

졸업작품 발표

2025.6.11

김다희 2288014
남경서 2288042

목 차

1. 작품 개요

게임 장르
개발 환경
기술 스택

2. 핵심 구현 기능

네트워크 관련 기능
메인 기능
그래픽 관련 기능

3. 소감

졸업 작품을 제작하며

4. 마무리

작품 개요

작품 개요

1

게임 장르

VR 2인 판타지 방탈출

2

개발 환경

유니티
블렌더

3

기술 스택

서버/클라이언트 적용
깃허브를 통한 버전 관리

핵심 구현 기능

핵심 구현 기능

1

네트워크 관련 기능

Relay

Remove

1.2.0 · February 24, 2025

From Unity Registry by Unity Technologies Inc.

`com.unity.services.relay`

[Documentation](#) | [Changelog](#) | [Licenses](#) | [Go to Dashboard](#)

Description

Version History

Dependencies

Samples

Connect the players of your peer-hosted games with Relay. Your games will also benefit from improved reliability, privacy and the ability to communicate across platforms. To get started, enable listen-server UDP networking for your project on the Unity Dashboard.

Features include an allocation service to join game hosts via code, multi-region support, and more.

Netcode for GameObjects

Remove

1.10.0 · July 25, 2024

Release

From Unity Registry by Unity Technologies Inc.

`com.unity.netcode.gameobjects`

[Documentation](#) | [Changelog](#) | [Licenses](#)

Description

Version History

Dependencies

Samples

Netcode for GameObjects is a high-level netcode SDK that provides networking capabilities to GameObject/MonoBehaviour workflows within Unity and sits on top of underlying transport layer.

멀티 플레이는 유니티의 Relay + NGO(Netcode for GameObjects)로 구현


핵심 구현 기능

1

네트워크 관련 기능

```
public static async Task StartRelayServer()
{
    var allocation = await RelayService.Instance.CreateAllocationAsync(2);
    JoinCode = await RelayService.Instance.GetJoinCodeAsync(allocation.AllocationId);
    Debug.Log($"[Relay] 서버 JoinCode: {JoinCode}");

    var relayServerData = new RelayServerData(allocation, "dtls");
    NetworkManager.Singleton.GetComponent<UnityTransport>().SetRelayServerData(relayServerData);
    NetworkManager.Singleton.StartServer();
}
```

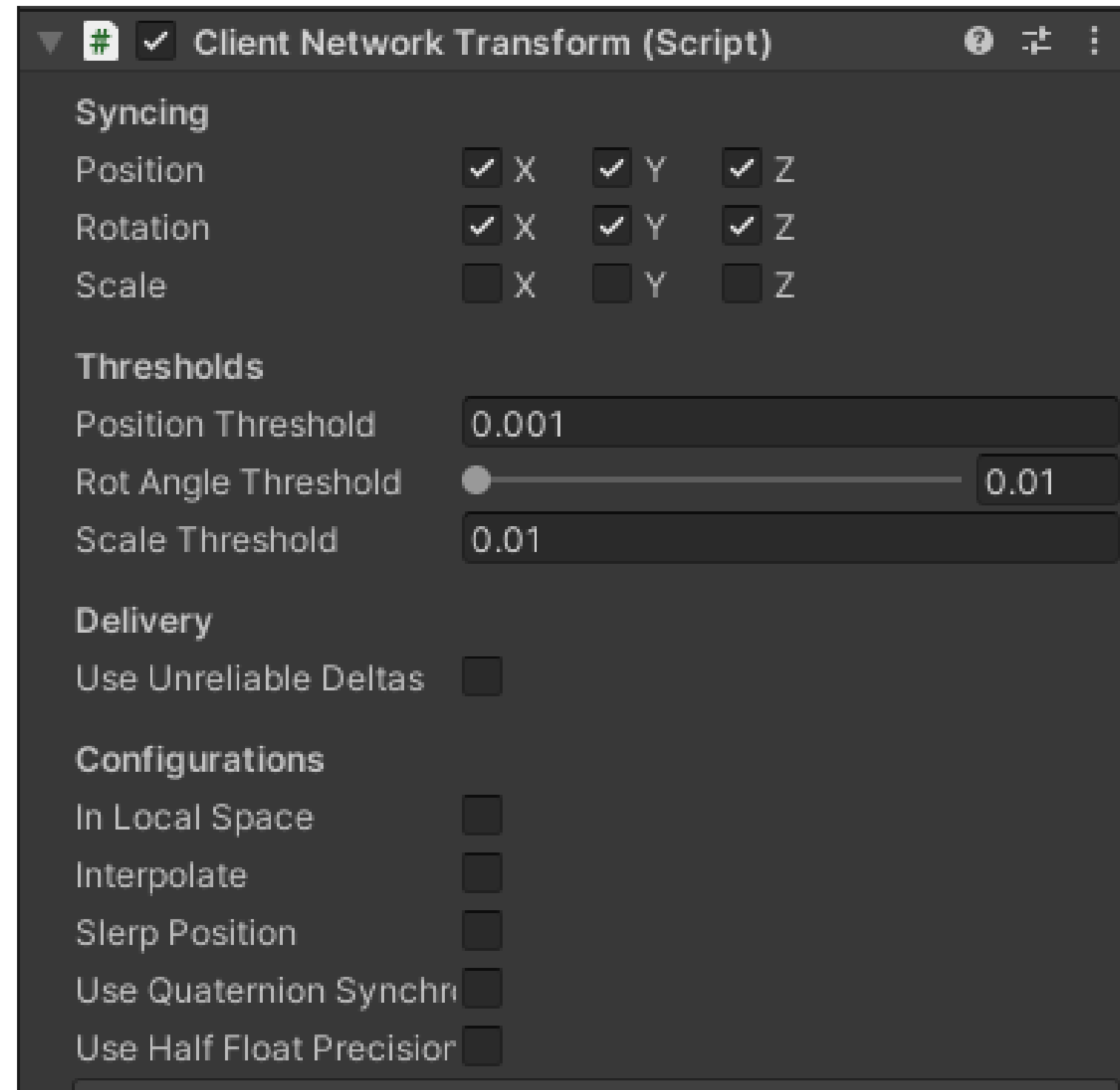
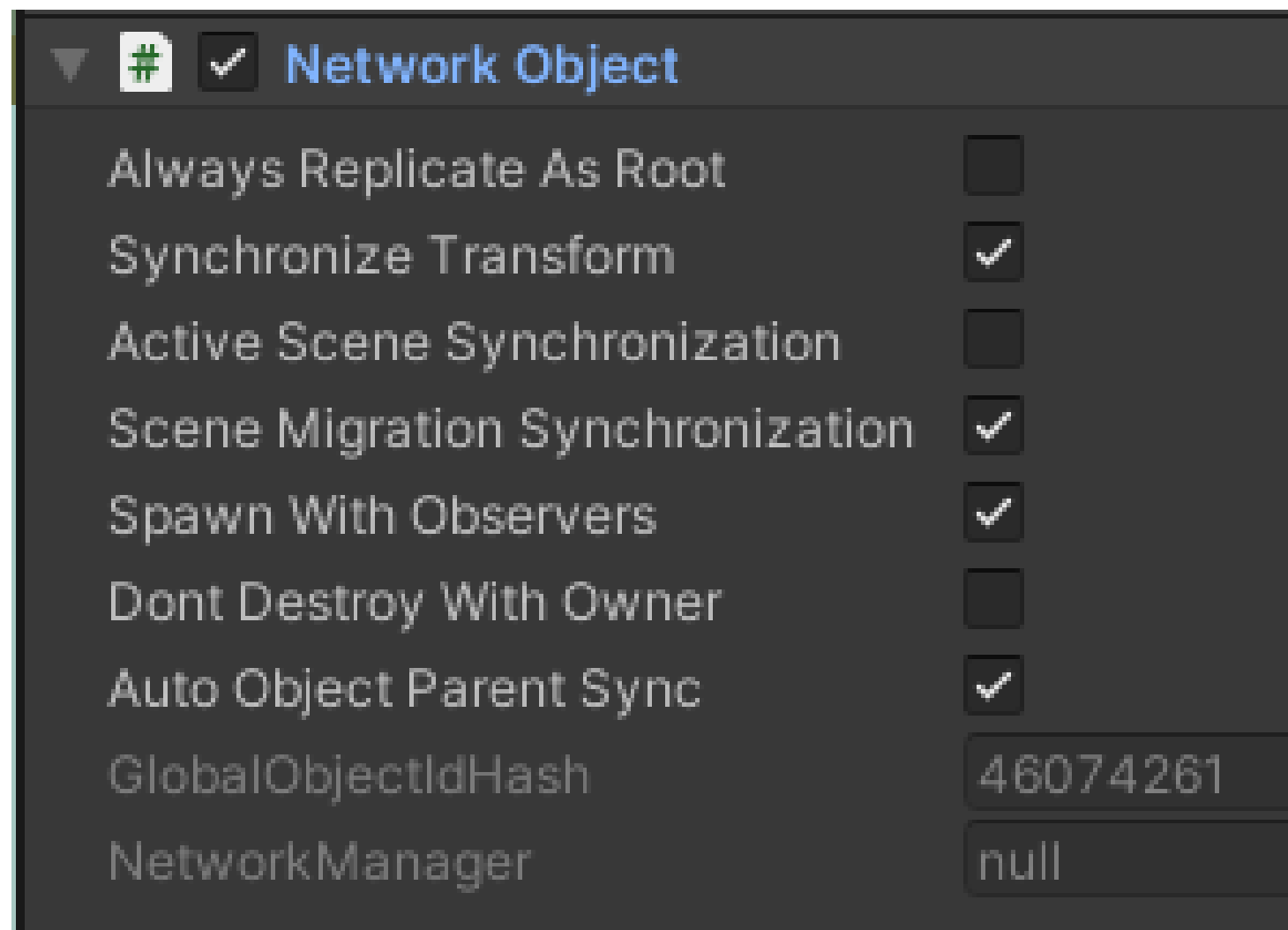


서버에서 참여 코드를 생성하고 클라이언트가 입력하여 참여하는 방식

핵심 구현 기능

1

네트워크 관련 기능

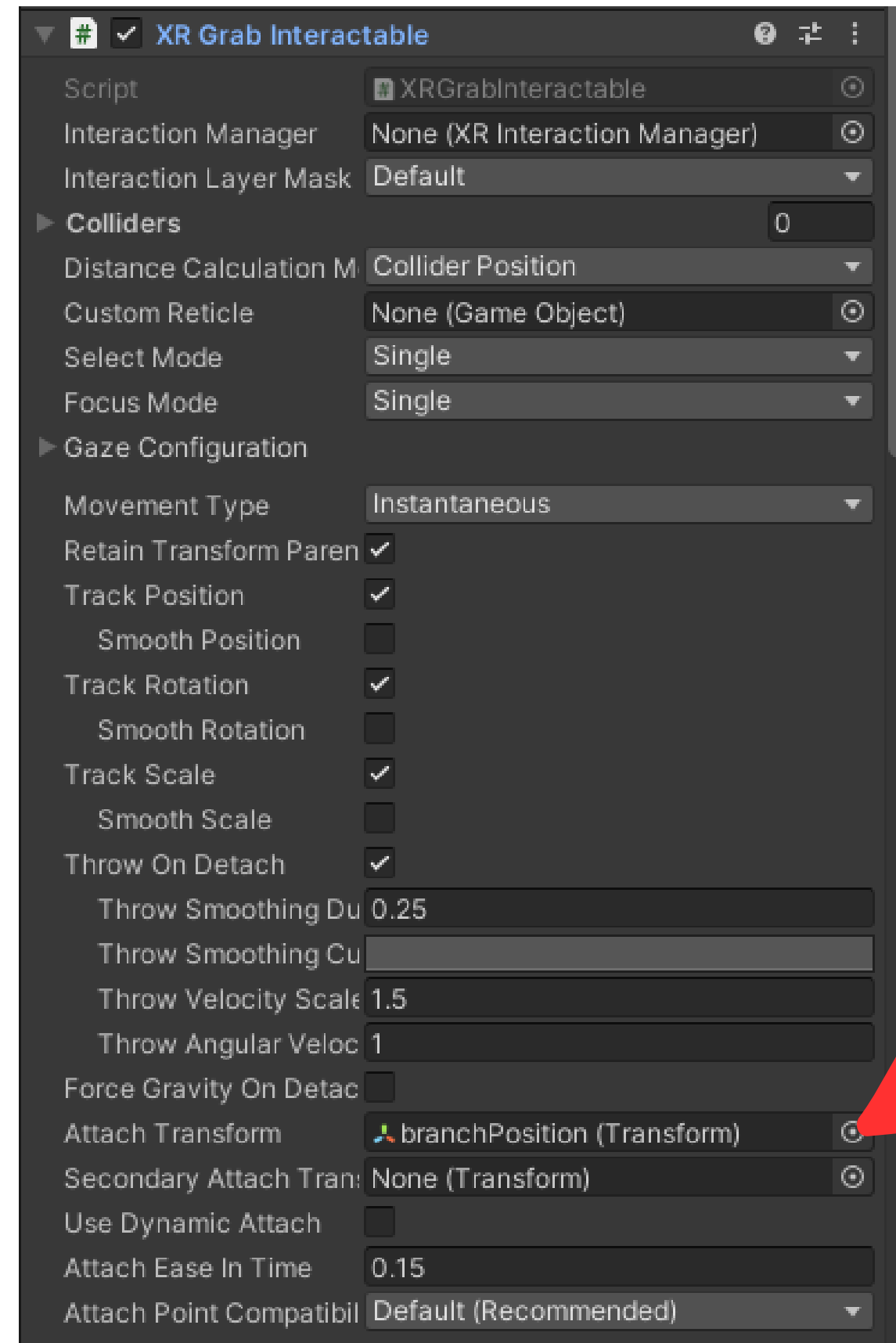


플레이어 및 오브젝트 동기화 = ClientNetworkTransform + NetworkObject 컴포넌트

핵심 구현 기능

1

네트워크 관련 기능



손(컨트롤러)에 붙는 위치

잡을 수 있는 오브젝트 = XR Grab Interactable(VR 컴포넌트)

핵심 구현 기능

1

네트워크 관련 기능

```
[ServerRpc(RequireOwnership = false)]  
참조 1개  
void RequestOwnershipServerRpc(ulong clientId)  
{  
    NetworkObject netObj = GetComponent<NetworkObject>();  
    if (netObj != null && netObj.OwnerClientId != clientId)  
    {  
        netObj.ChangeOwnership(clientId);  
        Debug.Log($"소유권을 클라이언트 {clientId}에게 넘김");  
    }  
}
```

소유권을 넘겨야
서로의 오브젝트 동기화가 이루어짐

오브젝트를 집었을 때, 집은 사람에게 소유권 넘기기 = ChangeOwnership(클라이언트ID)

핵심 구현 기능

2

메인 기능

```
if(key.activeSelf)
{
    // 오브젝트 사이 위치 계산
    float distance = Vector3.Distance(key.transform.position, gameObject.transform.position);

    if (distance < 0.2f)
    {
        key.SetActive(false);

        if(gameObject.name.Contains("1P"))
        {
            SetTeleporterCanPortByTag("Teleporter_A", true); // P1용
        }
        else
        {
            SetTeleporterCanPortByTag("Teleporter_B", true); // P2용
        }
    }
}
```

열쇠와 문이 가깝다면, 열쇠가 소모되고
텔레포트 할 수 있게 됨

```
else if (isPlayerPortal && other.CompareTag("Player") && canPort)
{
    Vector3 newPos = arrivePosObj.transform.position;
    newPos.y += yOffset;
    other.transform.position = newPos;

    ResetTrigger();

    // 일정 시간 후 다시 포탈을 활성화
    StartCoroutine(ReactivateTeleportersAfterDelay(4f)); // 3~5초 조절 가능
}

if (!isTeleported && (other.tag.Contains("key") || other.tag.Contains("Object")))
{
    Vector3 newPos = arrivePosObj.transform.position;
    newPos.y += yOffset;
    other.transform.position = newPos;
}
```

충돌한 오브젝트가 플레이어 텔레포터 = canPort가 true인지 확인 후 이동
태그에 key나 Object가 포함된 오브젝트 텔레포트 = 즉시 반대쪽으로 이동

핵심 구현 기능

2

메인 기능

오브젝트를 양 손으로 집은 상태에서
두 오브젝트를 부딪히면 새로운 오브젝트로 변환

CheckHandTransform(손 거리 계산 스크립트)의
CheckDistanceNCreate()를 통해 계산

MergeObjects(오브젝트 병합 스크립트)

MakeChessman(체스말 제작 스크립트)

핵심 구현 기능

2

메인 기능

오브젝트를 양 손으로 집은 상태에서
두 오브젝트를 부딪히면 새로운 오브젝트로 변환

```
if (CheckDistanceNCreate(objA, objB, mergedPrefab))
{
    Debug.Log("[MergeObjects] 병합 성공: " + mergedPrefab.name);
    isMerged = true;
    OnMergeCompleted?.Invoke(mergedPrefab); // 병합 완료 이벤트

    objA.SetActive(false);
    objB.SetActive(false);

    Invoke(nameof(ResetTrigger), resetDelay); // 일정 시간 후 병합 가능하게
    return;
}
```

MergeObject 스크립트의 병합 코드

```
public bool CheckDistanceNCreate(GameObject obj, GameObject obj2, GameObject newObj)
{
    float distance = Vector3.Distance(obj.transform.position, obj2.transform.position);
    Vector3 spawnPos = GetSpawnPosition(obj, obj2);
    float handDis = Vector3.Distance(leftHand.position, rightHand.position);

    //Debug.Log($"거리: {distance}, 손 거리: {handDis}"); // 디버그 로그 추가

    if (!obj.activeSelf || !obj2.activeSelf) return false;

    if (xr_input.isLPressed && xr_input.isRPressed && distance < 1.0f && handDis < 0.15f)
    {
        Debug.Log("충분히 가까움");

        var obj1Net = obj.GetComponent<NetworkObject>();
        var obj2Net = obj2.GetComponent<NetworkObject>();

        if (IsServer)
        {
            obj.SetActive(false);
            obj2.SetActive(false);

            GameObject spawned = Instantiate(newObj, spawnPos, Quaternion.identity);
            var spawnedNet = spawned.GetComponent<NetworkObject>();
            spawnedNet.Spawn();
        }
    }
}
```

CheckHandTransform 스크립트의 주요 함수

```
if (jar.name.Contains("KnightHead") && !spawnedChessman[0])
{
    if (CheckDistanceNCreate(jar, merged, chessman1))
    {
        Debug.Log("Knight 생성됨");
        spawnedChessman[0] = true;
        SendSpawnRequest(jar, merged);
    }
}
```

MakeChessman 스크립트의 병합 코드

핵심 구현 기능

2

메인 기능

```
private void Start()
{
    objects1 = new List<GameObject>();
    objects2 = new List<GameObject>();
}

[ServerRpc(RequireOwnership = false)]
참조 1개
public void InsertKeyServerRpc(ulong netId, string name)
{
    NetworkObject netObj = NetworkManager.Singleton.SpawnManager.SpawnedObjects[netId];

    GameObject obj = netObj.gameObject;

    if (name.Contains("Jar_Knight"))
    {
        objects1.Add(obj);
        cnt1++;
    }
    else if (name.Contains("Jar_Rook"))
    {
        objects2.Add(obj);
        cnt2++;
    }

    if (cnt1 == 2 && !isInit)
    {
        isInit = true;
        var newObj = Instantiate(resultObj1, transform.position, Quaternion.identity);
        newObj.GetComponent<NetworkObject>().Spawn();
    }
}
```

항아리에 특정 오브젝트를 넣었을 때,
새로운 오브젝트로 변환

```
private void Update()
{
    if (assignedKey == null || leftHand == null || rightHand == null)
    {
        return;
    }

    float distL = Vector3.Distance(leftHand.position, assignedKey.transform.position);
    float distR = Vector3.Distance(rightHand.position, assignedKey.transform.position);

    if (distL < 0.1f || distR < 0.1f)
    {
        hoverTimer += Time.deltaTime;

        Debug.Log($"Hovering: {OwnerClientId}"); // 디버그 로그 추가

        if (hoverTimer >= hoverThreshold)
        {
            manager.ReportHoverComplete(OwnerClientId);
        }
    }
    else
    {
        hoverTimer = 0f;
    }
}
```

```
if (p1Complete && p2Complete)
{
    Debug.Log("두 플레이어 모두 키에 손을 댄! 문이 열립니다.");

    // 플레이어 태그를 가진 오브젝트 찾기
    GameObject[] players = GameObject.FindGameObjectsWithTag("Player");

    // 플레이어 전부 비활성화
    foreach (GameObject player in players)
    {
        if (player.name.Contains("Player0"))
        {
            player1 = player;
            player.SetActive(false);
        }
        else if (player.name.Contains("Player1"))
        {
            player2 = player;
            player.SetActive(false);
        }
    }

    // 엔딩 카메라 활성화
    cameraObj.SetActive(true);
}
```

특정 오브젝트를 두 플레이어가 동시에 대고 있으면 엔딩

핵심 구현 기능

3

그래픽 관련 기능

- URPSHader를 기반으로한 스텐실 기능 구현.



• 스텐실 적용전



• 스텐실 적용 후

- URP렌더파이프라인에 Render Objects내의 Layer Mask를 사용해 StencilMask를 적용했습니다.

소감

소감

김다희

처음으로 VR 멀티 플레이 게임을 제작해보았습니다. 많은 어려움과 기간 부족으로 인해 힘든 시간을 보냈지만, 한편으로는 시작부터 엔딩까지 무사히 마쳤다고 생각합니다. 경서와 마무리 테스트를 할 때에는 정말 다른 게임을 같이 하듯이 즐긴 것 같아 뿌듯합니다.

남경서

졸업작품 마무리를 하며, 한편으로는 후련하고 허무한 기분입니다. 첫 VR게임을 제작하며 어려움도 많았고, 막막했지만 좋은 팀원과 함께 업무를 분담하여 진행한것이 이번 프로젝트가 좋게 마무리 될 수 있었던 이유 같습니다. 부족한 면도 보이지만 그럼에도 재미있는 게임을 만들게 된 것 같아 기쁩니다 ^^

마무리

Q&A