습니다.

또한 state 변수를 이용하여
자신의 상태에 맞는 상호작용 문구 및 행동을 가지고 있도록 구현하였습니다.

플레이어가 상호작용 할 경우 각각의 오브젝트는
자신의 현재 상태에서 해야하는 행동을 하게 됩니다.

상태에 따른 상호작용 문구 및 행동 변화

```
public interface IInteraction
{
    string InteractionText;

    void Interaction() { }
}
```



- 상호작용

```
/// <summary>
/// 상호작용시 호출
/// </summary>
public void Interaction()
{
    Move();
}

private void Move()
{
    IsMoveDoor = true;
    if (_isOpen)
    {
        _doorAudio.clip = DoorClose;
        InteractionText = _doorOpenText;
        StartCoroutine(Close());
        _isOpen = false;
    }
    else
    {
        _doorAudio.clip = DoorOpen;
        InteractionText = _doorCloseText;
        _doorAudio.Play();
        StartCoroutine(Open());
        _isOpen = true;
    }
}
```

문 상호작용 스크립트 일부

```
public IEnumerator Open()
{
    while (true)
    {
        transform.Rotate(0, _moveSpeed, 0);
        _boxCollider.enabled = false;
        _elapsedTime += Time.fixedDeltaTime;
        if (_elapsedTime > _doorOpenCooltime)
        {
            _elapsedTime = _initTime;
            IsMoveDoor = false;
            _boxCollider.enabled = true;
            break;
        }
        yield return null;
    }
}

public IEnumerator Close()
{
    while (true)
    {
        transform.Rotate(0, -_moveSpeed, 0);

        _elapsedTime += Time.fixedDeltaTime;
        if (_elapsedTime > _doorOpenCooltime)
        {
            _elapsedTime = _initTime;
            IsMoveDoor = false;
            _doorAudio.Play();
            break;
        }
        yield return null;
    }
}
```

문 상호작용 스크립트 일부

상호작용이 가능한 오브젝트가 각각 가지고 있는 상태 변수로
현재 자신의 상태에 따라 실행해야 하는 동작을 수행합니다.

• 아이템 & 인벤토리

```
public class ItemData : ScriptableObject
{
    public string ItemName;
    public Sprite ItemImage;
}
```

아이템 오브젝트는 ScriptableObject를 사용하여 각각의 아이템 오브젝트를 관리 하였습니다.

아이템과 상호작용 하여 아이템을 획득 할 경우 인벤토리의 왼쪽 상단부터 차례대로 아이템이 채워지게 끔 구현했습니다.

아이템은 사용하면 사라지게 됩니다.

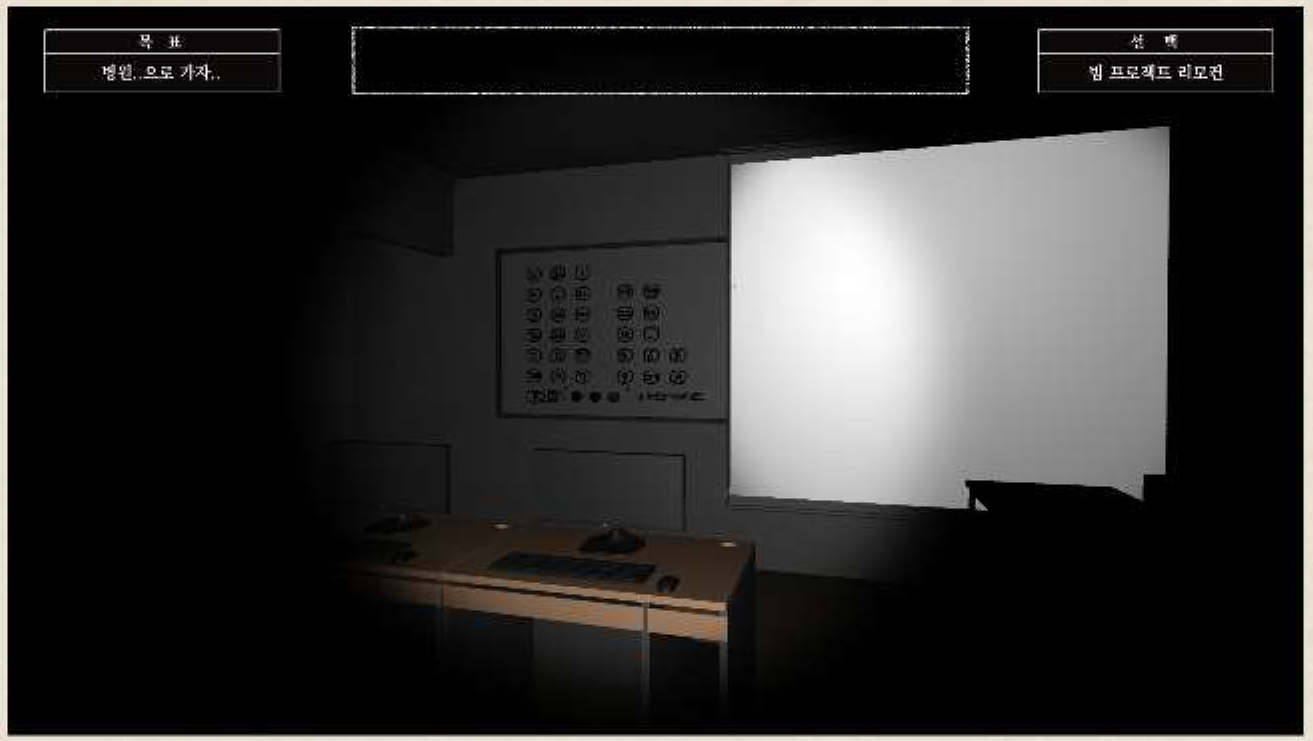
아이템을 사용한 후 인벤토리에서사라진 모습

마우스로 클릭하여 선택

선택하고 나면 현재 선택된 아이템이 갱신

● 그 외 제작 컨텐츠

배열을 이용한 모니터 퍼즐



렌더 텍스쳐를 이용한 퍼즐

특정 조건을 만족하면 발생하는 히든 이벤트

금고 이벤트

3. 스와이프 벽돌깨기 모작

장르

캐주얼

개발기간

2일

개발인원

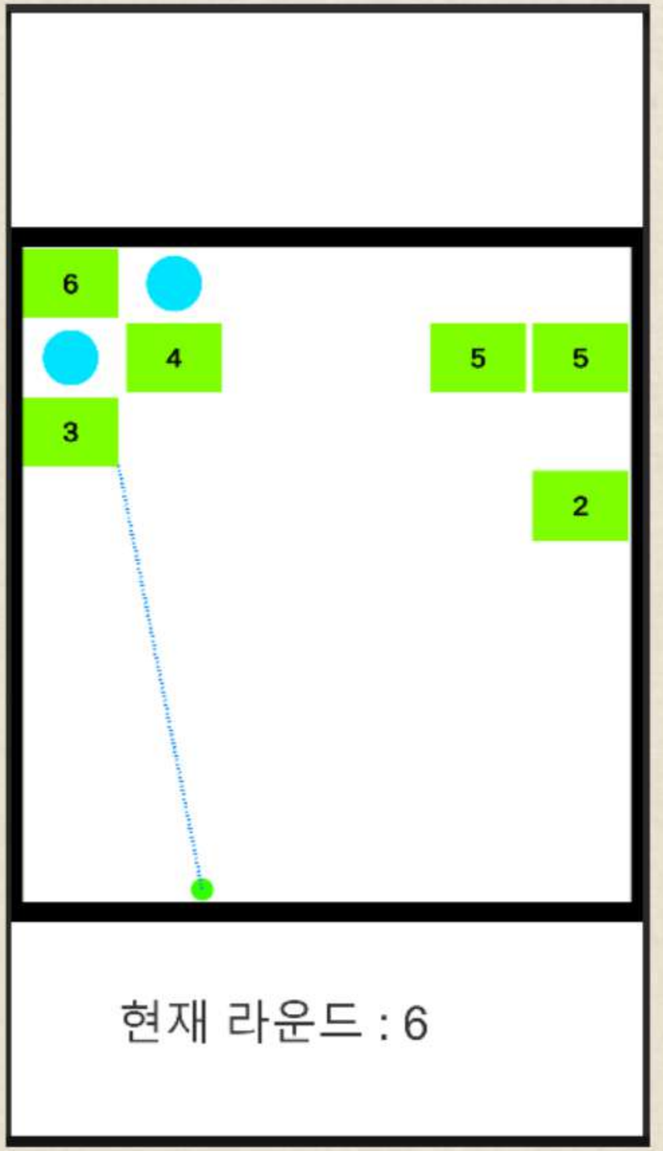
1인

개발환경

- Unity
- C#

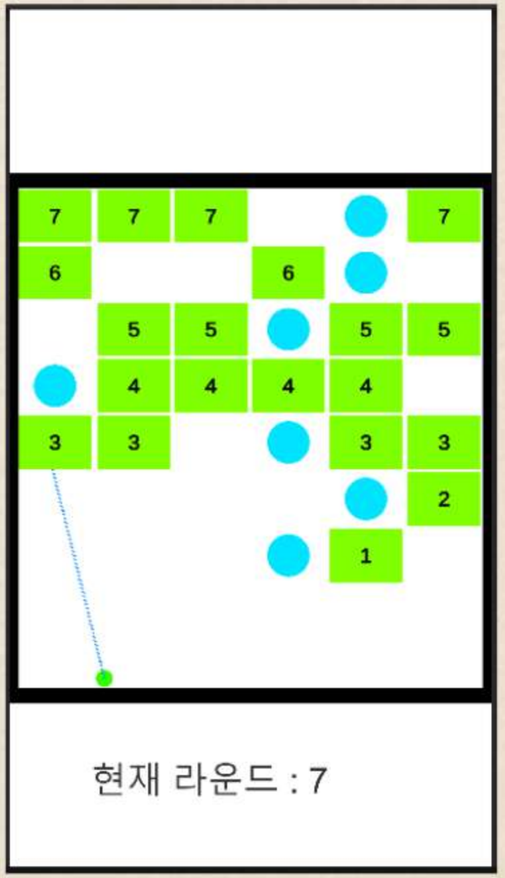
링크

https://github.com/roomMaker/MiniGame_Project



- 블록 및 아이템 랜덤 생성

```
/// <summary>
/// 블록 한 라인 생성
/// </summary>
참조 1개
private void MakeBlockLine()
{
    randItemIndex = Random.Range(0, 5);

    for (int i = 0; i < _blockPosition.Length; ++i)
    {
        if (i == randItemIndex)
        {
            Instantiate(_item, _blockPosition[i].localPosition, Quaternion.identity, Blocks.transform);
            continue;
        }

        randBlockIndex = Random.Range(0, 2);

        if (randBlockIndex != 0)
        {
            Instantiate(_block, _blockPosition[i].localPosition, Quaternion.identity, Blocks.transform);
        }
    }
}
```



아이템이 들어갈 위치의 인덱스를 0~5 사이 랜덤한 값으로 저장한 후 반복문을 돌며 0~2 사이의 랜덤한 값을 생성하면서 값이 0이 아닐 경우에만 블록을 생성하고, 아이템 인덱스일 경우에는 아이템을 생성하는 방식으로 블록과 아이템이 랜덤하게 나오게 구현하였습니다.

감사합니다