

O R B I T

졸업작품 오르빗


게임 개요



○ 장르

- AI 몬스터를 처치하여 목표를 달성하는 싱글플레이 / 플레이어들끼리의 전투를 지원하는 멀티플레이로 구성된 FPS게임

○ 세부 정보

- 게임 엔진 :  Unity 6 HDRP
- 네트워크 : Mirror API – 리슨 서버 방식 사용
- 데이터베이스 : MariaDB
- 데이터베이스 클라우드 : Google Cloud SQL

○ 주요 콘텐츠

- 싱글플레이 :
 1. 구역 점령 시스템 (점령한 지역에서는 몬스터가 스폰되지 않음)
 2. 무기 교환 시스템 (몬스터를 처치하여 일정확률로 얻거나, 구역을 점령하여 대량으로 획득할 수 있는 '온전한 칩'을 통해 무기 구매 가능)
 3. 업적 시스템 (싱글플레이에서 누적된 데이터를 확인할 수 있음)
 4. 로컬 플레이 데이터 저장으로, 실시간 데이터 저장
- 멀티플레이 :
 1. 멀티 플레이어간 처치 가능(처치한 플레이어는 경험치 증가, 처치당한 플레이어는 경험치 감소)
 2. 실시간 채팅 시스템
 3. 클라우드 DB 서버에 데이터를 저장하여 어디서든 동일한 데이터로 플레이 가능

주요 기능

○ 캐릭터

- 상 / 하체 애니메이션 시스템 분리(UpperBody 사용)를 통한 다중 애니메이션 제어
- 총기 발사시 반동(recoil) 시스템 (총알 발사시 총기가 위로 올라감)
- 앉기 / 달리기 등의 애니메이션이 별도 구현되어있고, 멀티플레이에서 플레이어의 피격 / 사망 애니메이션 구현

○ 몬스터

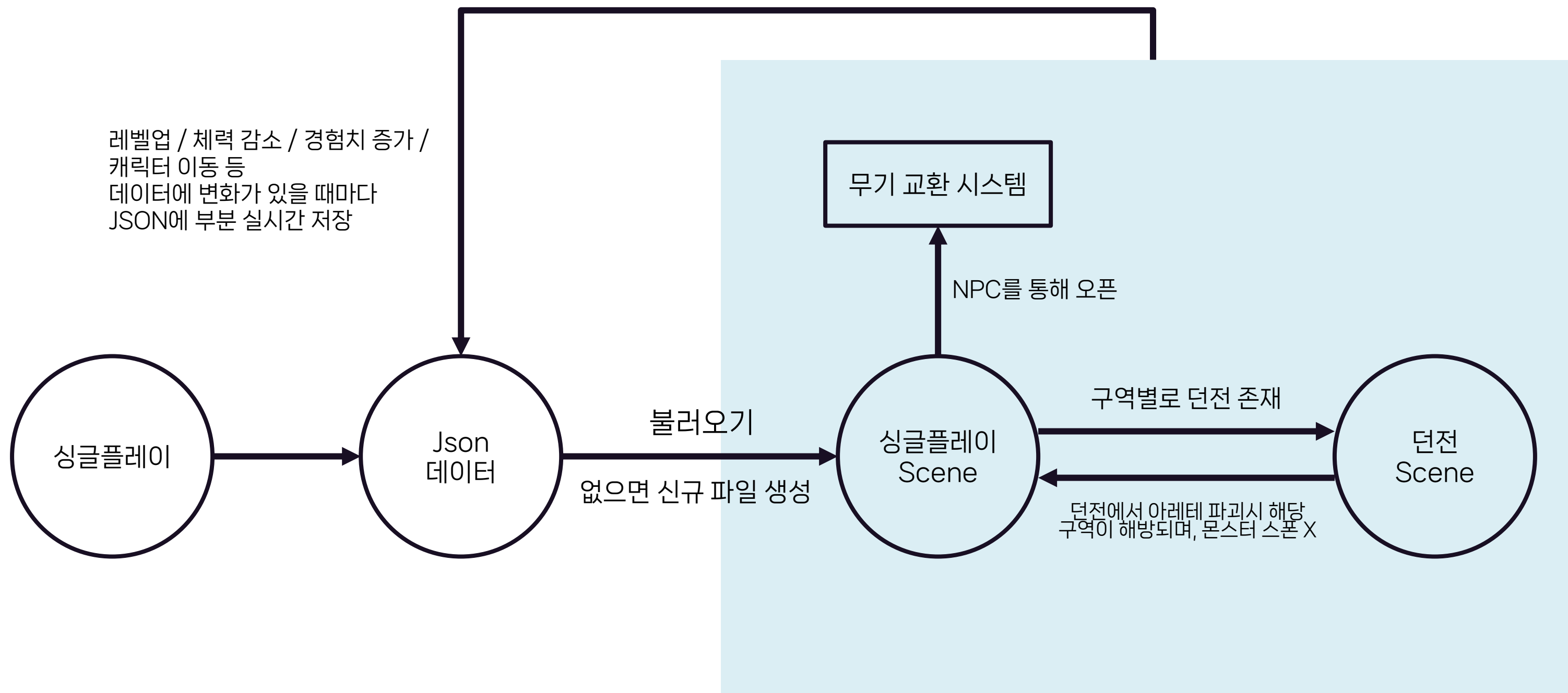
- NavMesh AI를 사용해 AI 몬스터가 길을 찾아 이동
- 5가지 EnemyState를 통해 IDLE, WANDER, ATTACK 등의 상태 구현

○ 멀티플레이

- SyncVar(hook)을 통해 개별 플레이어의 HP를 모든 클라이언트와 싱크 처리 → 플레이어의 HP에 변동이 있으면 모든 클라이언트에 동기화됨
- 플레이어 피격시 클라이언트가 서버로 요청을 보내 서버에서 피격 요청을 처리 (Command), 이 때 피격한 플레이어의 connectionId 값을 기억해둠.
- 피격당한 플레이어가 사망시 피격한 플레이어의 connectionId 값에 TargetRPC를 보내 처치알림과 경험치 획득을 부여

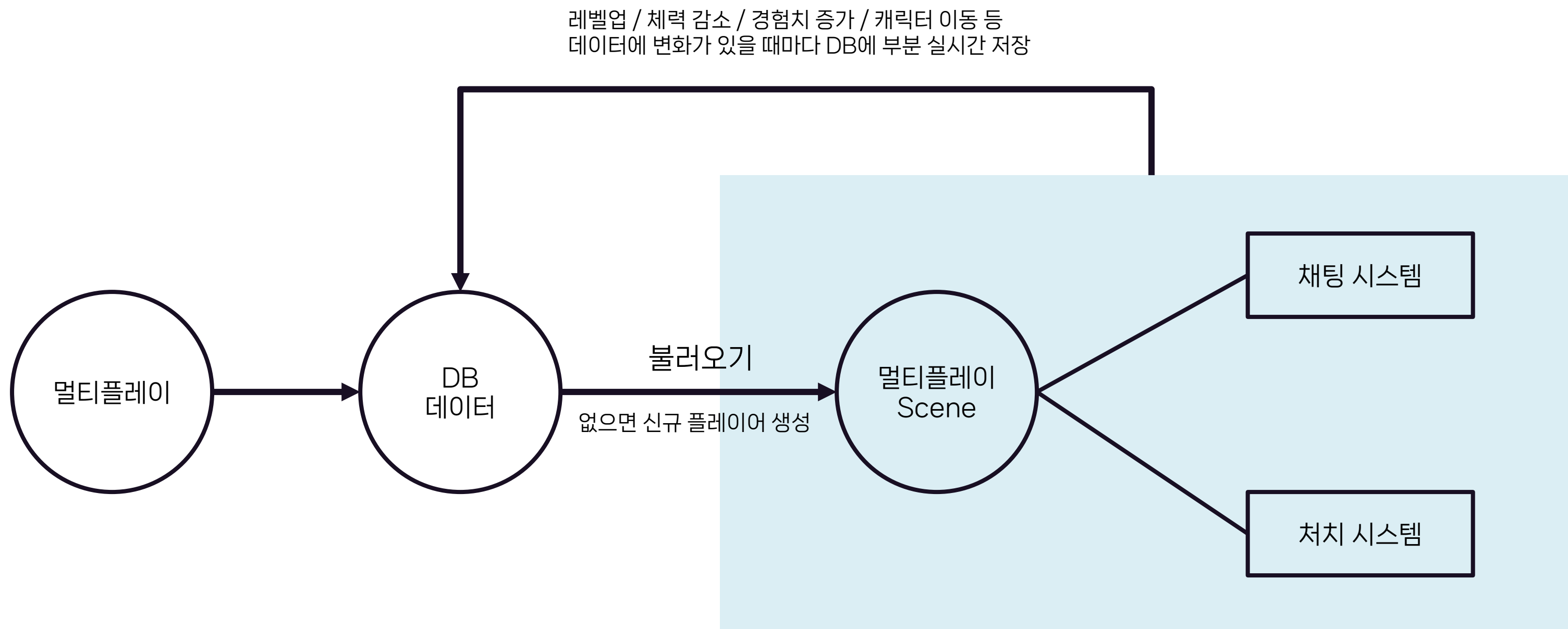
시스템 구성

싱글플레이

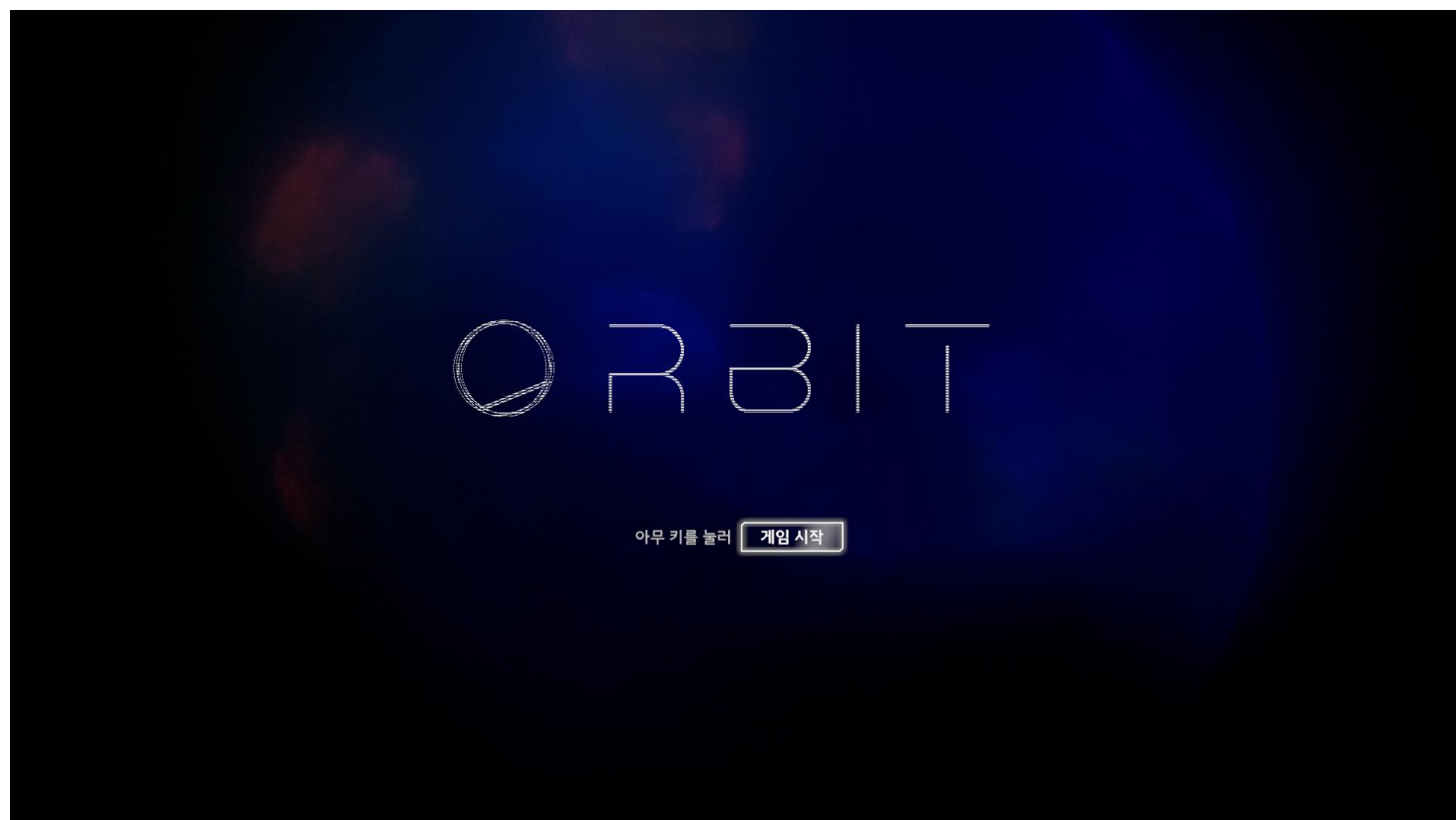


시스템 구성

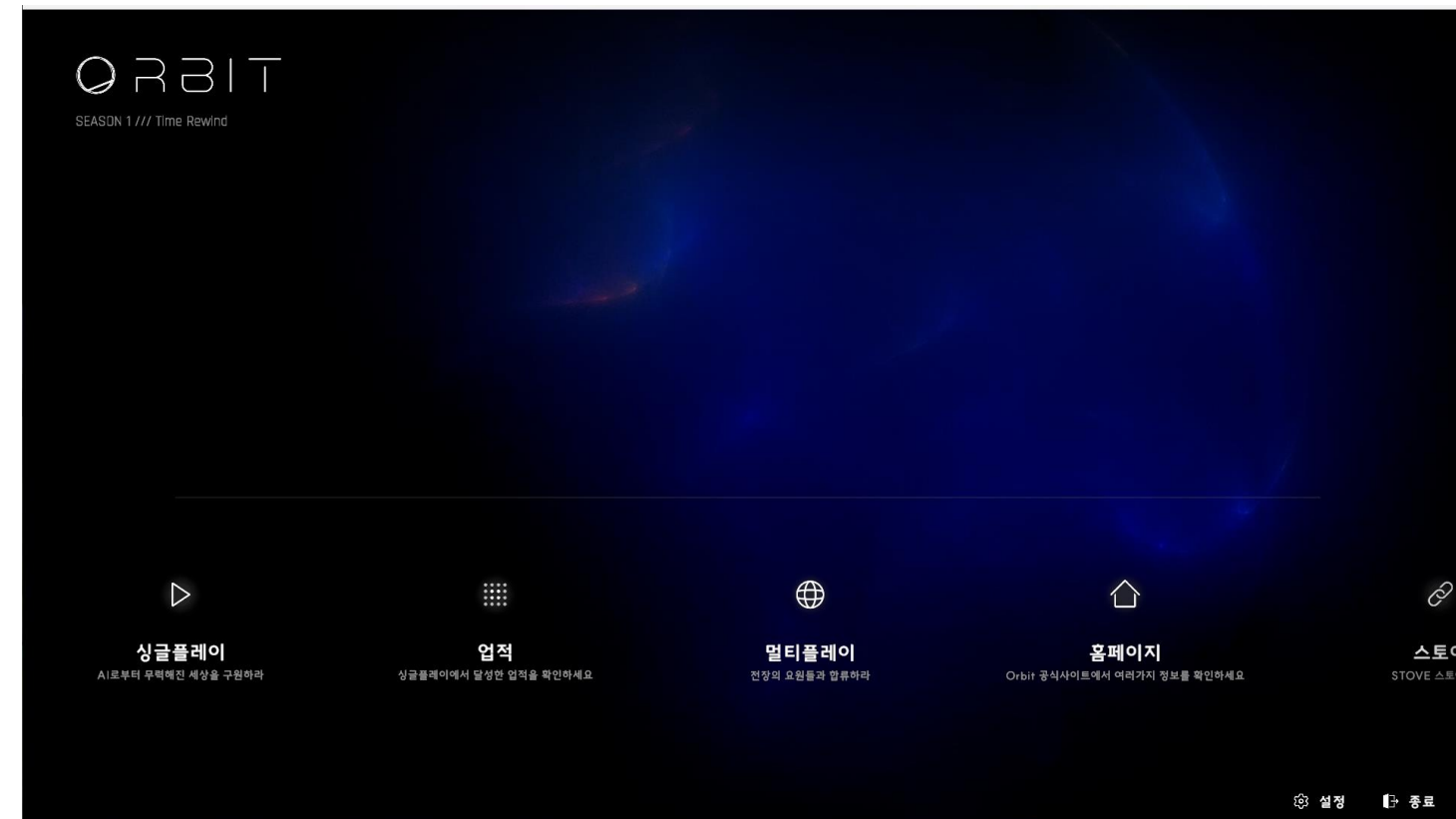
멀티플레이



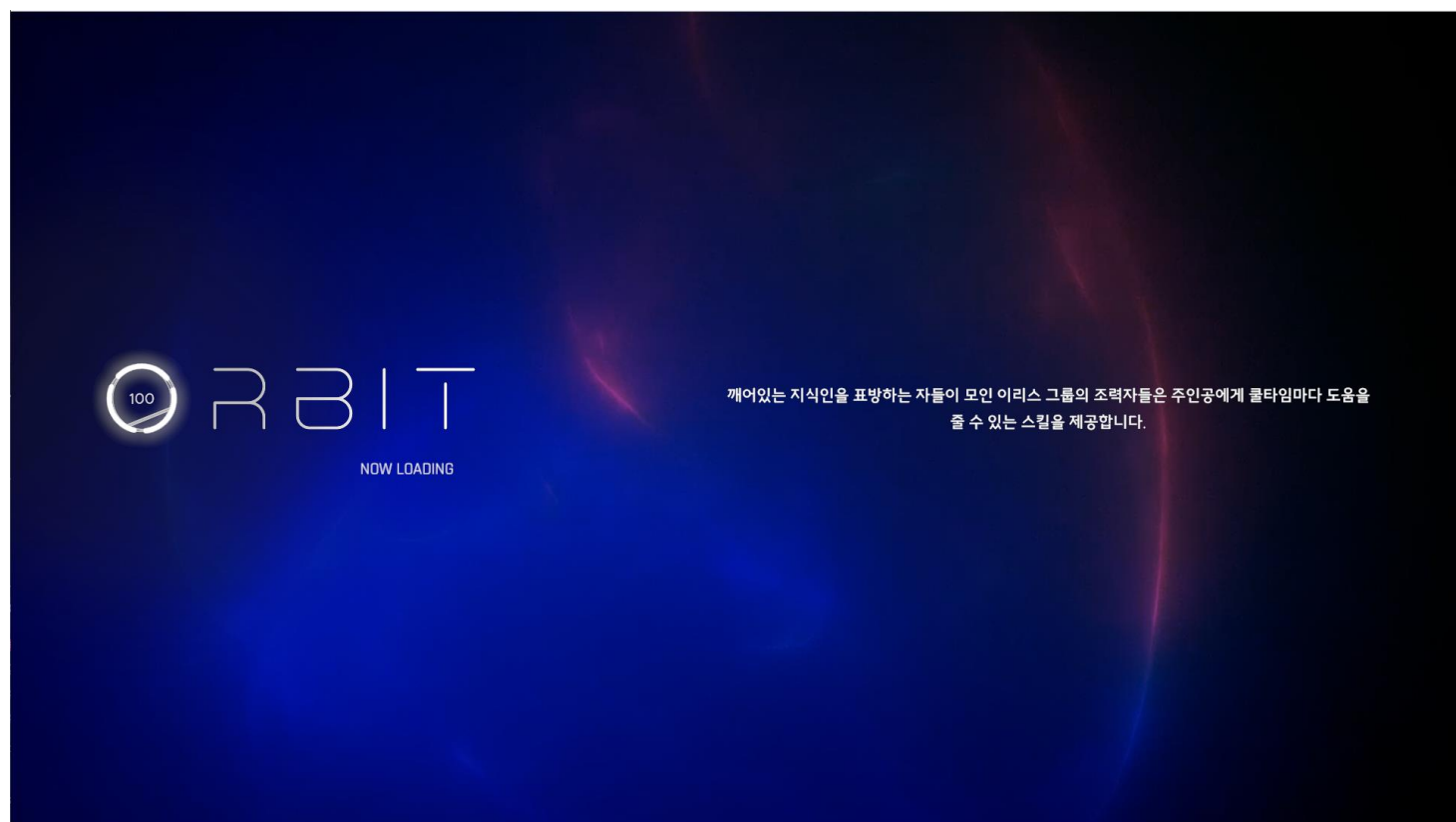
주요 UI



메인(게임시작)



로비



로딩화면



인게임 HUD

콘텐츠 소개

싱글플레이 - 구역 점령 시스템

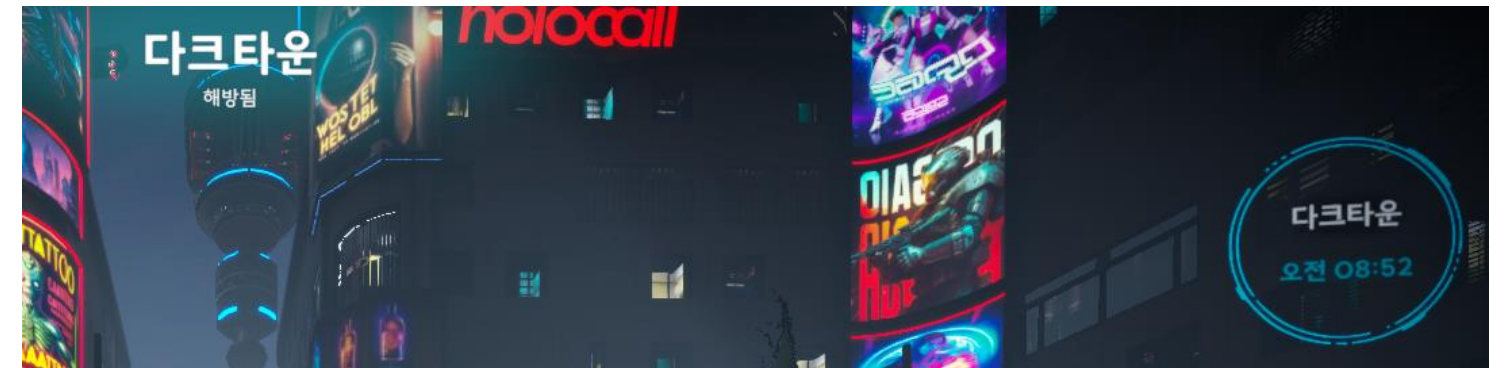
- 싱글플레이의 월드는 여러가지 구역으로 되어있음
- 각 구역에 존재하는 여신상을 통해 던전에 입장한 후 아레테를 파괴해야 해당 지역이 해방됨



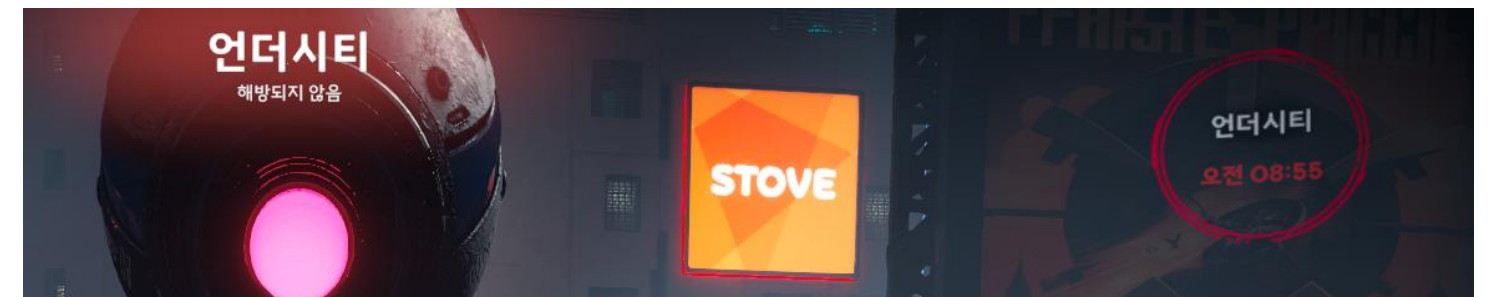
여신상



아레테



해방된 구역



미해방된 구역

콘텐츠 소개

싱글플레이 - 구역 점령 시스템(몬스터)

```

_enemyMemoryPool = new ObjectPool<GameObject>(
    createFunc: () =>
    {
        GameObject enemy = Instantiate(enemyPrefab);

        // 몬스터가 죽을 때 풀로 반환
        enemy.GetComponent<EnemyFSM>().OnDeath += () =>
        {
            _enemyMemoryPool.Release(enemy);
            _currentEnemyCount--; // 현재 활성화 몬스터 수 감소
        };

        return enemy;
    },
    actionOnGet: item =>
    {
        item.SetActive(true);
        item.GetComponent<EnemyFSM>().ResetState(); // 상태 초기화
        item.name = enemyPrefab.name; // 이름 변경
    },
    actionOnRelease: item =>
    {
        item.SetActive(false); // 비활성화 처리
    },
    actionOnDestroy: Destroy,
    collectionCheck: false,
    defaultCapacity: 10,
    maxSize: 30
);

```

```

public enum EnemyState
{
    NONE = -1,
    IDLE = 0,
    WANDER,
    PURSUIT,
    ATTACK
}

```

```

private void Update()
{
    FloatingEffect();

    EnemyState newState = CalculateDistanceToTargetAndSelectState();
    if (newState != _currentState)
    {
        ChangeState(newState);
    }
}

```

- 싱글플레이의 미해방 구역에서 두 가지 종류의 몬스터가 스폰됨
두 몬스터는 각기 다른 ObjectPool에서 소환되며, 각기 다른 공격 사거리, 공격력 등을 지님
- 소환된 몬스터는 NONE, IDLE, WANDER, PURSUIT, ATTACK의 상태값을 가지며, Update 함수에서 계속 조건 값을 계산하며 Coroutine 함수를 통해 상태 함수를 실행함

ObjectPool :

- 대량으로 생성되는 몬스터 등의 오브젝트를 Instantiate를 통해 각각 인스턴스를 생성하는 것이 아니라, 오브젝트 풀에서 관리하여 메모리를 절약할 수 있음
- 몬스터를 처치할 때 Destroy 처리가 아닌 ObjectPool에 반환처리를 함

콘텐츠 소개

싱글플레이 - 무기 교환 시스템

- 필드에 존재하는 몬스터를 처치하여 일정 확률로 획득하거나, 던전의 아레테를 파괴하여 대량으로 획득할 수 있는 온전한 칩을 사용하여 구매 가능
- 잠겨있는 무기를 새롭게 구매하거나, 기존에 구매했던 무기를 장착하거나, 장착된 무기의 순서를 변경할 수 있음

```

FPSItem weapon = fpsController._instantiatedWeapons[i];
Weapon weaponC = weapon.gameObject.GetComponent<Weapon>();
GunFire gunFire = weapon.gameObject.GetComponent<GunFire>();

GameObject weaponButton = Instantiate(weaponButtonPrefab, leftWeaponListParent);

// 무기 이미지
Image weaponImage = weaponButton.transform.Find("WeaponImage").GetComponent<Image>();
weaponImage.sprite = weapon.weaponPreview;

// 무기 이름
TMP_Text weaponName = weaponButton.transform.Find("WeaponName").GetComponent<TMP_Text>();
weaponName.text = weapon.name.Replace("(Clone)", "").Trim();

// 무기 가격
TMP_Text cost = weaponButton.transform.Find("Lock/LockText/Chip/ChipCost").GetComponent<TMP_Text>();
cost.text = weapon.cost.ToString();

//무기 정보
TMP_Text weaponInfo = weaponButton.transform.Find("WeaponInfo").GetComponent<TMP_Text>();
weaponInfo.text = $"연사속도 {weaponC.fireRate} / 데미지 {gunFire.damage}";

Button purchaseButton = weaponButton.transform.Find("Lock/LockText/PurchaseButton").GetComponent<Button>();
Button equipButton = weaponButton.transform.Find("EquipButton").GetComponent<Button>();
GameObject equipWeapon = weaponButton.transform.Find("EquipWeapon").gameObject;
  
```

무기 프리팹에 할당되어 있는 정보를 불러와서 리스트 아이템에 할당

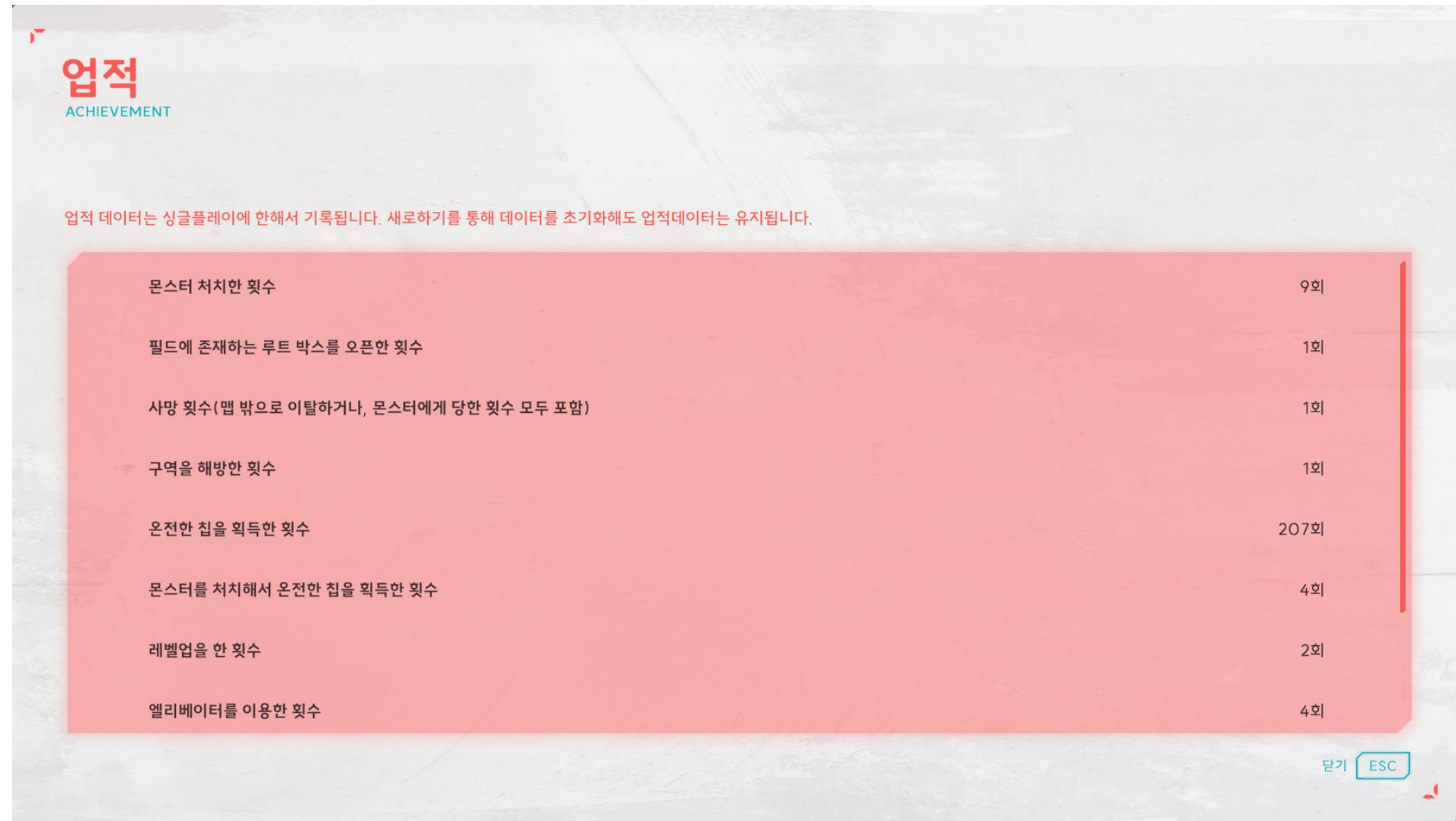


무기 교환 시스템 UI

콘텐츠 소개

싱글플레이 - 업적 시스템

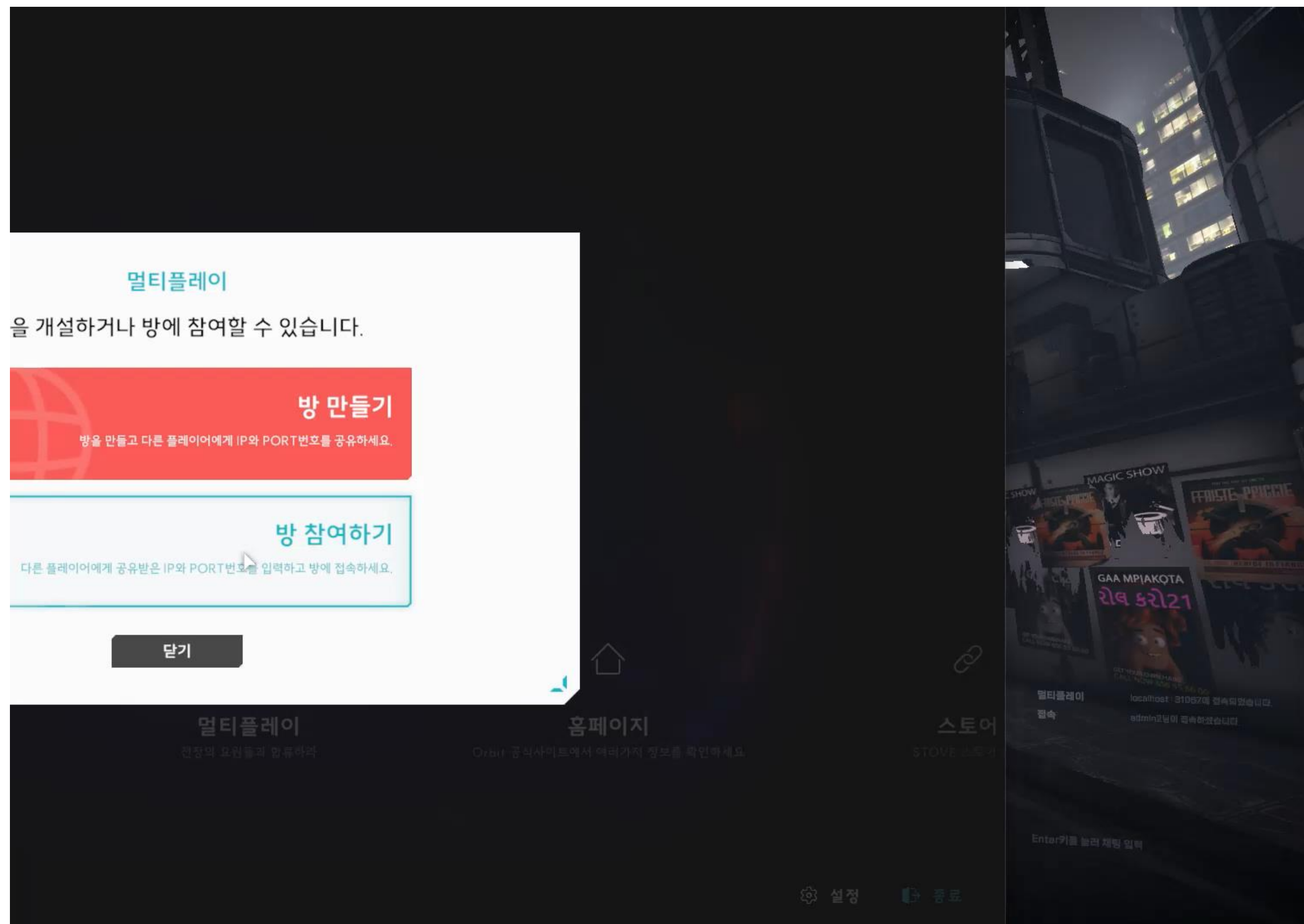
- 싱글플레이에서 진행한 내용을 기반으로 업적이 기록됨
- 업적 데이터는 싱글 플레이 데이터와 별도로 저장되어, 싱글 플레이 데이터를 초기화(새로하기)해도 유지됨



콘텐츠 소개

멀티플레이 - 채팅 시스템

- 멀티플레이에서 채팅창을 통해 다른 플레이어의 입/퇴장 알림을 받을 수 있음
- 플레이어 닉네임과 채팅 내용의 전송을 통해 실시간 소통을 할 수 있음
- 모든 채팅 내용은 [Command]를 통해 서버로 전송된 후 [ClientRPC]로 모든 클라이언트에게 메시지를 전달



콘텐츠 소개

멀티플레이 – 처치 시스템

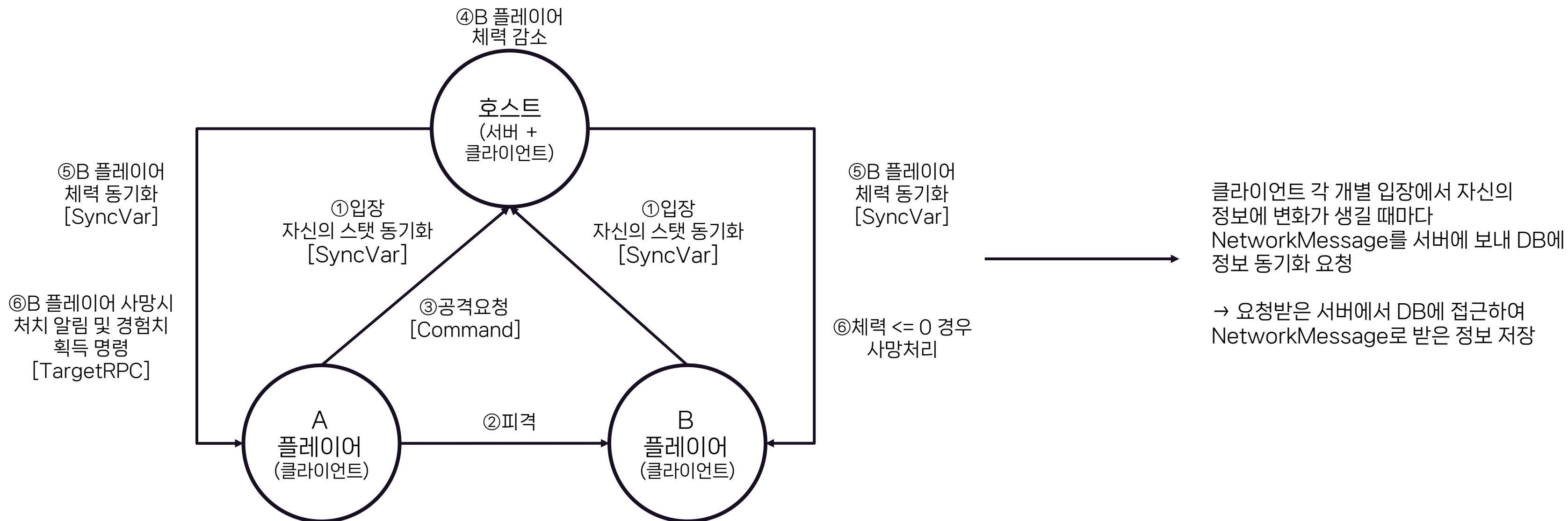
- 멀티플레이에서는 플레이어가 다른 플레이어를 처치할 수 있음
- 처치한 플레이어는 경험치를 획득하고, 처치당한 플레이어는 현재 경험치의 30%를 잃게 됨



콘텐츠 소개

멀티플레이 – 처치 시스템(네트워킹 과정)

- 멀티플레이에서 플레이어 피격시 실행되는 Network Flow



시연

ORBIT

아무 키를 눌러

게임 시작



ORBIT

감사합니다



<https://github.com/starting-run>



<https://starting.run/>



<https://store.onstove.com/ko/games/3783>