UNITY
-CAHPTER7-

SOUL SEEK

목차

1. Navigation System
2. Zombie Set – Enemy Script

NAVIGATION SYSTEM

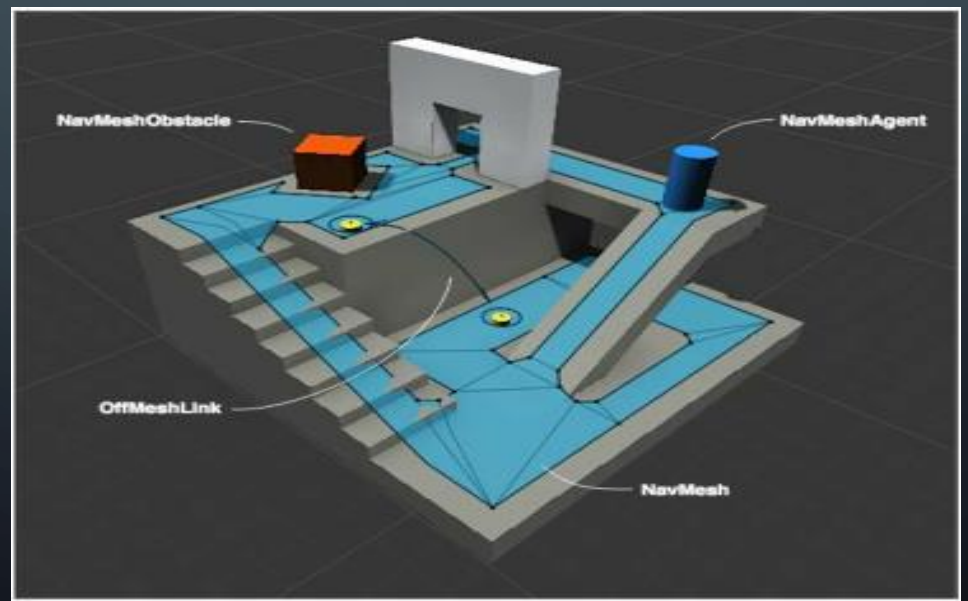
1. NAVIGATION SYSTEM

Navigation System

- **A*** 알고리즘을 기반으로 하는 **System**
- 한 위치에서 다른 위치로의 경로를 계산하고 실시간으로 장애물을 피하며 이동하는 인공지능을 만드는 시스템

Navigation System에 포함되는 Object의 종류

- **NavMesh : Agent**가 걸어 다닐 수 있는 표면**.**
- **NavMesh Agent : NaviMesh** 위에서 경로를 계산하고 이동하는 캐릭터 또는 **Component**
- **NavMesh Obstacle : Agent**의 경로를 막는 장애물
- **Off Mesh Link** 끊어진 **NaviMesh** 영역 사이를 잇는 연결 지점**(**뛰어넘을 수 있는 울타리나 타고 올라갈 수 있는 담벼락을 구현하는데 사용**)**

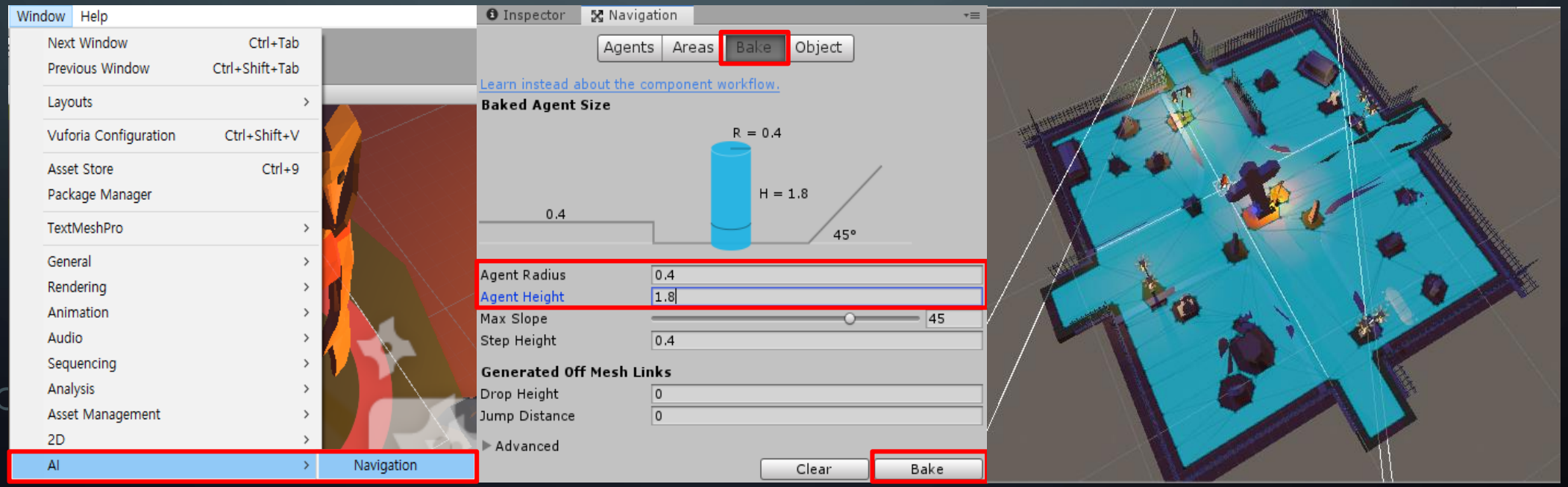


2. NAVIMESH BUILD

- **NaviMesh**는 **World**내에서 **NaviMesh Agent**가 걸어 다닐 표면이다.
  →**NaviMesh Agent**는 **NaviMesh** 위에 있는 한 점에서 다른 점으로 경로를 계산하고 이동할 수 있다.
- **NaviMesh**는 정적 **GameObject**를 대상으로 생성된다.
  →**Level Art GameObject**와 그 하위 자식 **GameObject**
  →**NaviMesh**는 **Game Play** 도중에 실시간으로 생성할 수 없기 때문에 **NaviMesh**를 미리 **Baked**해야 한다.

NaviMesh Baking

1. **Navigation Open(Window > AI > Navigation), Navigation View**에서 **Bake Tab** 클릭
2. **Agent Radius**를 **0.4 Agent Height**를 **1.8**로 변경, 버튼 **Bake** 클릭

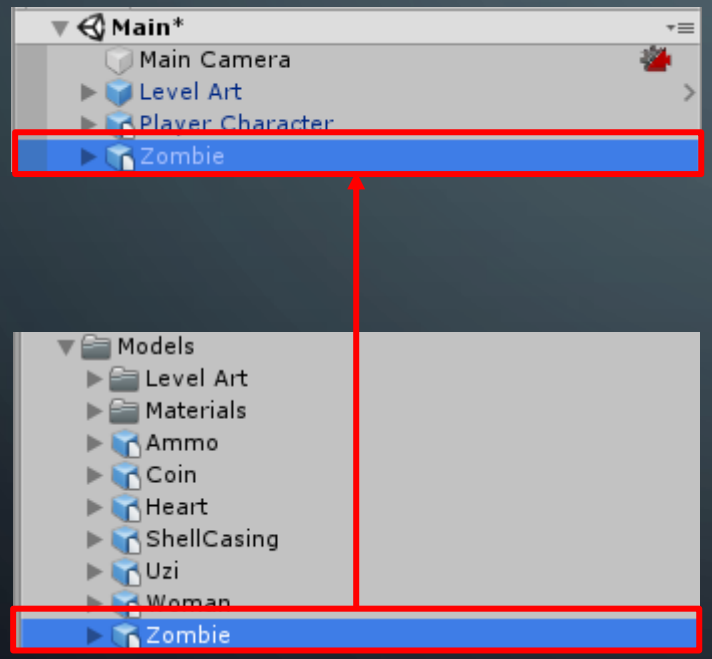


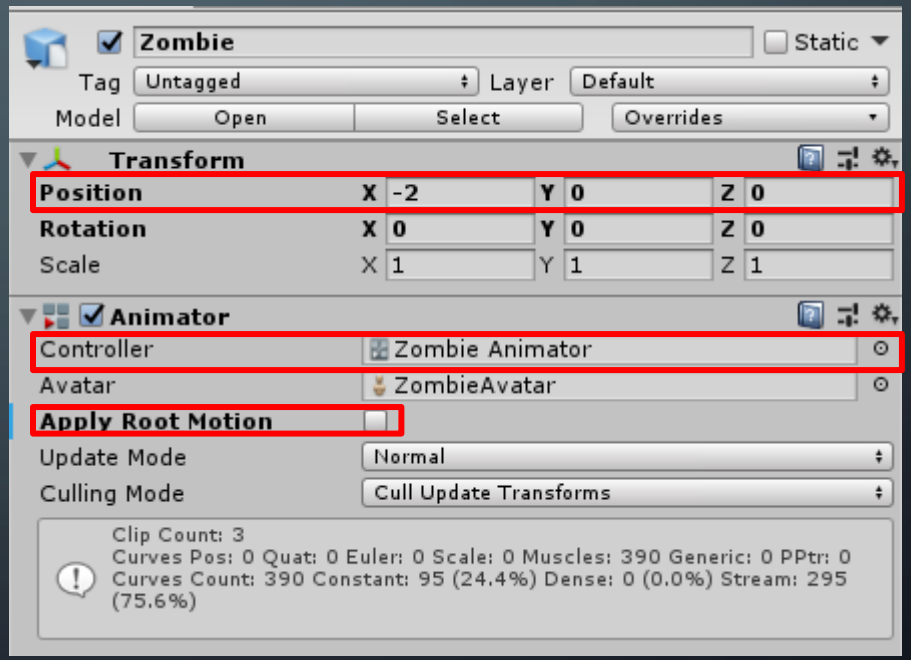
ZOMBIE SET – ENEMY SCRIPT

1. ENEMY GAMEOBJECT 준비

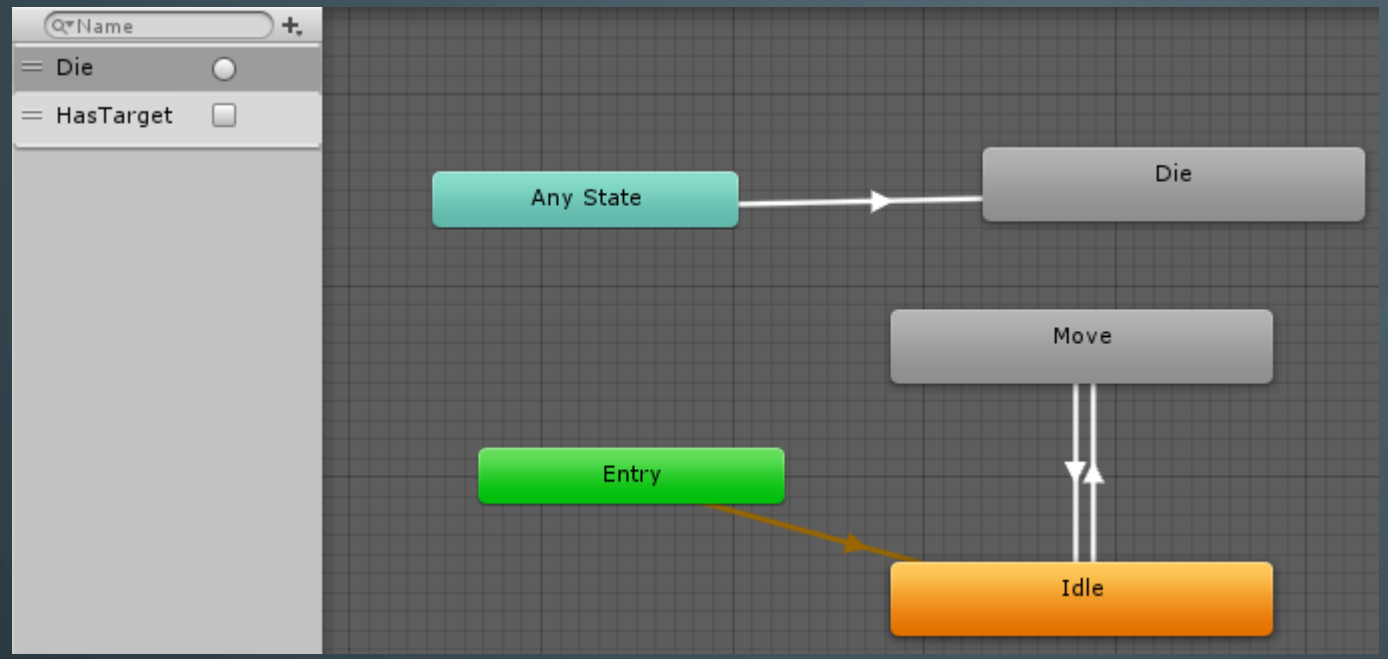
Zombie Object 추가

1. **Model Folder**에서 **Zombie Model**을 하이어라키 창으로 **Drag & Drop,** 생성된 **Zombie GameOb ject**위 위치를 **(-2, 0, 0)**으로 변경
2. **Animator Component**의 **Controller Field**에 **ZombieAnimator**를 **AnimatorContorller**에 할당 **(Controller Field** 옆에 선택버튼 클릭 **> Zombie Animator** 선택
3. **Animator Component**의 **Apply Root Motion** 체크 해제





2. ZOMBIE ANIMATOR CONTROLLER 구성



Zombie Animator의 상태값

Idle : 가만히 서 있는 **Animation Clip** 재생
Move : 뛰는 **Animation Clip** 재생
Die : 사망 **Animation Clip** 재생

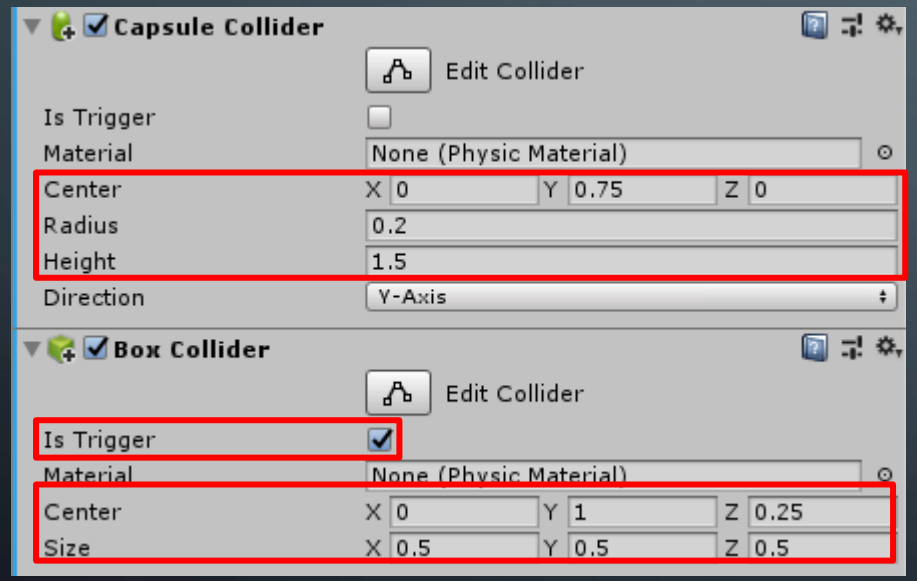
Parameta

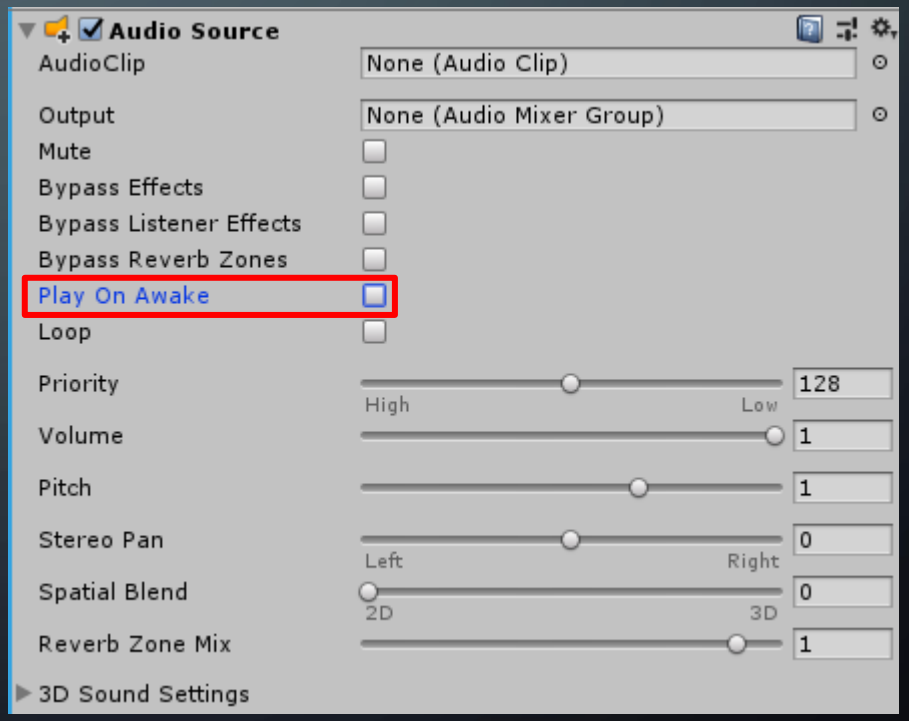
Die : Trigger Type, 사망 시 발동
Has Target : bool Type, 추적 대상이 있으면 **true,** 추적 대상이 없으면 **false**

3. ZOMBIE COMPONENT 설정

Collider & AudioSouce 추가

1. **Capsule Collider Component추가(Add Component > Physics > Capsule Collider)**
2. **Capsule Collider Component의 Center를 (0, 0.75, 0), Radius를 0.2, Height를 1.5로 변경**
3. **Box Collider Component 추가(Add Component > Physics > Box Collider), Box Collider 만 Is Trigger 체크, Center를 (0, 1, 0.25), Size를 (0.5, 0.5, 0.5)로 변경**
4. **Audio Source Component 추가(Add Componentd > Audio > Audio Source)**
5. **Audio Source Component의 Play On Awake 체크**

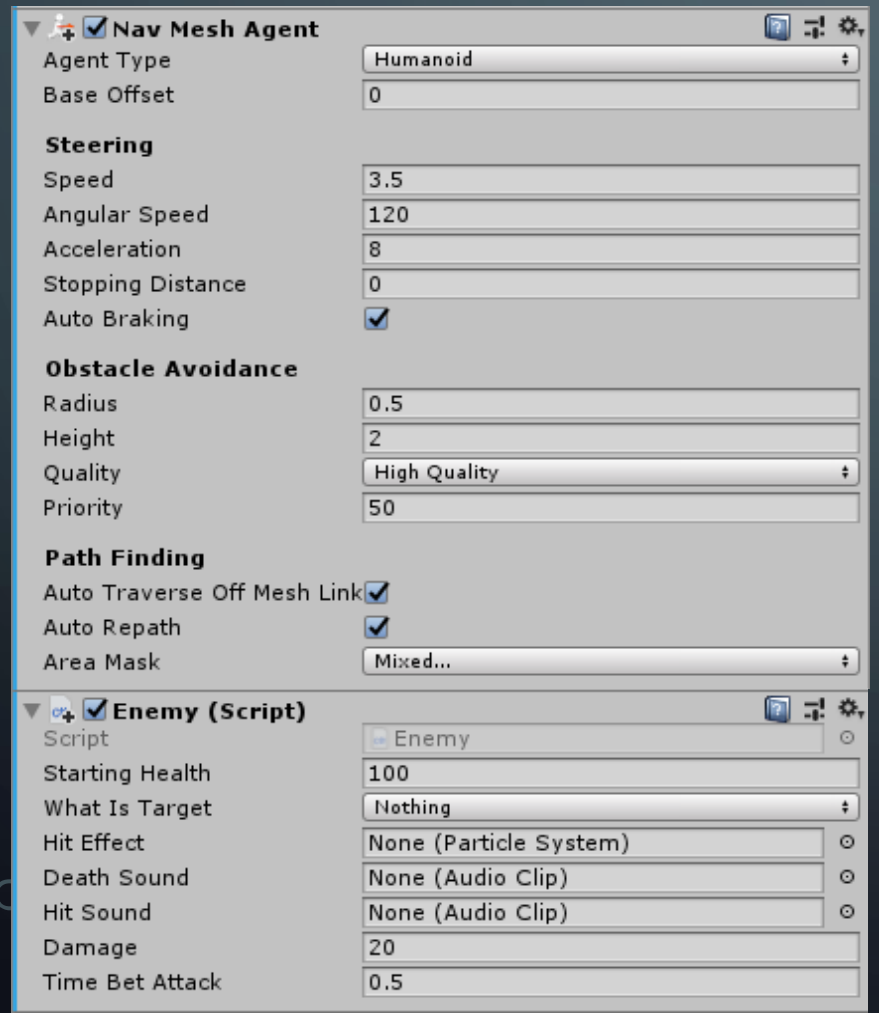


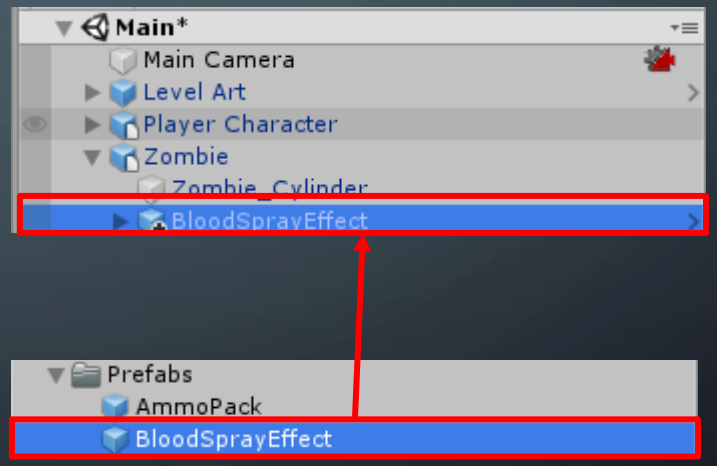


3. ENEMY SCRIPT

인공지능 추가 & 피탄 효과 추가

Nav Mesh Agent Component 추가**(Add Component > Navigation > Nav Mesh Agent)**
Enemy Script 추가
Prefabs Polder의 **BloodSprayPrefab**을 **Zombie GameObject**에 **Drag & Drop**





3. ZOMBIE COMPONENT 설정

Enemy Script의 기능들.

- **LivingEntity**에서 제공하는 기본 생명체 기능, 외부에서 **Enemy**의 초기 능력치 셋업 기능, 주기적으로 목표 위치를 찾아 경로 갱신, 공격받았을 때 피탄 효과 재생, 트리거 콜라이더를 이용해 감지된 상대방을 공 격, 사망 시 추적 중단, 사망 시 사망 효과 재생

Enemy의 Field

```
using UnityEngine.AI; // AI, 내비게이션 시스템 관련 코드를 가져오기
public LayerMask whatIsTarget;
private LivingEntity targetEntity; // 추적할 대상
private NavMeshAgent pathFinder; // 경로계산 AI 에이전트
public ParticleSystem hitEffect; // 피격시 재생할 파티클 효과
public AudioClip deathSound; // 사망시 재생할 소리
public AudioClip hitSound; // 피격시 재생할 소리
private Animator enemyAnimator; // 애니메이터 컴포넌트
private AudioSource enemyAudioPlayer; // 오디오 소스 컴포넌트
private Renderer enemyRenderer; // 렌더러 컴포넌트
public float damage = 20f; // 공격력
public float timeBetAttack = 0.5f; // 공격 간격
private float lastAttackTime; // 마지막 공격 시점
```

3. ZOMBIE COMPONENT 설정

hasTarget 프로퍼티

```
private bool hasTarget
{
    get
    {
        // 추적할 대상이 존재하고, 대상이 사망하지 않았다면 true
        if (targetEntity != null && !targetEntity.dead)
        {
            return true;
        }

        // 그렇지 않다면 false
        return false;
    }
}
```

- 추적대상이 존재하는지 알려주는 프로퍼티
- **set** 접근자가 없으므로 임의로 값을 할당할 수 없으며**,** 값을 읽는 것만 가능하다**.**
- **get** 접근자는 추적할 대상 **targetEntity**가 존재하고 **targetEntity**가 사망하지 않은 경우에만 **true**를 반환한다 → **targetEntity**가 **null**이 아니고 **targetEntity.dead**가 **false**인 경우**.**
- **hasTarget**을 받아서 **AI**가 추적할 대상이 존재하는지 알 수 있다**.**

3. ZOMBIE COMPONENT 설정

Awake() Method

- **Enemy Class**에서 가장 먼저 실행되는 **Awake() Method**는 사용할 **Component**를 찾아 변수에 할당.

```
private void Awake()
{
    //GameObject로부터 사용할 Component 가져오기
    pathFinder = GetComponent<NavMeshAgent>();
    enemyAnimator = GetComponent<Animator>();
    enemyAudioPlayer = GetComponent<AudioSource>();

    //Renderer Component 자식 GameObject에 있으므로
    //GetcomponentInChildren() Method 사용
    enemyRenderer = GetComponentInChildren<Renderer>();
}
```

- **Awake() Method**를 사용해 **NaviMesh Agent, Animator, Audio Source Component**를 **Game Object**에서 찾아온다.
- **Zombie**의 외형을 그리는 **Renderer Component**는 **Zombie GameObject**의 자식인 **Zombie_Cyli nder GameObject**에 추가되어 있다.
- 따라서 **enemyRenderer**에 할당할 **Renderer Component**는 자식 **GameObject**에서 **Component** 를 찾는 **GetComponentInChildren() Method**를 사용해 찾아온다.

3. ZOMBIE COMPONENT 설정

Setup() Method

- 공격력, 체력, 이동 속도 등 적의 능력치를 설정.
- **Enemy Script** 내부에서 직접 사용하지 않는다.
- →생성되는 적 스스로가 실행하는 **Method**가 아니고 적을 실시간으로 생성하는 **'적 생성기'**가 생성한 적의 초기 능력치를 설정하기 위해 사용하는 **Method**로 준비되어 있다.

```
// 적 AI의 초기 스펙을 결정하는 셋업 메서드
public void Setup(float newHealth, float newDamage, float newSpeed, Color skinColor)
{
    //체력 설정
    startingHealth = newHealth;
    health = newHealth;
    //공격력 설정
    damage = newDamage;
    //내비메시 에이전트의 이동 속도 설정
    pathFinder.speed = newSpeed;
    //Renderer가 사용 중인 Material의 컬러를 변경, 외형 색이 변함
    enemyRenderer.material.color = skinColor;
}
```

- 입력 받은 체력과 공격력을 그대로 적의 체력과 공격력으로 사용한다.
- 인공지능이 움직이는 속도를 조정하고 **pathFinder**에 할당된 **NaviMesh Agent Component**는 이동 속도를 나타내는 **speed Field**를 가지고 있다. **speed**의 값을 입력 받은 새로운 속도 값인 **newSpeed**의 값으로 변경한다.
- **Renderer Component**가 **3D** 모델의 외형을 그릴 때 사용하는 **Material**은 **material Field**를 사용해 접근할 수 있고 해당 **Material**의 기본 **Tint Color**는 **material.color**로 접근 가능하기 때문에 색깔 설정도 가능하다 → **'좀비 보스나 위험한 좀비는 붉은 스킨색을 가진다'** 라고 설정할 때 필요.

3. ZOMBIE COMPONENT 설정

Start() & Update()

Start() Method

- 경로 갱신을 위한 **Coroutine UpdatePath()**을 시작한다**.**

```
private void Start()
{
    // 게임 오브젝트 활성화와 동시에 AI의 추적 루틴 시작
    StartCoroutine(UpdatePath());
}
```

Update() Method

- **Animator**의 **HasTarget** 파라미터에 **hasTarget** 프로퍼티의 값을 할당하여 추적 대상의 존재 여부에 따라 알맞은 **Animation**이 재생되도록 한다**.**

```
private void Update()
{
    // 추적 대상의 존재 여부에 따라 다른 애니메이션을 재생
    enemyAnimator.SetBool("HasTarget", hasTarget);
}
```

3. ZOMBIE COMPONENT 설정

UpdatePath() Method

- 추적할 대상의 갱신된 위치를 일정주기로 파악하고, 인공지능의 목적지를 재설정하는 **Coroutine**
  →지속적인 경로 갱신을 위해 **UpdatePath()**는 **MainThread**가 진행하는 동시에 진행되어야 하기 때문 에 **Coroutine** 메서드로 구현한다.
  →**MainThread**에 영향을 미치지 않게 흘러가는 구조를 만들 수 있다. **Update, FixedUpdate** 같은 **Main Thread** 구간에서는 과도한 계산부분은 적절하지 않기 때문이며, **Update, FixedUpdate**보다 더 간격이 넓게**(1** 프레임의 간격은 엄청 짧은 시간**)** 체크할 필요가 있을 경우에는 더욱 장점을 가진다.

```
private IEnumerator UpdatePath()
{
    // 살아있는 동안 무한 루프
    while (!dead)
    {
        if(hasTarget)
        {
            //추적 대상 존재 :  경로를 갱신하고 AI 이동을 계속 진행
            pathFinder.isStopped = false;
            pathFinder.SetDestination(targetEntity.transform.position);
        }
```

- **AI** 자신이 사망하지 않은 동안 처리를 영원히 반복하여 실행하고 있고, **hasTarget**으로 추적 대상의 존재 여부를 체크하고, 존재할 경우 추적 및 이동을 계속 진행하게 된다.
- **NaviMesh Agent Component**는 이동 중단 여부를 나타내는 **isStopeed Field**와 목표 위치를 입력 받아 이동 경로를 갱신하는 **SetDestination() Method**를 가지고 있다. **isStopped**를 **false**로 변경하여 이동을 계속 진행하며, **SetDestination() Method**를 실행하고 추적 대상의 위치를 입력하여 경로를 갱신한다.

3. ZOMBIE COMPONENT 설정

hasTarget이 **false**라면 추적할 대상이 없으므로 **NaviMesh Agent Component**의 **isStopped**의 값을 **false**로 변경하여 추적과 이동을 중단한다.

```
else
{
    //추적 대상 없음 : AI 이동 중지
    pathFinder.isStopped = true;
```

자신의 위치에서 화면에 보이지 않는 반지름 **20**유닛의 가상의 구를 그리고, 구와 겹쳐 있는 모든 **Collider**를 찾는 배열로 가져온다.

```
    //20유닛의 반지름을 가진 가상의 구를 그렸을 때 구와 겹치는 모든 콜라이더를 가져옴
    //단, whatIsTarget Layer를 가진 콜라이더만 가져오도록 필터링
    Collider[] colliders = Physics.OverlapSphere(transform.position, 20f, whatIsTarget);
```

Physics.OverlapSphere() Method는 중심위치와 반지름을 입력 받아 가상의 구를 그리고, 구에 겹치는 모든 **Collider**를 반환한다. 그런데 아무런 필터링 없이 **OverlapSphere() Method**를 실행하면 성능을 낭비하게 되므로 세번째 값으로 **LayerMask**를 입력하여 특정 레이어만 감지하게 할 수 있다.
가져온 **Collider**들은 **colliders**배열에 전달 되었으며 이것을 순회하면서 해당 **Collider**를 가진 **Game Object**가 살아 있는 **LivingEntity**인지 체크한다.

```
    //모든 콜라이더를 순회하면서 살아 있는 LivingEntity 찾기
    for(int i = 0; i < colliders.Length; i++)
    {
        //콜라이더로부터 LivingEntity Component 가져오기
        LivingEntity livingEntity = colliders[i].GetComponent<LivingEntity>();

        //LivingEntity Component가 존재하며, 해당 LivingEntity가 살아 있다면
        if(livingEntity != null && !livingEntity.dead)
        {
            // 추적 대상을 해당 LivingEntity로 설정
            targetEntity = livingEntity;

            // for 문 루프 즉시 정지
            break;
        }
    }
```

만약, 살아있는 **LivingEntity**를 찾았다면 해당 **LivingEntity**를 추적 대상 **target Entity**로 삼고 **break** 루프를 빠져 나가면 된다.

3. ZOMBIE COMPONENT 설정

- **if** 또는 **else** 코드블록의 모든 처리가 끝난 다음에는 마지막으로 **while** 문의 다음 루프 회차가 실행되기 전 **yield** 문을 사용하여 **0.25**초 동안 처리를 쉰다. 즉, **while**문 내부의 경로 갱신 코드는 적 **AI**가 살이 있는 동안 **0.25**초마다 반복 실행한다.

```
// 0.25초 주기로 처리 반복
yield return new WaitForSeconds(0.25f);
```

OnDamage() Method

Damage를 적용하는 처리는 부모 클래스 **LivingEntity**의 **OnDamage() Method**에 구현된 것을 그대로 사용한다. **Enemy Script**에서는 기존 **Damage() Method**에 **Particle Effect**와 **Effect Sound** 재생을 추가한다.

```
public override void OnDamage(float damage, Vector3 hitPoint, Vector3 hitNormal)
{
    //아직 사망하지 않은 경우에만 피격 효과 재생
    if (!dead)
    {
        //공격받은 지점과 방향으로 파티클 효과 재생
        hitEffect.transform.position = hitPoint;
        hitEffect.transform.rotation = Quaternion.LookRotation(hitNormal);
        hitEffect.Play();

        //피격 효과음 재생
        enemyAudioPlayer.PlayOneShot(hitSound);
    }

    // LivingEntity의 OnDamage()를 실행하여 데미지 적용
    base.OnDamage(damage, hitPoint, hitNormal);
}
```

hitEffect.transform.Position : 공격받은 지점**(**피격 위치**)**
hitEffect.transform.rotation : 공격이 날아온 방향을 바라보는 방향**(**피격 방향**)**

3. ZOMBIE COMPONENT 설정

Die() Method

기본적인 사망 처리는 **LivingEntity**의 **Die() Method**에 구현된 것을 그대로 사용한다.

```
public override void Die()
{
    // LivingEntity의 Die()를 실행하여 기본 사망 처리 실행
    base.Die();

    //다른 AI를 방해하지 않도록 자신의 모든 콜라이터를 비활성화
    Collider[] enemyColliders = GetComponents<Collider>();
    for(int i = 0; i < enemyColliders.Length; i++)
    {
        enemyColliders[i].enabled = true;
    }

    //AI 추적을 중지하고 내비에서 컴포넌트 비활성화
    pathFinder.isStopped = true;
    pathFinder.enabled = false;

    //사망 애니메이션 재생
    enemyAnimator.SetTrigger("Die");
    //사망 효과음 재생
    enemyAudioPlayer.PlayOneShot(deathSound);
}
```

3. ZOMBIE COMPONENT 설정

Die() Method

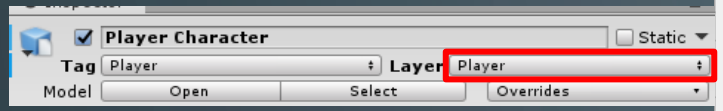
기본적인 사망 처리는 **LivingEntity**의 **Die() Method**에 구현된 것을 그대로 사용한다.

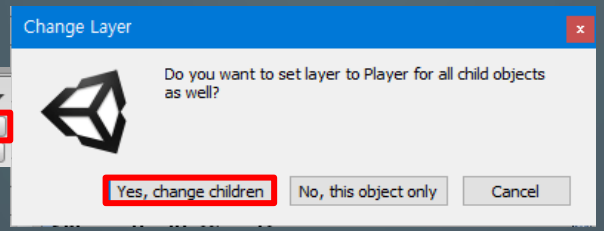
```
public override void Die()
{
    // LivingEntity의 Die()를 실행하여 기본 사망 처리 실행
    base.Die();

    //다른 AI를 방해하지 않도록 자신의 모든 콜라이터를 비활성화
    Collider[] enemyColliders = GetComponents<Collider>();
    for(int i = 0; i < enemyColliders.Length; i++)
    {
        enemyColliders[i].enabled = true;
    }

    //AI 추적을 중지하고 내비에서 컴포넌트 비활성화
    pathFinder.isStopped = true;
    pathFinder.enabled = false;

    //사망 애니메이션 재생
    enemyAnimator.SetTrigger("Die");
    //사망 효과음 재생
    enemyAudioPlayer.PlayOneShot(deathSound);
}
```

3. ZOMBIE COMPONENT 설정

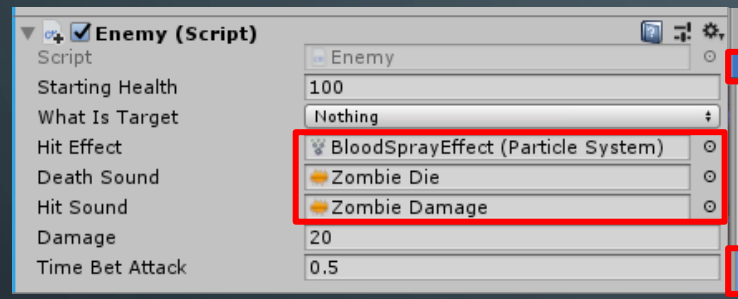
Enemy Component 설정

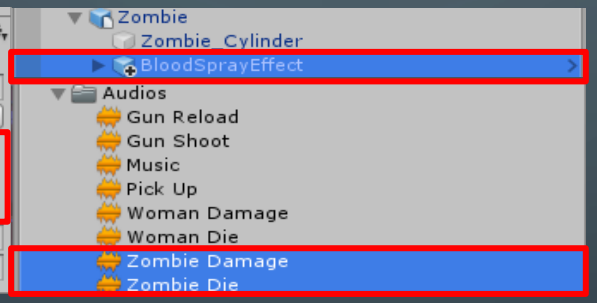
Player Layer를 Player로 변경





Enemy Public Field 설정





Zombie를 Prefab으로