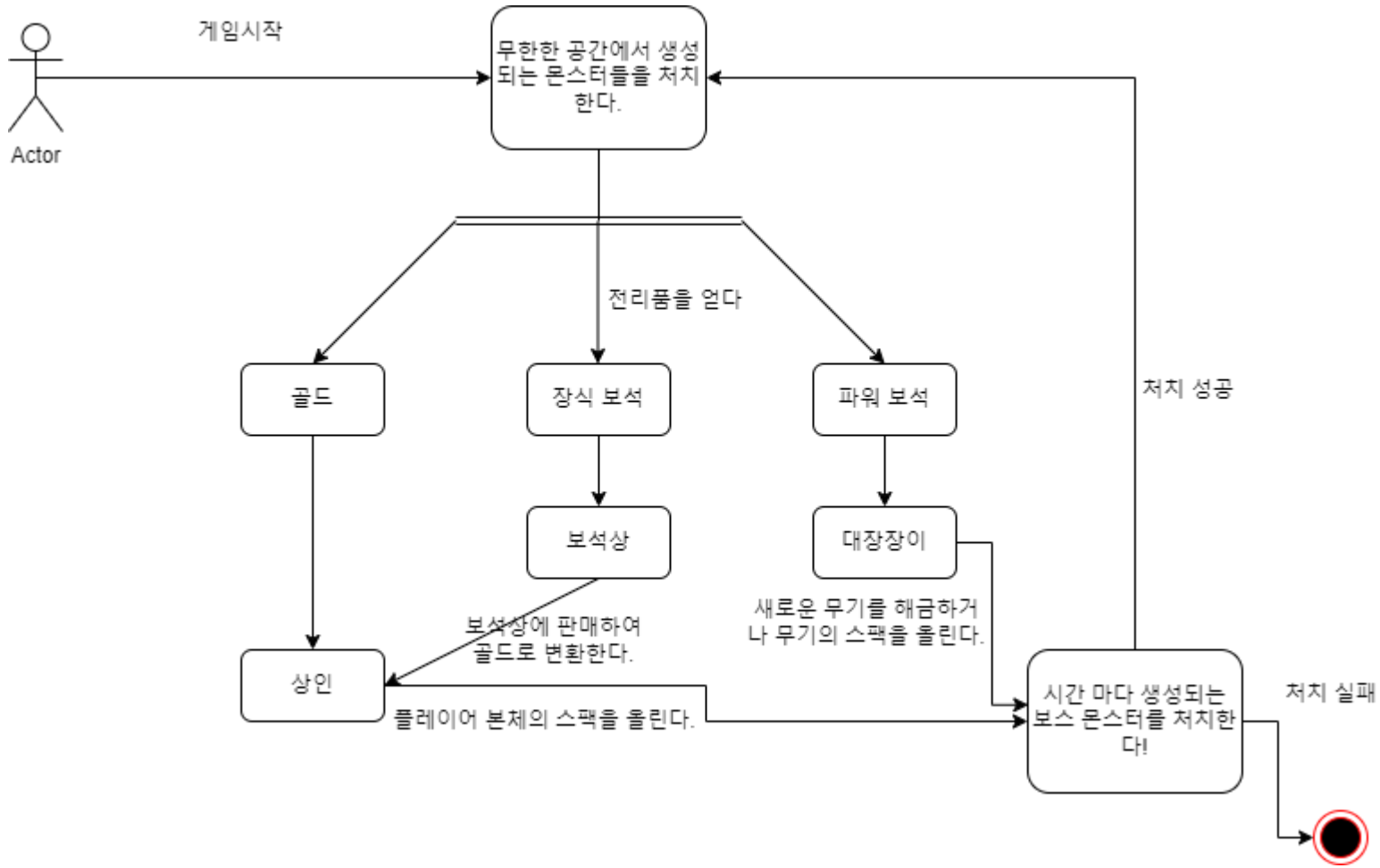

LiveALone
START
Unity로 제작된 뱀파이어서바이벌라이크
Github
Play

01 기획

게임 흐름도

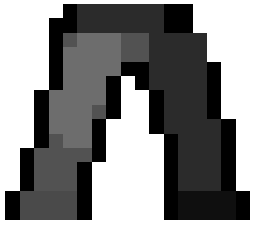

게임시작
Actor
무한한 공간에서 생성되는 몬스터들을 처치한다.
전리품을 얻다
골드
장식 보석
파워 보석
보석상
대장장이
보석상에 판매하여 골드로 변환한다.
새로운 무기를 해금하거나 무기의 스팩을 올린다.
상인
플레이어 본체의 스팩을 올린다.
시간 마다 생성되는 보스 몬스터를 처치한다!
처치 성공
처치 실패

02 구현

플레이어

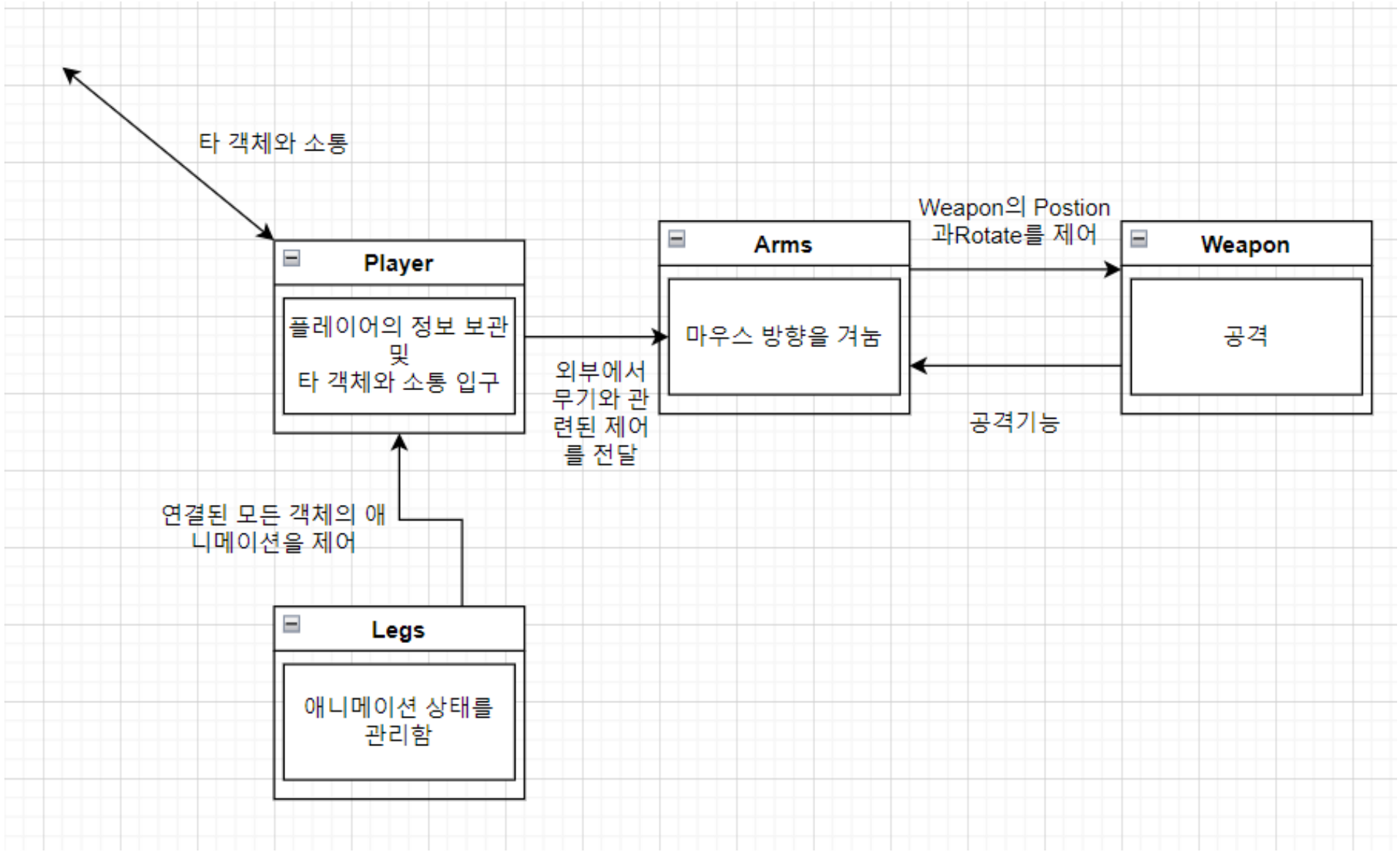
Body – Player 객체의 최상위 객체
플레이어로서 가져야 할 정보들을 관리하며,
타 객체와 플레이어의 이름으로 소통하는 통로 입니다.

Arms– Player가 장비한 무기들을 관리하는 객체
플레이어 객체와 무기를 연결해주는 중간관리자 입니다.



Legs– Player의 애니메이션을 관리하는 객체 입니다.

플레이어



하이라키 상의 구조

플레이어

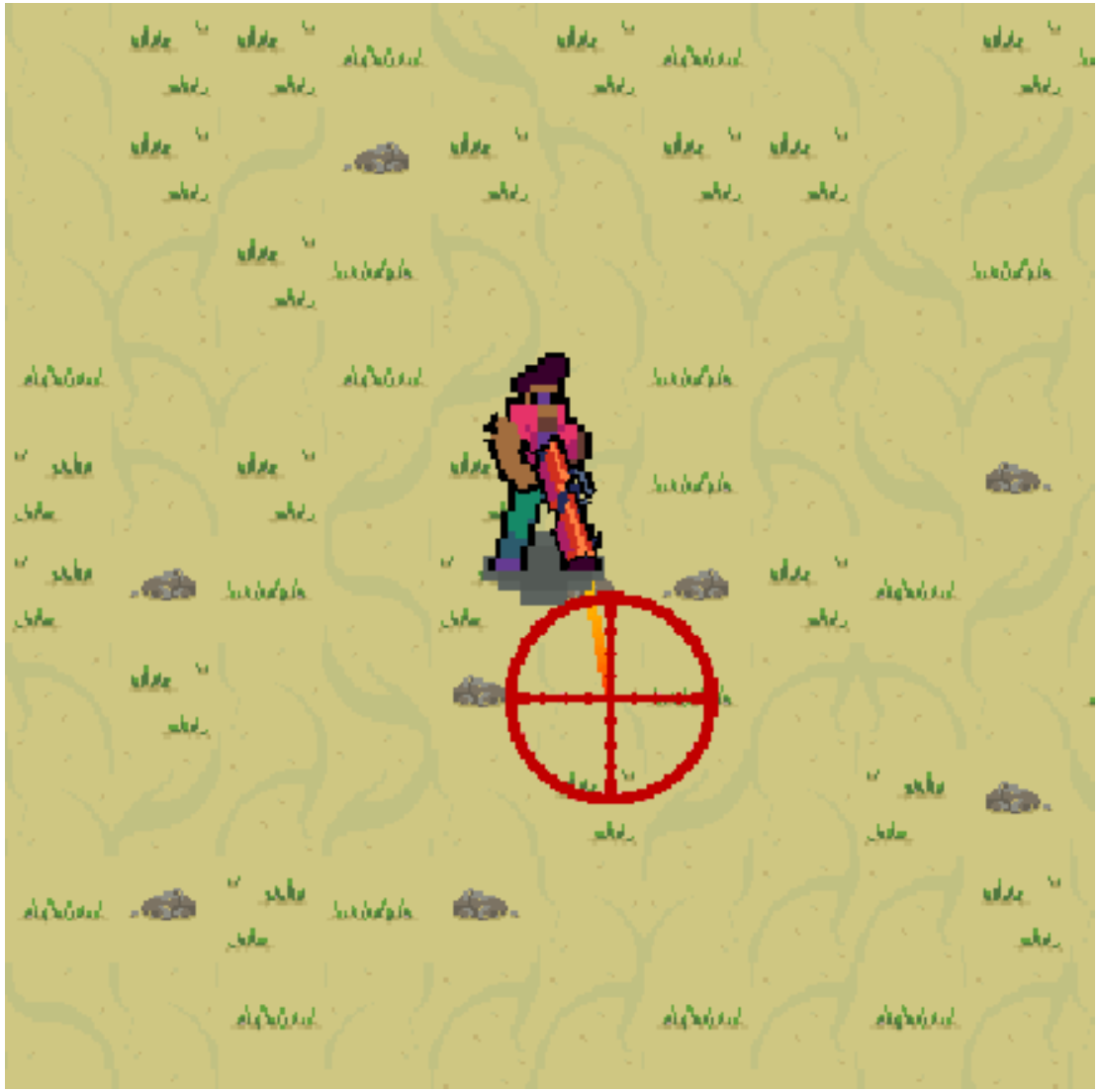
```
Unity 메시지 | 참조 0개
void Awake()
{
    hp = max_hp;
    equipmentWeaponNumber = 0;
    aimPointer = GetComponentInChildren<AimPointer>();
    Rigidbody2D = GetComponent<Rigidbody2D>();
    spriteRenderer = GetComponent<SpriteRenderer>();
    animator = GetComponent<Animator>();
    scanner = GetComponent<Scanner>();
    legs = transform.Find("PlayerLegs");
    arms = transform.Find("PlayerArms");
    armSpriteRenderer = arms.GetComponent<SpriteRenderer>();
    legSpriteRenderer = legs.GetComponent<SpriteRenderer>();
    legAnimator = legs.GetComponent<Animator>();
}
```

플레이어 객체가 선언되면서
플레이어의 파츠, arms와 legs의 컴포넌트를 가져옵니다.

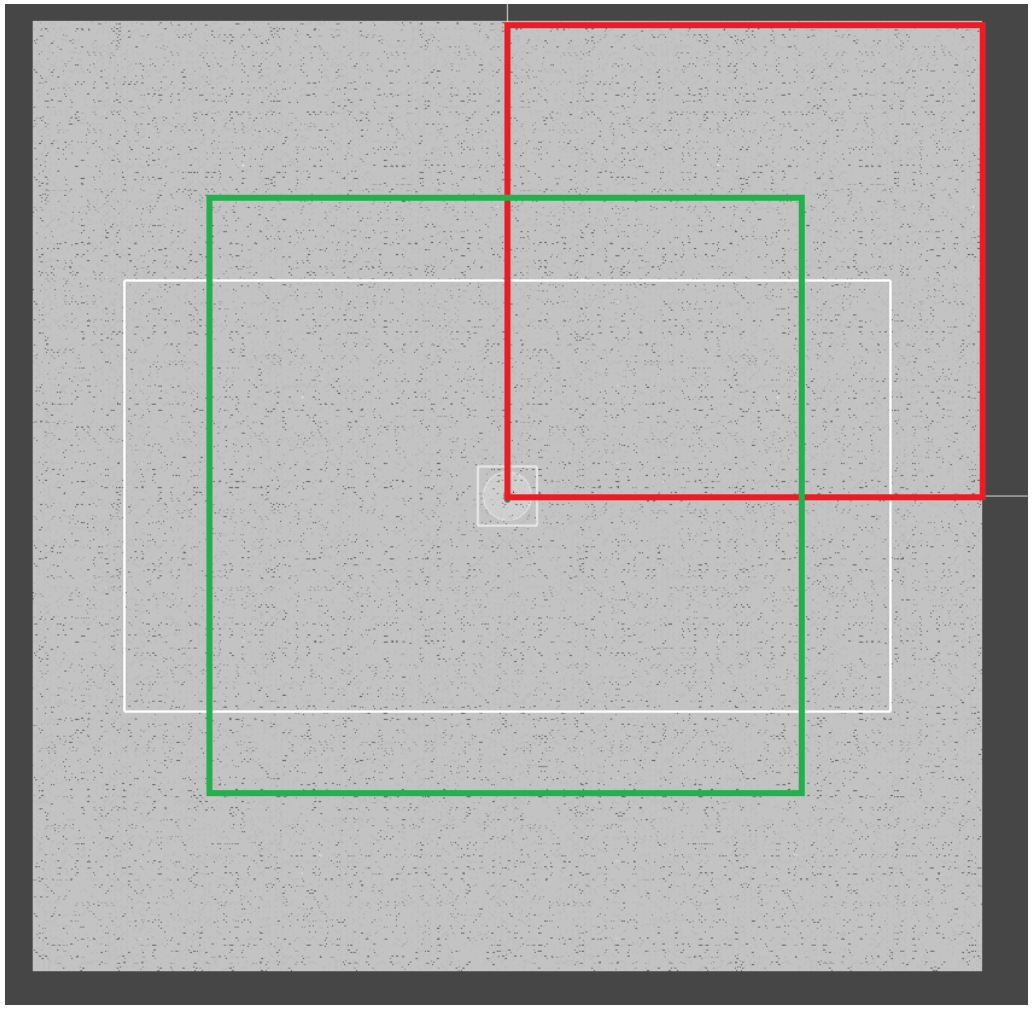
```
Unity 메시지 | 참조 0개
void LateUpdate()
{
    //exception
    if (!GameManager.instance.isLive) { return; }

    //act
    //마우스 방향을 바라보도록
    Vector3 aimPointPos = aimPointer.transform.position;
    Vector3 myPosition = transform.position;
    Vector3 direction = (aimPointPos - myPosition).normalized;

    float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;
    float normalizedAngle = (angle + 360) % 360;
    spriteRenderer.flipX = normalizedAngle >= 90 && normalizedAngle < 270;

    //다리의 방향
    legAnimator.SetFloat("Speed", vec_input.magnitude);
    if (vec_input.x != 0)
    {
        legSpriteRenderer.flipX = vec_input.x < 0;
    }
}
```

플레이어의 LateUpdate에서
몸통은 마우스를 바라보도록, 다리는
이동 방향을 보여주도록 설정 합니다.

플레이어 이동



2D 무한맵 구현

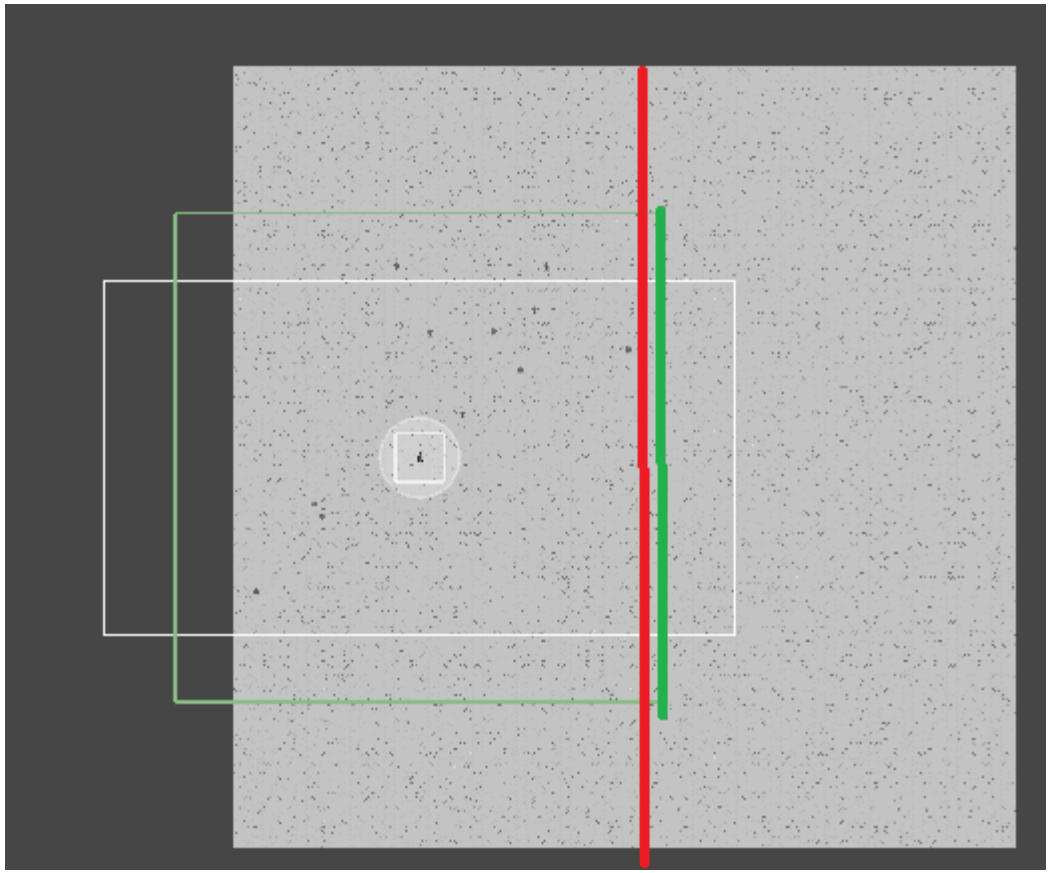


(적색)
하나의 Cell은 32*32 pix의 크기
하나의 Grid는 80*80 cell의 크기

(녹색)
감지 Area는 100*100 Cell의 크기

(백색)
플레이어가 보는 카메라의 크기
640 * 360 pix의 크기 (1920*1080 해상도의 사항)

2D 무한맵 구현



Area는 OnTriggerExit2D로 본인 범위의 콜라이더 밖으로 빠져나가는 Tile grid를 감지 합니다.

해당 타일맵을 분석하여, 플레이어의 동서남북 어디에 있었는지 찾아 낸 다음 자리하고 있던 장소에서 타일맵크기*2 만큼 진행 방향으로 순간 이동 합니다.

2D 무한맵 구현

```
void OnTriggerExit2D(Collider2D collision)
{
    if (!collision.CompareTag("Area")) { return; }

    Vector3 player_posision = GameManager.instance.player.transform.positic
    Vector3 my_posision = transform.position;

    switch (transform.tag)
    {
        case "Ground":
            float diffx = player_posision.x - my_posision.x;
            float diffy = player_posision.y - my_posision.y;

            float dirx = diffx < 0 ? -1 : 1;
            float diry = diffy < 0 ? -1 : 1;
            diffx = Mathf.Abs(diffx);
            diffy = Mathf.Abs(diffy);

            if (diffx > diffy)
            {
                transform.Translate(Vector3.right * dirx * 160);
            }
            else if (diffx < diffy)
            {
                transform.Translate(Vector3.up * diry * 160);
            }
            else
            {
                transform.Translate(dirx * 160, diry * 160, 0);
            }
            break;
```

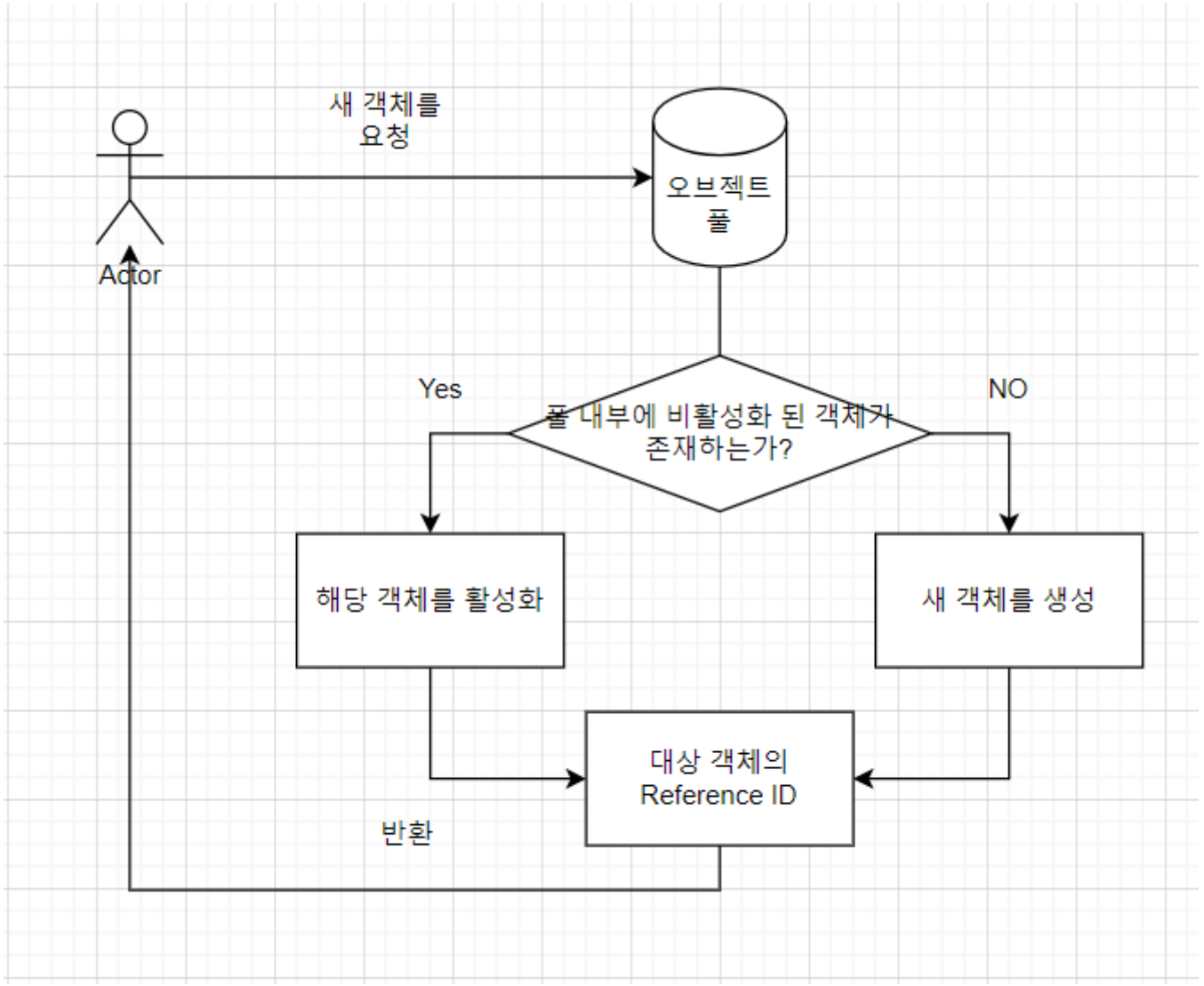
나와 겹쳐있는 무언가가 충돌에서 벗어날때

나의 태그가 Ground이면

대상과 나의 Vector를 서로 빼서
위,아래 / 왼쪽,오른쪽으로 나갔는지 알아낸후
상하, 좌우의 좌표 차이 절대값을 비교하여
상하 인지, 좌우 인지 검사 한 후에

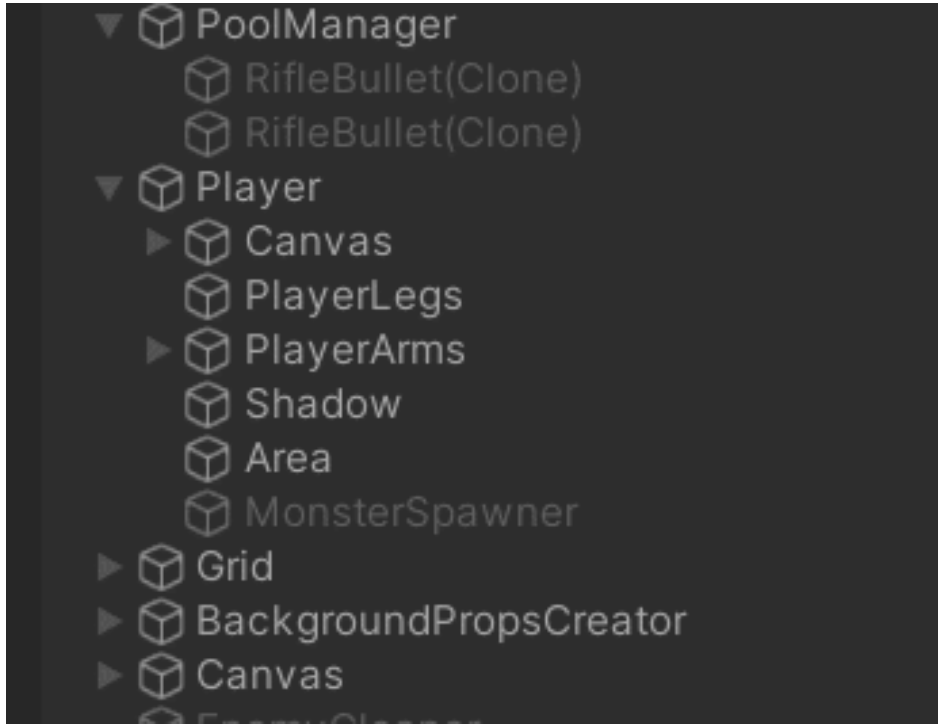
그 결과에 따라 나의 크기의 2배만큼 그 반대 방향으로 이동합니다.

Object Pooling



Object Pooling

```
public GameObject Get(int i)
{
    GameObject select = null;
    foreach(GameObject obj in pools[i])
    {
        if (!obj.activeSelf)
        {
            select = obj;
            select.SetActive(true);
            break;
        }
    }
    if (select == null)
    {
        select = Instantiate(gameObjects[i], transform);
        pools[i].Add(select);
    }

    return select;
}
```



풀에는 프리팹을 등록할 수 있게 구성되어 있고 해당 프리팹을 게임 내에 생성하기 위해선 PoolManager를 통하여 생성 요청을 해야 하도록 구현 하였습니다. 자연적으로 PoolManager는 관리할 객체들을 하위 객체로 보유하게 됩니다.

게임 통합 매니저

```
Unity 메시지 | 참조 0개
void Awake()
{
    instance = this;
    jsonLoader = GetComponent<JsonDataLoader>();
    jsonLoader.Init(language);
}
```

싱글톤 GameManager 객체를 만들고 Static으로 선언하여 항상 메모리에 올라와 있도록 표시를 합니다.

Instance에 this를 저장하여 GameManager.instance로 어떤 객체든 자유로이 관리자 객체에 접근 할 수 있도록 했습니다.

현지화가 필요한 항목 (NPC의 대사집, 아이템 설명 등)의 Text들을 GameManager가 가지고있다가
아이템을 생성하는 ItemFactory가 그 정보를 보고 아이템을 생성하게 하여

언어 설정 별로 서로 다른 정보의 텍스트가 출력되게 됩니다.

외부 Json 불러오기

```
참조 1개
public void LoadShopItemData(string language)
{
    string filePath = Application.dataPath + "/Resources/localization/ShopItemData_" + language + ".json";
    string jsonContent = File.ReadAllText(filePath);
    ItemDatabase list = JsonUtility.FromJson<ItemDatabase>(jsonContent);
    shopItemDataList = list.items;
    shopSellItemDataList = list.sellItems;
}
```

외부에 작성된 Json들을

```
참조 1개
public void Init(string language)
{
    LoadShopItemData(language);
    LoadDropItemData(language);
    LoadShopDialog(language);
    LoadWeaponLevelUpDesc(language);
    LoadPistolLevelUpData();
    LoadAk47LevelUpData();
    LoadLaserGunLevelUpData();
    LoadChainGunLevelUpData();
    LoadFireGunLevelUpData();
    LoadShotGunLevelUpData();
}
```

전부 읽어옵니다.

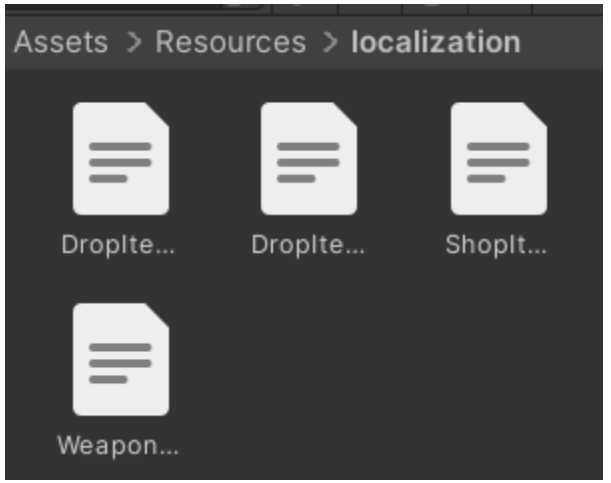
```
public JsonDataLoader jsonLoader;

Unity 메시지 | 참조 0개
void Awake()
{
    instance = this;
    jsonLoader = GetComponent<JsonDataLoader>();
    jsonLoader.Init(language);
}
```

JsonLoader 컴포넌트는 싱글톤 GameManager에 유일하게 달려있으므로

GameManaer.instance.JsonLoader로 어떠한 객체든 Json에 저장된 정보에 접근 할 수 있습니다.

현지화

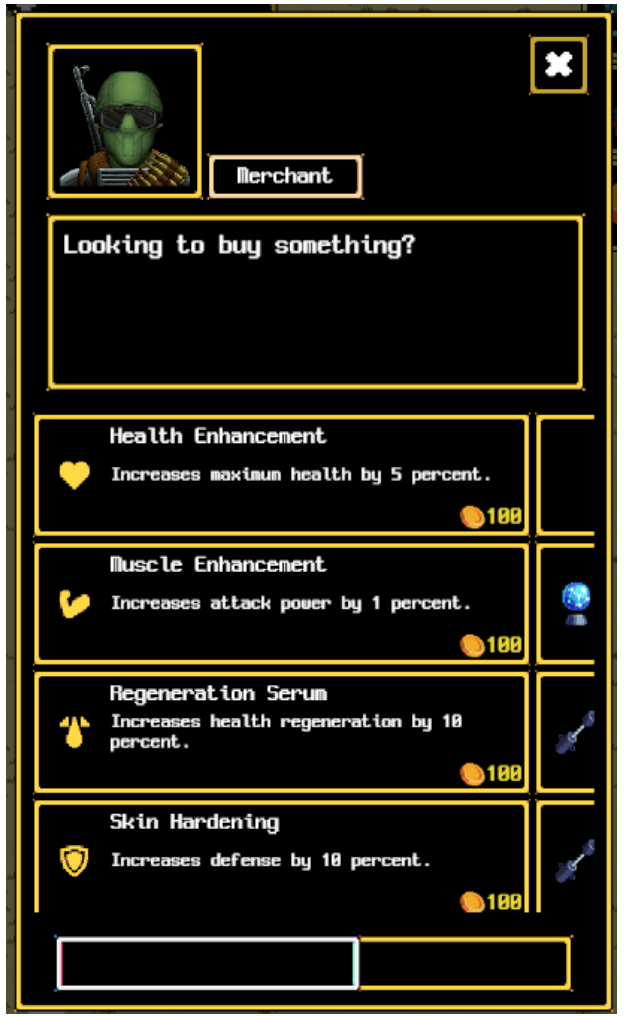
```
참조 1개
public void LoadShopItemData(string language)
{
    string filePath = Application.dataPath + "/Resources/localization/ShopItemData_" + language + ".json";
    string jsonContent = File.ReadAllText(filePath);
    ItemDatabase list = JsonUtility.FromJson<ItemDatabase>(jsonContent);
    shopItemDataList = list.items;
    shopSellItemDataList = list.sellItems;
}
```



JsonLoader는 JsonUtility.FormJson으로 Json을 읽어옵니다.
이때 인자는 해당 Json의 경로와 이름인데, 이때 인자로
DropItemData_kor.json 을 주느냐,
DropItemData_eng.json을 주느냐의 차이로 한글, 영어 설정이 달라집니다.

현지화


상인
뭐라도 살텐가?
체력 강화
최대 체력을 5% 상승시킨다.
100
근육 강화
공격력을 1% 상승시킨다.
100
재생 혈청
체력 재생을 10% 상승시킨다.
100
피부 경화
방어력을 10% 상승시킨다.
100


Merchant
Looking to buy something?
Health Enhancement
Increases maximum health by 5 percent.
100
Muscle Enhancement
Increases attack power by 1 percent.
100
Regeneration Serum
Increases health regeneration by 10 percent.
100
Skin Hardening
Increases defense by 10 percent.
100

사격을 위해 적을 찾기

```
Unity 메시지 | 참조 0개
private void FixedUpdate()
{
    targets = Physics2D.CircleCastAll(transform.position, scan_range, Vector2.zero, 0, target_layer);
    nearest_target = GetNearest();
}

참조 1개
public Transform GetNearest()
{
    Transform result = null;
    float diff = 100;

    foreach(RaycastHit2D raycastHit2D in targets)
    {
        Vector3 my_pos = transform.position;
        Vector3 target_pos = raycastHit2D.transform.position;
        float curent_diff = Vector3.Distance(my_pos, target_pos);
        if (curent_diff < diff)
        {
            diff = curent_diff;
            result = raycastHit2D.transform;
        }
    }

    return result;
}
```

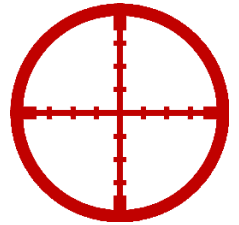
매 프레임마다 레이케스트를 쏘아 플레이어에게 가장 가까운 Transform을 찾아 반환합니다.

자동과 수동의 공통 사격점

```
if (Input.GetMouseButton(0))
{
    StartCoroutine(Shoot());
    timer = 0;
}
if (autoFire)
{
    if (player.aimPointer.aimingTarget != null)
    {
        StartCoroutine(Shoot());
        timer = 0;
    }
}
```

마우스 클릭(화면 터치)로 수동 공격을 하는 것과
자동공격에 같은 함수를 재 활용 할 수 있도록

```
참조 1개
void Fire()
{
    Vector2 target_pos = player.aimPointer.transform.position;
    Vector2 target_dir = new Vector2 (target_pos.x - transform.position.x, target_pos.y - transform.position.y);
    target_dir = target_dir.normalized;

    Transform bullet = GameManager.instance.pool.Get(bulletId).transform;
    bullet.position = firePosTransform.position;
    bullet.rotation = Quaternion.FromToRotation(Vector3.up, target_dir) * Quaternion.Euler(0, 0, 90f); ;
    bullet.GetComponent<RifleBullet>().Init(damage, count, target_dir); // -1 is INF
    firePosTransform.GetComponent<FireEffect>().StartFireEffect();
    PublishFireEvent(bulletCount);

    AudioManager.instance.PlaySfx(AudioManager.SFX.Range);
}
```



AimPointer라는 객체를 두어 해당 Transform의 위치로 발사하도록 설정

자동과 수동의 공통 사격점

```
Unity 메시지 | 참조 0개
private void FixedUpdate()
{
    if (Input.GetMouseButton(0))
    {
        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        targetPosition = mousePosition;
    }
    else
    {
        aimingTarget = scanner.nearest_target;
        if (aimingTarget != null)
        {
            targetPosition = aimingTarget.position;
        }
    }

    weaponFirePos = player.GetPlayerWeapon().transform.GetChild(0).transform.position;
    transform.position = Vector3.MoveTowards(transform.position, targetPosition, AimSpeed * Time.deltaTime);

    Vector3[] pos = { transform.position, weaponFirePos };
    lineRenderer.SetPositions(pos);
}
```

AimPointer는 매 프레임마다 마우스가 눌렸다면 마우스 위치로,
마우스가 눌리지 않았다면 가장 가까운 적 위치로 이동합니다.

사격

드랍 아이템

```
참조 1개
void DropGold()
{
    GameObject gold = GameManager.instance.pool.Get(3);
    gold.transform.position = transform.position;
    gold.GetComponent<Gold>().Init(dropGold, false);
}
```

적이 죽으며 DropGold메소드로 골드를 생성 합니다.

```
isConfirmedUnique = uniqueCheck;
//보석 인지 아닌지
int randomNumber = Random.Range(1, 101);

if (randomNumber <= 10 || isConfirmedUnique)
{
    CreateGem();
}
```

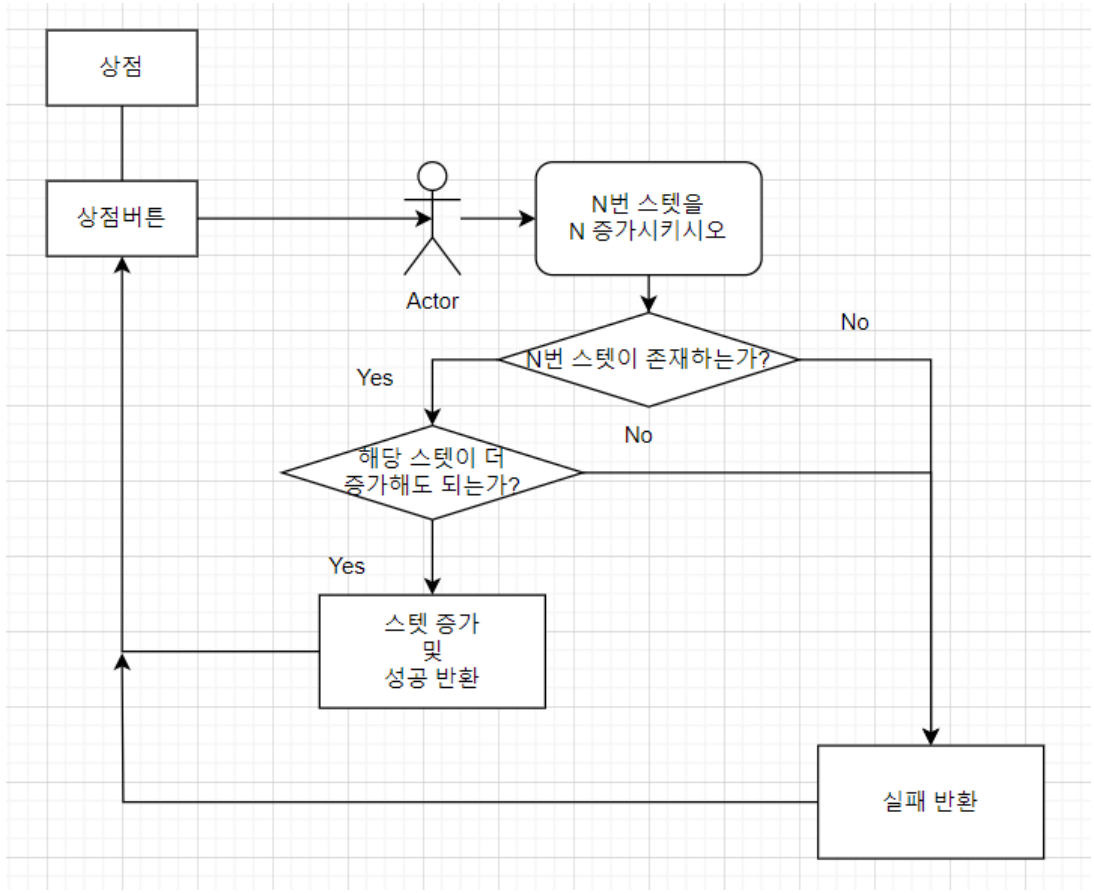
골드는 10% 확률로 아이템으로 바뀔수 있습니다.

```
참조 1개
public void CreateGem()
{
    isGem = true;
    itemId = Random.Range(0, 9);
    int randomNumber = Random.Range(1, 101);
    if (randomNumber <= 20 || isConfirmedUnique)
    {
        itemId = Random.Range(9, gemData.Count);
    }

    nameColor = gemData[itemId].nameColor;
    itemName = gemData[itemId].textName;
    spriteRenderer.sprite = gemSpriteList[itemId];

    itemInfo.text = "<color="+ nameColor + ">"+itemName+"</color>";
}
```

아이템은 20%확률로
희귀 아이템으로 바뀔수 있습니다.

isConfirmedUnique를 true로 인자로 받으면
100%확률로 희귀 아이템 (EX 보스 드랍)

골드 상점



상점은 플레이어 스텟 제어를 직접적으로 하지 않고 플레이어에게 제어 요청만 하도록 설계 하였습니다.
플레이어는 제어 요청을 받고 실패와 성공 여부만 반환합니다.

상점은 실패시 무반응, 성공시 골드 감소등의 행위를 반환 합니다.

```
참조 0개
public void OnClick()
{
    if(!isAct) { return; }
    myLevel++;
    isAct = player.ChangeStat(statId, value);
    UpToDateInfo();
}
```

무기 개조상

```
참조 0개
public void OnClick()
{
    if (weapon.level == weapon.maxLevel) {
        shop.SetDialogMessage(shop.blacksmithDialog.fail);
        return;
    }

    InventoryItem requireGem = null;
    if (weapon.level == 0)
    {
        requireGem = GameManager.instance.player.InventoryFindItem(makeGemId);
    }
    else
    {
        requireGem = GameManager.instance.player.InventoryFindItem(upradeGemId);
    }

    if (requireGem == null)
    {
        shop.SetDialogMessage(shop.blacksmithDialog.moreGold);
    }
    else
    {
        shop.SetDialogMessage(shop.blacksmithDialog.buy);
        weapon.LevelUp();
        GameManager.instance.player.InventoryRemove(requireGem.itemId, 1);
    }

    UpToDateInfo();
}
```

무기 개조 또한 무기의 능력치를 직접 변경하지 않고 무기의 LevelUp 메소드를 통해 간접 제어합니다.

02 상점


고물상
어떤 쓰레기라도 다 수거해간다고!
Reset
1 : 8
쓸모 없는 폐품
개당 10원에 일괄 판매합니다.
재활용 가능한 물자
개당 100원에 일괄 판매합니다.
사용 가능한 중고품
개당 800원에 일괄 판매합니다.
무기 개조상
재료만 줘, 돈은 받지 않는다.
FU-P2001
Lv. 1
공격 계수 3 증가
공격딜레이 0.05 감소
상인
뭐라도 살텐가?
체력 강화
최대 체력을 5% 상승시킨다.
100
근육 강화
공격력을 1% 상승시킨다.
100
재생 혈청
체력 재생을 10% 상승시킨다.
100
피부 경화
방어력을 10% 상승시킨다.
100

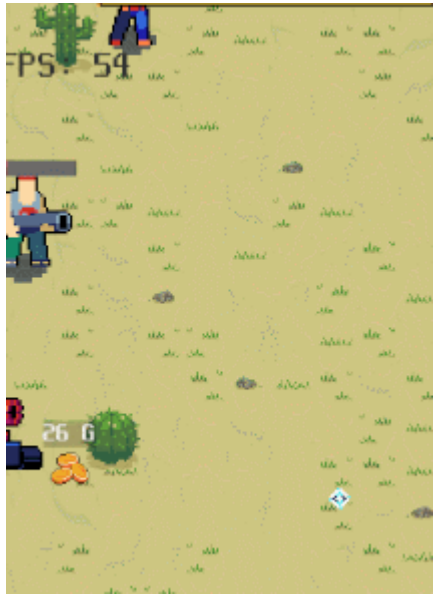
02 무기 설계 – 점사

```
참조 2개
IEnumerator Shoot()
{
    isShooting = true;

    int count = 0;
    while (count < fireCount)
    {
        if (bulletCount > 0) { Fire();  bulletCount--; }
        count++;
        yield return new WaitForSeconds(fireRate);
    }
    isShooting = false;
}

참조 1개
void Fire()
{
    Vector2 target_pos = player.aimPointer.transform.position;
    Vector2 target_dir = new Vector2 (target_pos.x - transform.position.x, target_pos.y - transform.position.y);
    target_dir = target_dir.normalized;

    Transform bullet = GameManager.instance.pool.Get(bulletId).transform;
    bullet.position = firePosTransform.position;
    bullet.rotation = Quaternion.FromToRotation(Vector3.up, target_dir) * Quaternion.Euler(0, 0, 90f); ;
    bullet.GetComponent<RifleBullet>().Init(damage * player.attackPower, count, target_dir); // -1 is INF
    firePosTransform.GetComponent<FireEffect>().StartFireEffect();
    PublishFireEvent(bulletCount);

    AudioManager.instance.PlaySfx(AudioManager.SFX.Range);
}
```

02 무기 설계 – 레이저

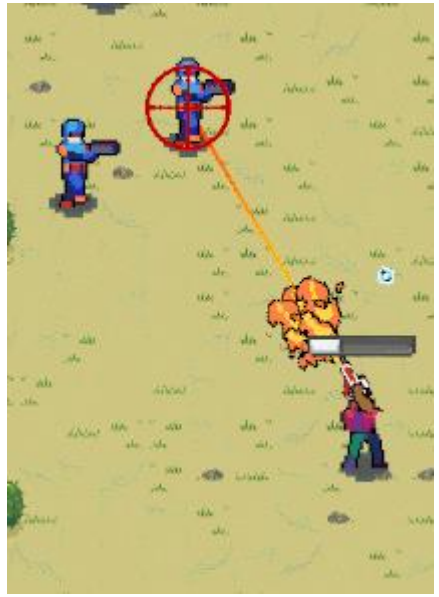
```
참조 1개
void Fire()
{
    Vector2 player_pos = firePosTransform.transform.position;
    Vector2 target_pos = player.aimPointer.transform.position;

    lineRenderer.SetPosition(0, player_pos);
    lineRenderer.SetPosition(1, target_pos);


    int layerMask = LayerMask.GetMask("Enemy");
    RaycastHit2D[] hit = Physics2D.LinecastAll(player_pos, target_pos, layerMask);
    foreach (RaycastHit2D raycastHit2D in hit)
    {
        if (raycastHit2D.transform.CompareTag("Enemy"))
        {
            raycastHit2D.transform.GetComponent<Enemy>().Hit(damage * player.attackPower);
        }
    }

}

참조 2개
IEnumerator LaserFire()
{
    isShooting = true;
    lineRenderer.enabled = true;
    PublishFireEvent(bulletCount);
    AudioManager.instance.PlaySfx(AudioManager.SFX.Range);
    float laserLife = 0;
    while(laserLife < laserWidth) {
        Fire();
        laserLife += Time.deltaTime;
        yield return new WaitForFixedUpdate();
    }
    lineRenderer.enabled = false;
    isShooting = false;
}
```

02 무기 설계 – 체인 라이트닝




```
참조 2개
void Fire()
{
    List<Transform> list = new List<Transform>();
    List<Transform> hitlist = new List<Transform>();
    list.Add(transform);
    GameObject bullet = Instantiate(myBullet, firePosTransform.transform.position, Quaternion.identity);
    bullet.GetComponent<ChainGunBullet>().Init(damage, shiftLife - 1, shiftRange, list, hitlist);

    PublishFireEvent(bulletCount);

    AudioManager.instance.PlaySfx(AudioManager.SFX.Range);
}
참조 2개
public void Init(float inputDamage, int shiftLife, float shiftRan, List<Transform> list, List<Transform> hitlist)
{
    damage = inputDamage;
    shiftRange = shiftRan;
    parentsList = list.ToList();
    hitList = hitlist.ToList();
    targets = Physics2D.CircleCastAll(transform.position, shiftRange, Vector2.zero, 0, target_layer);
    nearest_target = GetNearest();
    myParents = parentsList[parentsList.Count - 1];
    if (nearest_target != null && myParents != null)
    {
        myPostion = nearest_target.position;
        targetPostion = myParents.position;

        transform.position = myPostion;
        nearest_target.GetComponent<Enemy>().Hit(damage*GameManager.instance.player.attackPower);
        parentsList.Add(transform);
        hitList.Add(nearest_target);
        if (shiftLife > 0)
        {
            GameObject bullet = Instantiate(prefab, myPostion, Quaternion.identity);
            bullet.GetComponent<ChainGunBullet>().Init(damage, shiftLife - 1, shiftRange, parentsList, hitList);
        }
    }
    Invoke("DestroyObj", life);
}
```

02 무기 설계 – 화염방사



```
참조 2개
void Fire()
{
    float minAngle = -30f;
    float maxAngle = 30f;
    float randomAngle = Random.Range(minAngle, maxAngle);
    Vector2 target_pos = player.aimPointer.transform.position;
    Vector2 target_dir = Quaternion.Euler(0f, 0f, randomAngle) * new Vector2(target_pos.x - transform.position.x, target_pos.y - transform.position.y);
    target_dir = target_dir.normalized;

    Transform bullet = GameManager.instance.pool.Get(bulletId).transform;
    bullet.position = firePosTransform.position;
    bullet.rotation = Quaternion.FromToRotation(Vector3.up, target_dir) * Quaternion.Euler(0, 0, 90f); ;
    bullet.GetComponent<FireBullet>().Init(damage * player.attackPower, target_dir); // -1 is INF
}

참조 2개
IEnumerator FireFire()
{
    isShooting = true;
    PublishFireEvent(bulletCount);
    AudioManager.instance.PlaySfx(AudioManager.SFX.Range);
    bulletCount = fireCount;
    while (bulletCount > 0)
    {
        Fire();
        Fire();
        bulletCount--;
        yield return new WaitForFixedUpdate();
    }
    isShooting = false;
}
```

02 무기 설계 - 샷건


```
참조 2개
void Fire()
{
    float minAngle = -30f;
    float maxAngle = 30f;
    float randomAngle = Random.Range(minAngle, maxAngle);
    Vector2 target_pos = player.aimPointer.transform.position;
    Vector2 target_dir = Quaternion.Euler(0f, 0f, randomAngle) * new Vector2(target_pos.x - transform.position.x, target_pos.y - transform.position.y);
    target_dir = target_dir.normalized;


    Transform bullet = GameManager.instance.pool.Get(bulletId).transform;
    bullet.position = firePosTransform.position;
    bullet.rotation = Quaternion.FromToRotation(Vector3.up, target_dir) * Quaternion.Euler(0, 0, 90f); ;
    bullet.GetComponent<RifleBullet>().Init(damage * player.attackPower, 1, target_dir); // -1 is INF
    firePosTransform.GetComponent<FireEffect>().StartFireEffect();
}

참조 2개
IEnumerator ShotgunFire()
{
    isShooting = true;
    PublishFireEvent(bulletCount);
    AudioManager.instance.PlaySfx(AudioManager.SFX.Range);
    bulletCount = (int)bulletNumber;
    while (bulletCount > 0)
    {
        Fire();
        Fire();
        bulletCount--;
        yield return new WaitForFixedUpdate();
    }
    isShooting = false;
}
```

02 알람 시스템

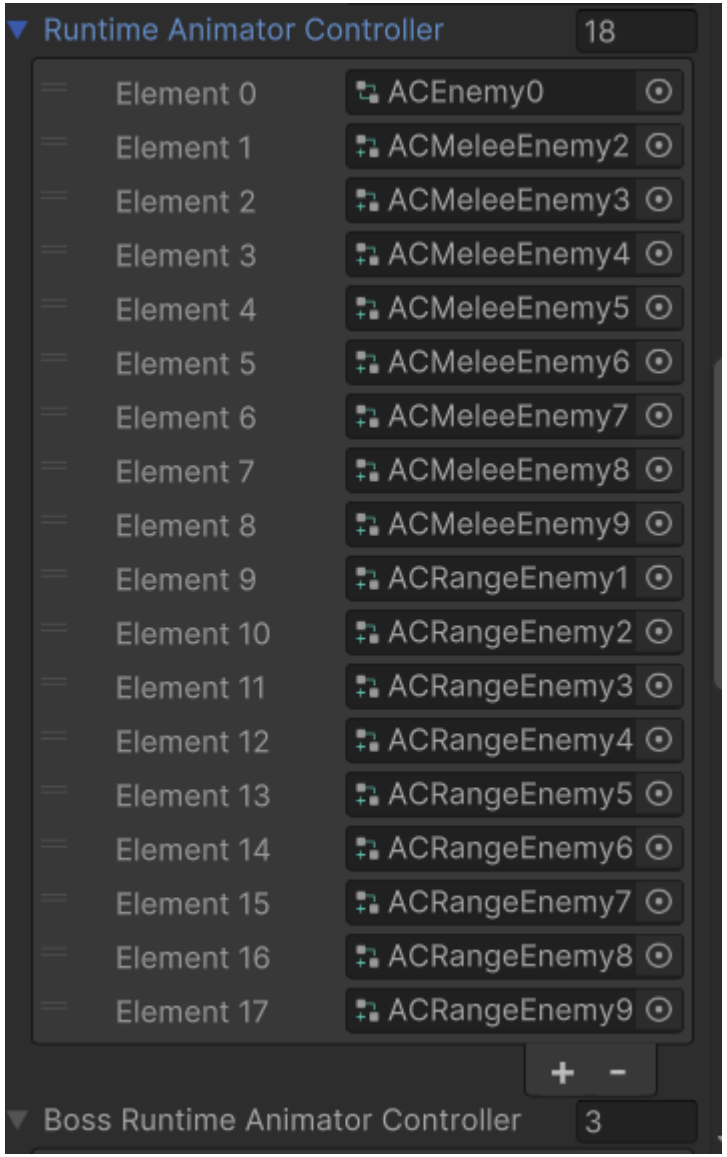

```
Unity 메시지 | 참조 0개
private void FixedUpdate()
{
    if(alertQueue.Count > 0 && !showing)
    {
        StartCoroutine(ShowAlert());
    }
}

참조 4개
public void InsertAlertQueue(string title, string content)
{
    Alert a = new Alert(title, content);
    alertQueue.Enqueue(a);
}

참조 1개
IEnumerator ShowAlert()
{
    showing = true;
    Alert a = alertQueue.Dequeue();
    titleText.text = a.title;
    contentText.text = a.content;
    while(transform.position.x < 10)
    {
        transform.position += Vector3.right * popupSpeed;
        yield return new WaitForFixedUpdate();
    }
    yield return new WaitForSeconds(3f);
    while (transform.position.x > -400)
    {
        transform.position += Vector3.left * popupSpeed;
        yield return new WaitForFixedUpdate();
    }
    showing = false;
}
```

알려야 할 내용이 있으면
전역 알람 큐를 불러 삽입 합니다.

알람은 계속해서 큐가 비었는지 확인하면서 코루틴으로 알람 메시지 표시합니다.

하나의 Enemy 프리팹을 사용해 스크립트블 오브젝트를 바꿔주는 방식으로 21가지의 몬스터를 만들어 냈습니다.

```
참조 2개
public void Init(MonsterData data)
{
    int gameLevel = GameManager.instance.gameLevel;
    monsterNumber = data.monsterNumber;
    animator.runtimeAnimatorController = runtimeAnimatorController[data.monsterNumber];
    dropGold = Random.Range(10, 15) * gameLevel;
    speed = 1 * data.speed;
    maxhp = 10 * gameLevel * data.hp;
    attackPoint = 2 * gameLevel * data.attackPoint;
    attackSpeed = 2 * data.attackSpeed;
    attackRange = 10 * data.range;

    if (isBoss)
    {
        animator.runtimeAnimatorController = bossRuntimeAnimatorController[data.monsterNumber];
        transform.localScale = Vector3.one * 4;
        maxhp *= gameLevel;
        attackPoint *= gameLevel;
    }

    hp = maxhp;
}
```

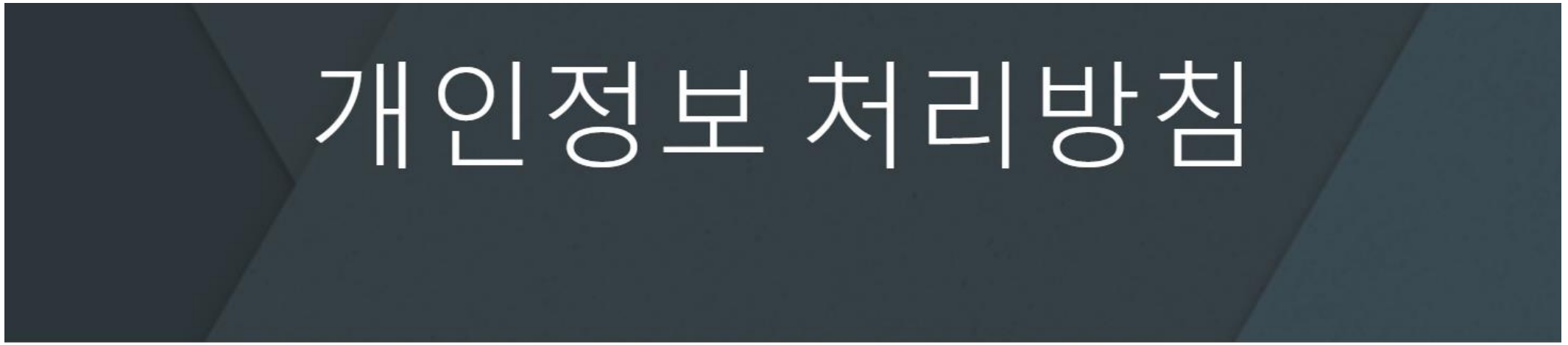


03 출시

03 등급 심사

현재 등급

IARC 상태	완료됨 세부정보 보기
이메일 주소	gkswogml997@gmail.com 수정
IARC 인증 ID	-
제출됨	2023년 11월 2일, 오후 12:29
내 등급	



개인정보 처리방침

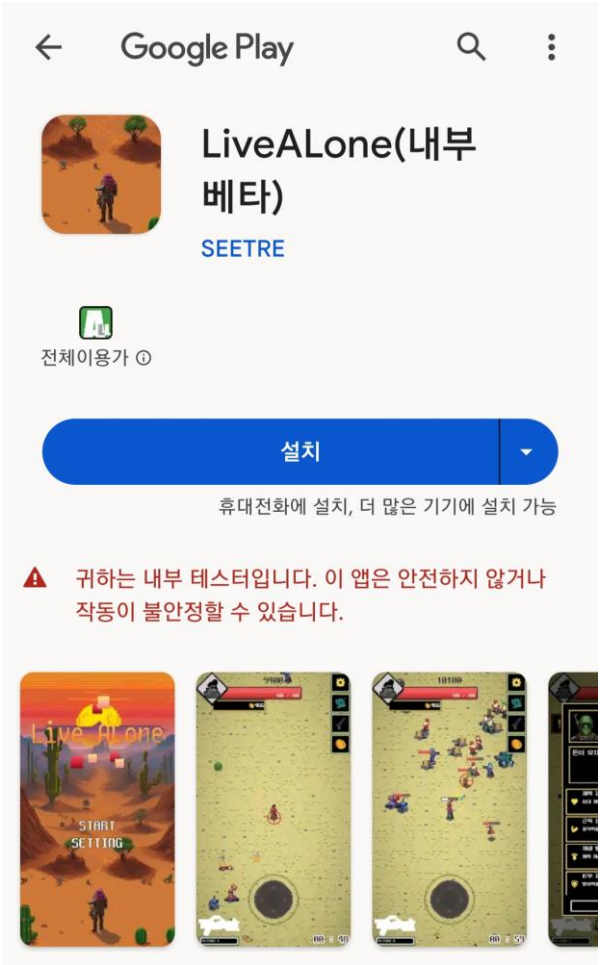
구글 플레이 스토어에 게시된 'LiveALone'은(는) 개인정보보호법 제30조에 의거하여 이용자의 개인정보 및 권익을 보호하고 이와 관련한 고충 및 불만을 신속하게 처리하기 위하여 아래와 같이 개인정보 처리방침을 수립하여 운영하고 있습니다. 본 게임은 관계법령에서 규정하고 있는 책임과 의무를 준수하고 실천하기 위해 최선의 노력을 하고 있습니다.

시행일 : 2023-09-15

목차

https://sites.google.com/view/privacy-livealone-game/%ED%99%88

03 출시



5일간의 검토 끝에 게임이 게시되었습니다.

감사합니다