Unity Zombie 下

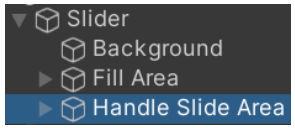
참고:레트로의 유니티 게임 프로그래밍 에센스

목차

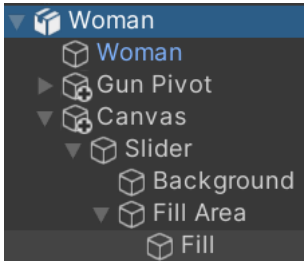
HP UI

- **Slider**
  - UI=>Canvas를 **먼저 만든 후** 그 하위에 UI=>Slider를 **만든다**

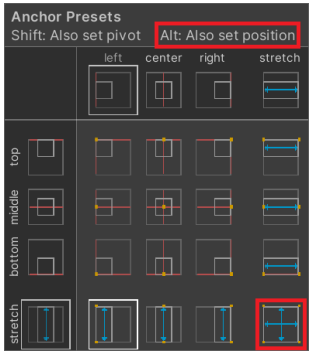
  - Slider의 **Handle Slide Area**를 **제거**

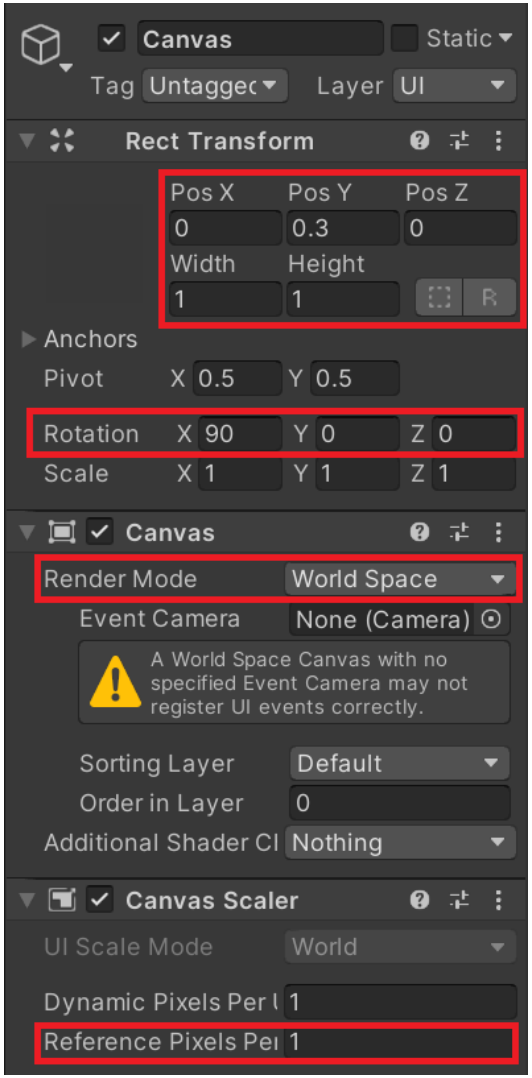


  - **Slider를 포함한 하위 자식**들을 **모두 선택**



  - **Anchor 창**에서 **Alt 키**를 **누른 상태**로 (stretch, stretch)를 선택
  - **Canvas를 Woman**의 **하위에 추가**
    - Render Mode : **World Space**
    - Reference Pixels Per Unit : **1**
    - Position : (0, 0.3, 0)
    - Width/Height : (1, 1)
    - Rotation : (90, 0, 0)

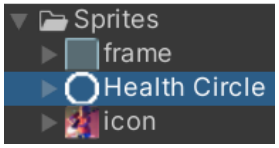


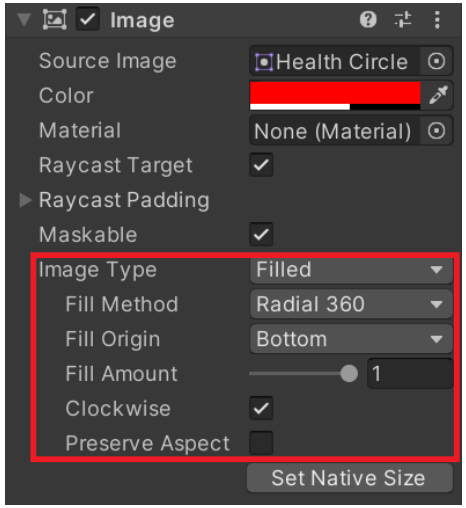


HP UI

- **Background**와 **Fill**을 **선택**

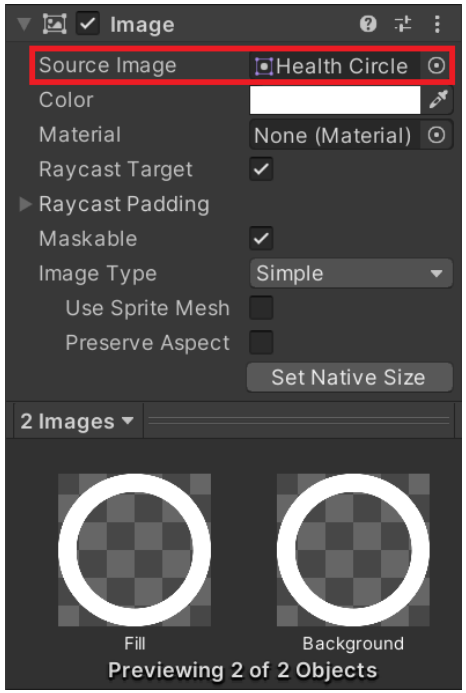
- Image의 **Source Image**를 Sprites/**Health Circle**로 **변경**

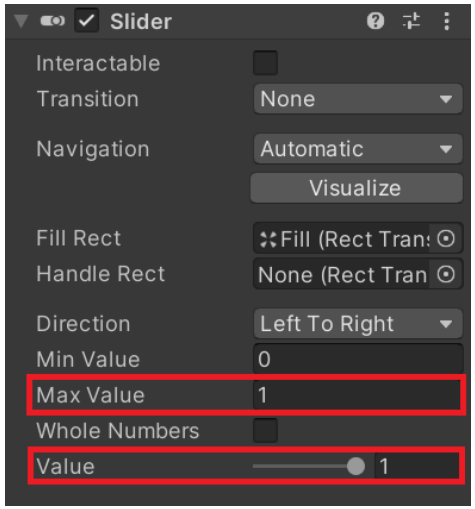




- **Background**
  - Color : (255, 255, 255, 30)

- **Fill**
  - Color : (255, 0, 0, 150)
  - Image Type : **Filled**
  - Fill Method : **Radial 360**





Damageable

- **IDamageable(Interface)**

- 데미지를 입을 수 있는 타입들이 공통적으로 가져야 하는 인터페이스

```
public interface IDamageable
{
    ///<summary> 피해량(damage), 맞은 지점(hitPoint), 맞은 표면의 법선 벡터(hitNormal)</summary>
    bool OnDamage(float damage, Vector3 hitPoint, Vector3 hitNormal);
    bool RestoreHealth(float value);
}
```

- **LivingEntity(Class)**

- 체력을 가진 오브젝트가 공통으로 가지는 Class
- 체력의 회복, 피격, 죽음 처리

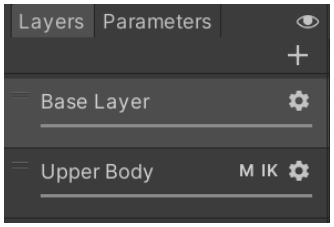
```
public abstract class LivingEntity : MonoBehaviour, IDamageable
{
    public float health { get; protected set; } = 0; // 현재 체력.
    public bool isDead => (0 >= health);             // 죽음 상태 확인.
    public event System.Action OnDamagedEvent;       // 피격 이벤트.
```

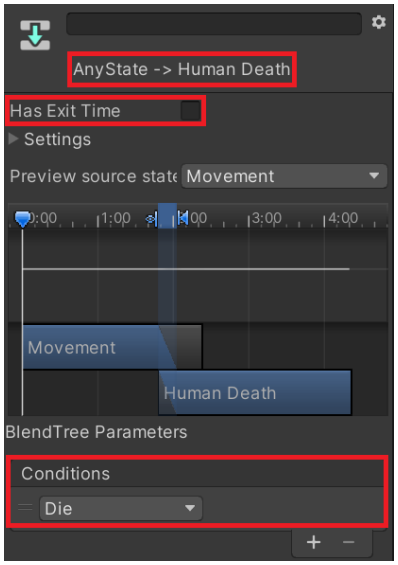
Damageable

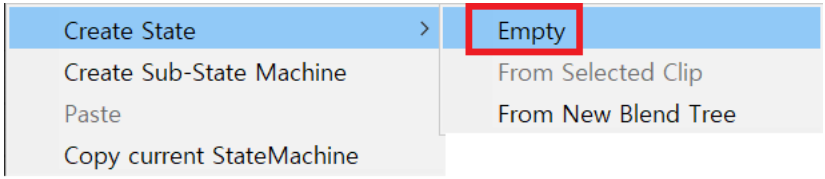
```
    public virtual bool OnDamage(float damage, Vector3 hitPoint, Vector3 hitNormal)
    {
        if (isDead) return false; // 이미 죽은 상태라면 더 이상 처리하지 않는다.

        health = Mathf.Max(health - damage, 0); // 데미지 만큼 체력 감소.
        OnDamagedEvent?.Invoke();
        return true;
    }

    public virtual bool RestoreHealth(float value)
    {
        if (isDead) return false; // 이미 죽은 상태에서는 체력을 회복할 수 없다.
        return true;
    }
} // class LivingEntity
```

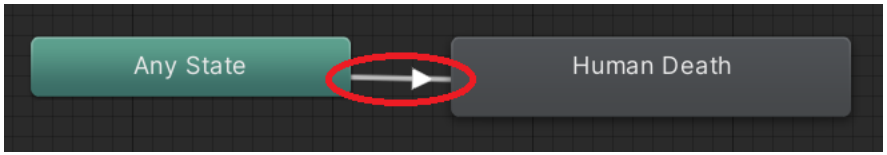
Damageable

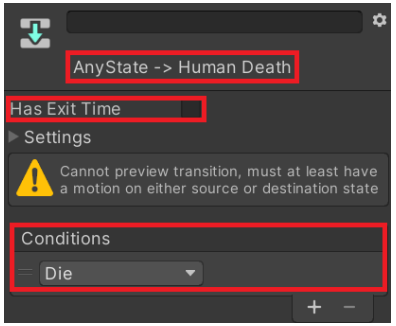
- **Animator Controller(Player)**
  - **Base Layer**
    - **Human Death** Animation Clip **추가**
    - Any State-> Human Death Has Exit Time : **false**
    - Any State->Human Death Conditions : **Die(Trigger)**
  - **Upper Layer**
    - Create State=>**Empty**(Human Death) **추가**
    - Any State->Human Death Has Exit Time : **false**
    - Any State->Human Death Conditions : **Die(Trigger)**





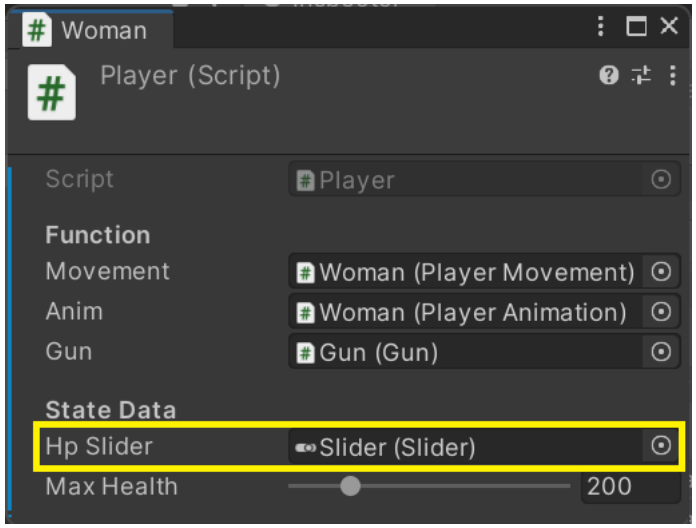






Damageable

```
public class PlayerAnimation : MonoBehaviour
{
    ...
    public void Die()
    {
        if (anim) anim.SetTrigger("Die");
    }
}

using UnityEngine.UI;
public class Player : LivingEntity // MonoBehaviour를 LivingEntity로 변경!!
{
    ...
    [Header("State Data")]
    [SerializeField] Slider hpSlider; // Inspector 창에서 Slider(GameObject) 연결.
    [SerializeField] [Range(1, 1000)] int maxHealth = 100;
    float animationHpValue = 0.0f;
    Coroutine coroutine = null;

    public event System.Action OnDieEvent = null;

    public void Initialize()
    {
        ...
        health = maxHealth;
        hpSlider.maxValue = maxHealth;
        hpSlider.value = health;
    }
```



Damageable

```
IEnumerator HpSliderAnimation()
{
    animationHpValue = hpSlider.value;
    float t = 0.0f;
    float elipsed = 1.0f / maxHealth;
    while (hpSlider.value != health)
    {
        t += elipsed;
        hpSlider.value = Mathf.Lerp(animationHpValue, health, t);

        yield return new WaitForSeconds(elipsed);
    }

    hpSlider.value = health;
    coroutine = null;
}

void OnHpAnimtion()
{
    if (null != coroutine) StopCoroutine(coroutine);
    coroutine = StartCoroutine(HpSliderAnimation());
}
```

Damageable

```
    public override bool OnDamage(float damage, Vector3 hitPoint, Vector3 hitNormal)
    {
        if (base.OnDamage(damage, hitPoint, hitNormal))
        {
            OnHpAnimtion();
            if (isDead)
            {
                movement.Stop();
                anim.Die();
                OnDieEvent?.Invoke();
            }
            return true;
        }
        return false;
    }

    public override bool RestoreHealth(float value)
    {
        if (base.RestoreHealth(value))
        {
            health = Mathf.Min(health + value, maxHealth);
            OnHpAnimtion();
            return true;
        }
        return false;
    }
} // class Player
```

Damageable

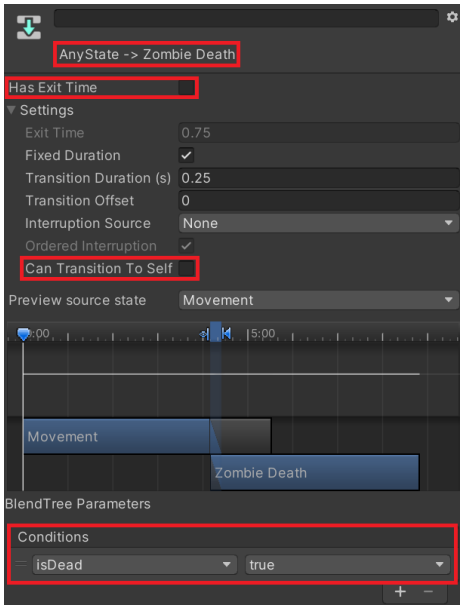
```
public sealed class GameMgr : MonoBehaviour
{
    ...
    void Initialize()
    {
        if (TryGetComponent(out input))
        {
            ...
            UIMgr.RestartEvent += StartGame;
            player.OnDieEvent += () =>
            {
                input.currentActionMap.Disable();
                UIMgr.GameOver();
            };
            ...
        }
    }
}
```

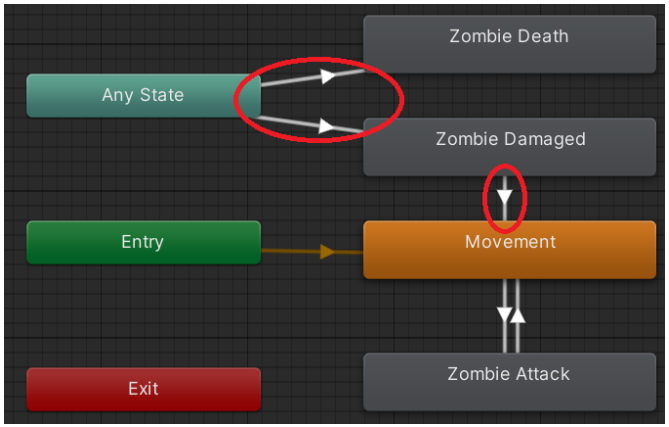
Damageable

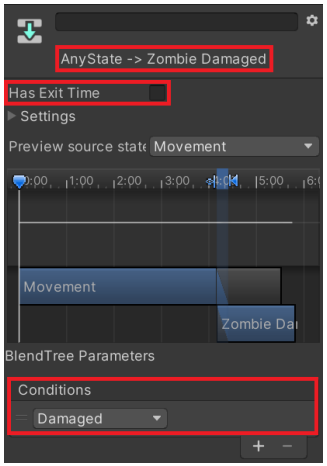
```
public class Gun : MonoBehaviour
{
    ...
    void Shot()
    {
        if (firePos)
        {
            ...
            if (Physics.Raycast(origin, dir, out RaycastHit hit, data.HitRange))
            {
                hitPos = hit.point; // 실제로 총알을 맞은 위치로 갱신.
                // Enemy(Zombie) Damage Code.
                if (hit.collider.TryGetComponent(out IDamageable target))
                {
                    target.OnDamage(data.AtkPower, hitPos, hit.normal);
                }
            }
            ...
        }
    }
    ...
}
```

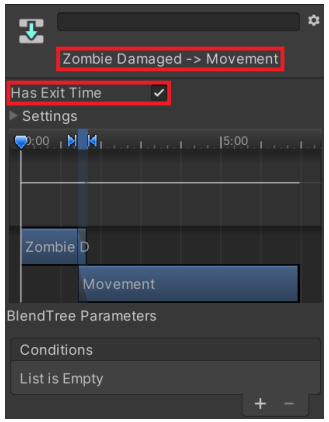
Damageable

- **Animator Controller(Zombie)**
  - **Zombie Death** Animation Clip **추가**
  - Any State->Zombie Death Conditions : **Die(Trigger)**
  - Any State->Zombie Death Has Exit Time : **false**
  - Any State->Zombie Death Setting/Can Transition To Self : **false**
  - **Zombie Damaged** Animation Clip **추가**
  - Any State->Zombie Damaged Conditions : **Damaged(Trigger)**
  - Any State->Zombie Damaged Has Exit Time : **false**
  - Zombie Damaged->Movement Has Exit Time : **true**



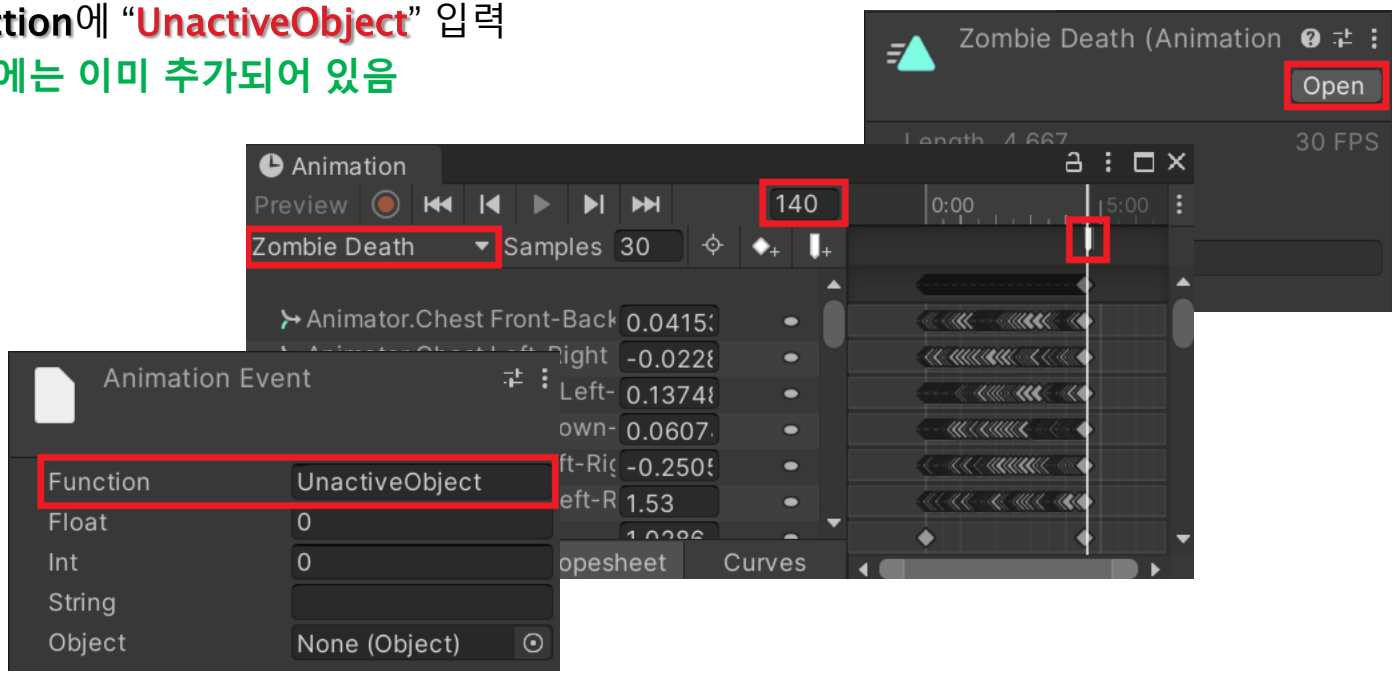






Damageable

- **Zombie Death(Animation Clip)**
  - Animation Events를 추가, Enemy Script의 UnactiveObject()를 호출
  - **Zombie Death Animation Clip**을 선택 하여 Inspector 창의 **[Open] 버튼 선택**
  - **140초** 때 **Animation Events를 추가**
  - **Function**에 "**UnactiveObject**" 입력
  - **예제에는 이미 추가되어 있음**



Damageable

```
public class Enemy : LivingEntity // MonoBehaviour를 LivingEntity로 변경!!
{
    ...
    Collider coll;
    Renderer mesh;
    Coroutine attackCoroutine = null;
    float stiffnessTimer = 0.0f;
    public bool IsStiff
    {
        get
        {
            stiffnessTimer -= updateTime;
            return (0.0f < stiffnessTimer);
        }
    }

    /// <summary> 호출 후 자동으로 delegate를 null로 비운다. </summary>
    public event System.Action<Enemy> OnUnactiveEvent = null;

    private void Awake()
    {
        ...
        coll = GetComponent<Collider>();
        mesh = GetComponentInChildren<Renderer>();
        OnDamagedEvent += OnDamagedState;
    }
    ...
```

Damageable

```
public void SetData(EnemyData enemyData)
{
    ...

    data = enemyData;
    health = data.MaxHealth;

    gameObject.SetActive(true);
    coll.enabled = true;
    mesh.material.color = Color.white;
    ...
}
...
public void ReadyAttack()
{
    float t = 1.0f - health / data.MaxHealth;                    // t : 0(white) ~ 1(red)
    mesh.material.color = Color.Lerp(Color.white, Color.red, t); // Lerp : 선형 보간.
    ...
}
```

Damageable

```
...
IEnumerator OnAttack()
{
    yield return new WaitForSeconds(data.AttackDuration * 0.5f);

    if (TargetLostCheck()) yield break;
    if (target.TryGetComponent(out IDamageable damageable))
    {
        Vector3 hitNormal = (transform.position - target.position).normalized;
        damageable.OnDamage(data.AtkPower, target.position + Vector3.up, hitNormal);
    }
}

public void AttackToTarget()
{
    ...
    // TODO : Player Damage Code.
    attackCoroutine = StartCoroutine(OnAttack());
}
```

Damageable

```
public void ReadyDamaged()
{
    if (null != attackCoroutine) StopCoroutine(attackCoroutine);
    ReadyAttack();
    stiffnessTimer = data.StiffnessDuration;
}

public void OnDamagedAnimation()
{
    if(!isDead) anim.SetTrigger("Damaged");
}

public void Die()
{
    if (null != attackCoroutine) StopCoroutine(attackCoroutine);
    MoveStop();
    coll.enabled = false;
    anim.SetTrigger("Die");
    // TODO : Add Score Code
    GameMgr.Instance.AddScore(data.GetScore);
}
```

Damageable

```
    // Zombie Death Animation Clip에서 호출 된다.
    void UnactiveObject()
    {
        gameObject.SetActive(false);

        OnUnactiveEvent?.Invoke(this);
        OnUnactiveEvent = null;
    }
} // class Enemy

public class DieState : ScriptableObject, IState
{
    public void Enter(Enemy owner)
    {
        Debug.Log("Enter Die State");
        owner.Die();
    }
    ...
}
```

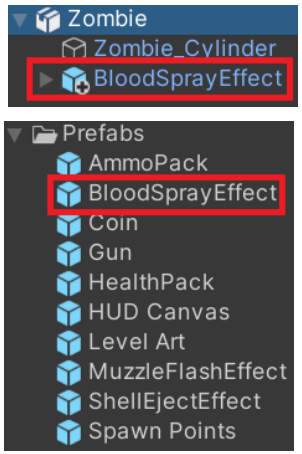
Damageable

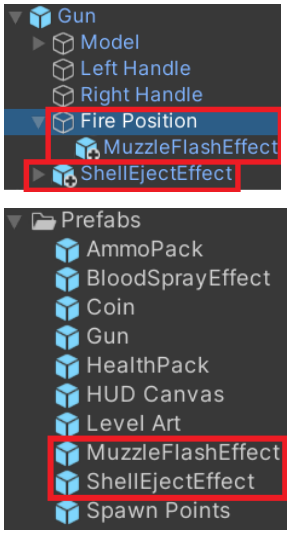
```
public class DamagedState : ScriptableObject, IState
{
    public void Enter(Enemy owner)
    {
        Debug.Log("Enter Damaged State");
        owner.ReadyDamaged();
        owner.OnDamagedAnimation();
    }

    public void Excute(Enemy owner)
    {
        if (owner.isDead)
        {
            owner.OnDieState(); // 죽었을 경우 바로 죽은 상태로 변경.
            return;
        }
        if (owner.IsStiff) return;

        owner.OnIdleState();
    }
    ...
}
```

Effects

- **Damaged Effect**
  - Zombie(GameObject)에 Prefabs/**BloodSprayEffect 추가**
  - Woman(GameObject)에 Prefabs/**BloodSprayEffect 추가**

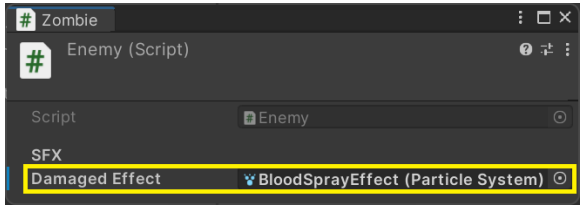
- **Gun Fire Effects**
  - Woman(GameObject)/Gun(GameObject)에 Prefabs/**ShellEjectEffect 추가**
  - Woman(GameObject)/Gun(GameObject)/Fire Position에 Prefabs/**MuzzleFlashEffect 추가**

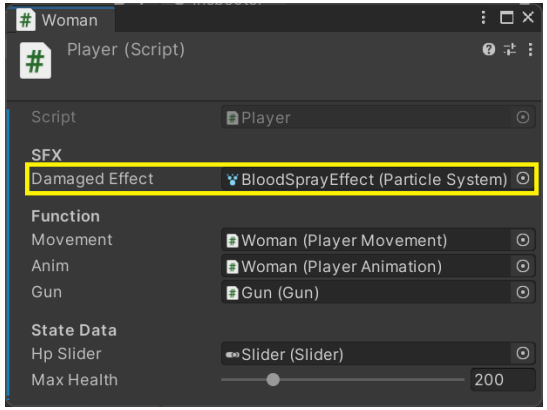




Effects

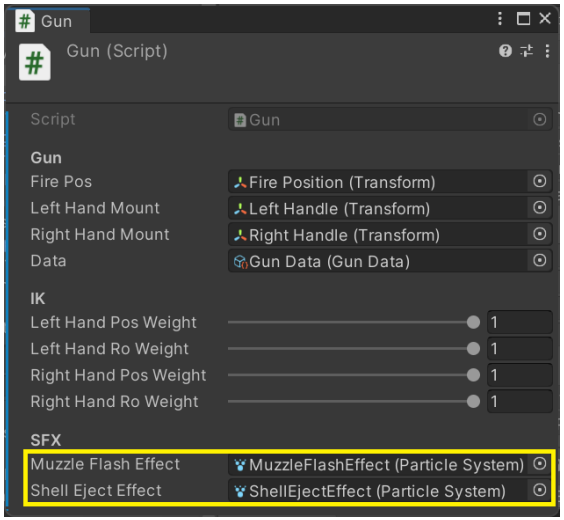
```
public abstract class LivingEntity : MonoBehaviour, IDamageable
{
    ...
    [SerializeField] ParticleSystem damagedEffect; // Inspector창에서 자신의 피격 이펙트 연결.

    public virtual bool OnDamage(float damage, Vector3 hitPoint, Vector3 hitNormal)
    {
        ...
        if (damagedEffect)
        {
            Transform effectTr = damagedEffect.transform;
            effectTr.position = hitPoint;
            effectTr.rotation = Quaternion.LookRotation(hitNormal);
            damagedEffect.Play();
        }
        return true;
    }
    ...
}
```





Effects

```
public class Gun : MonoBehaviour
{
    ...
    [Header("SFX")]
    [SerializeField] ParticleSystem muzzleFlashEffect; // 총구의 화염 효과.
    [SerializeField] ParticleSystem shellEjectEffect;  // 탄피 배출 효과.
    ...
    IEnumerator ShotEffect(Vector3 start, Vector3 end)
    {
        if (muzzleFlashEffect) muzzleFlashEffect.Play();
        if (shellEjectEffect) shellEjectEffect.Play();
        ...
    }
    ...
}
```

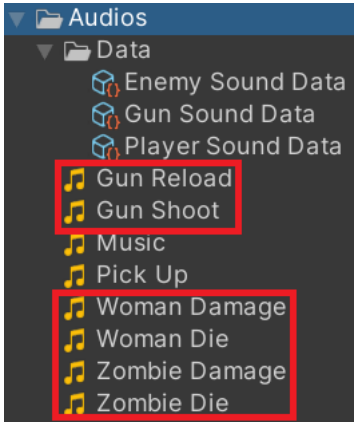


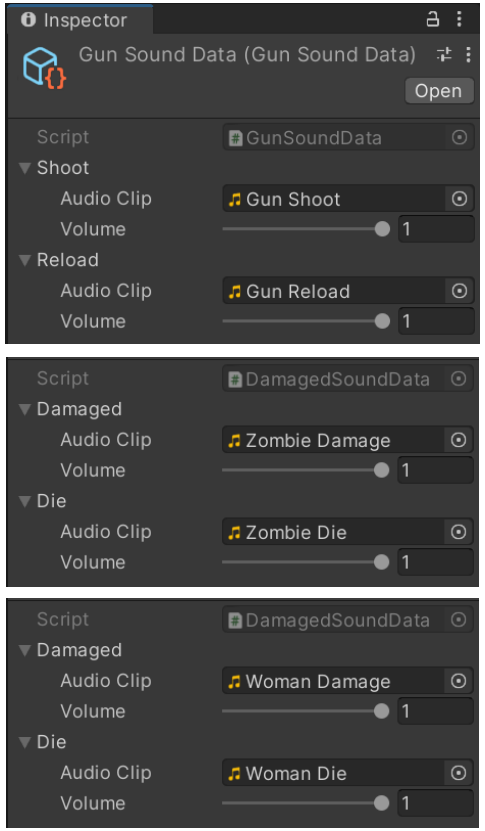
Sounds

```
[System.Serializable] // Inspector 창에 노출되기 위해 필요.
public struct SoundData
{
    public AudioClip audioClip;
    [Range(0, 1)] public float volume;

    public SoundData(float volume)
    {
        audioClip = null;
        this.volume = volume;
    }
}

[CreateAssetMenu(fileName = "Damaged Sound Data", menuName = "ScriptableObject/SFX/Damaged Sound Data", order = 1)]
public class DamagedSoundData : ScriptableObject
{
    [SerializeField] SoundData damaged;
    [SerializeField] SoundData die;

    public SoundData Damaged => damaged;
    public SoundData Die => die;

    public DamagedSoundData()
    {
        damaged = new SoundData(1.0f);
        die = new SoundData(1.0f);
    }
}
```

Sounds

```
[CreateAssetMenu(fileName = "Gun Sound Data", menuName = "ScriptableObject/SFX/Gun Sound Data", order = 2)]
public class GunSoundData : ScriptableObject
{
    [SerializeField] SoundData shoot;
    [SerializeField] SoundData reload;

    public SoundData Shoot => shoot;
    public SoundData Reload => reload;

    public GunSoundData()
    {
        shoot = new SoundData(1.0f);
        reload = new SoundData(1.0f);
    }
}
```





- **Damaged Effect Sounds**
  - Player Sound Data(DamagedSoundData), Enemy Sound Data(DamagedSoundData)
  - Damaged/Audio Clip에 Audios/(Woman/Zombie) Damaged 등록
  - Die/Audio Clip에 Audios/(Woman/Zombie) Die 등록
- **Gun Effect Sounds**
  - Gun Sound Data(Gun Sound Data)
  - Shoot/Audio Clip에 Audios/Gun Shoot 등록
  - Reload/Audio Clip에 Audios/Gun Reload 등록

Sounds

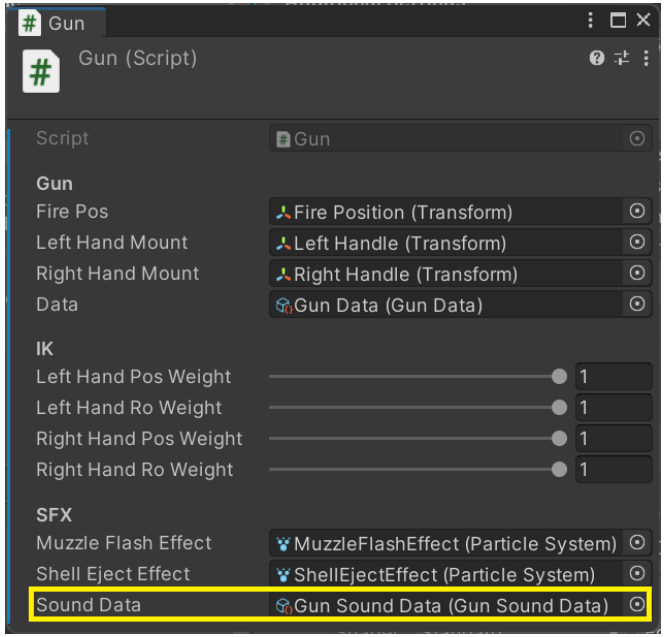
```
public abstract class LivingEntity : MonoBehaviour, IDamageable
{
    ...
    [SerializeField] protected DamagedSoundData soundData = null; // Inspector창에서 Sound Data 연결.
    protected AudioSource audioSource;

    protected void SetAudioSource()
    {
        if (!TryGetComponent(out audioSource))
        {
            audioSource = gameObject.AddComponent<AudioSource>();
        }
    }

    protected void DieSoundPlay()
    {
        if (soundData) audioSource.PlayOneShot(soundData.Die.audioClip, soundData.Die.volume);
    }

    public virtual bool OnDamage(float damage, Vector3 hitPoint, Vector3 hitNormal)
    {
        ...
        if (soundData) audioSource.PlayOneShot(soundData.Damaged.audioClip, soundData.Damaged.volume);
        return true;
    }
    ...
}
```

Sounds

```
public class Gun : MonoBehaviour
{
    ...
    [Header("SFX")]
    [SerializeField] ParticleSystem muzzleFlashEffect;
    [SerializeField] ParticleSystem shellEjectEffect;
    [SerializeField] GunSoundData soundData = null; // Inspector창에서 Sound Data 연결.
    AudioSource audioSource;
    ...
    private void Start()
    {
        ...
        if (!TryGetComponent(out audioSource))
        {
            audioSource = gameObject.AddComponent<AudioSource>();
        }
    }
    ...
    IEnumerator ShotEffect(Vector3 start, Vector3 end)
    {
        if (soundData) audioSource.PlayOneShot(soundData.Shoot.audioClip, soundData.Shoot.volume);
        ...
    }
```

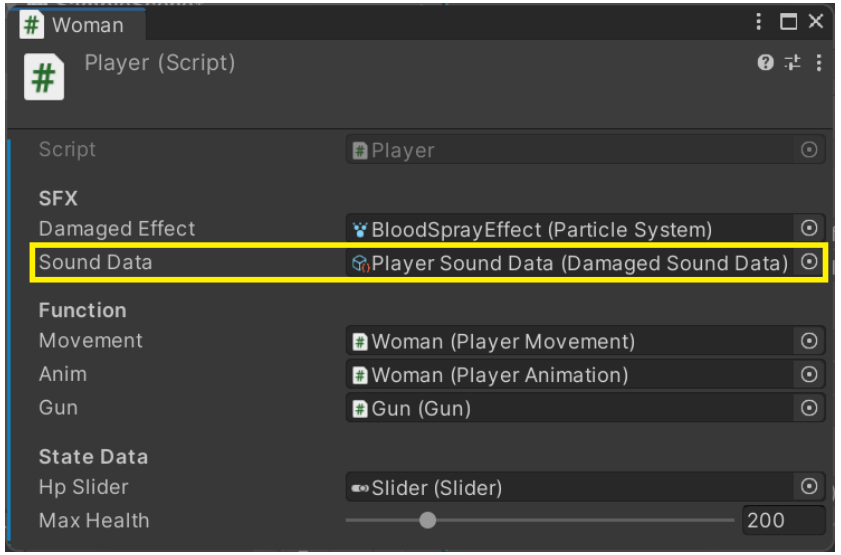
Sounds

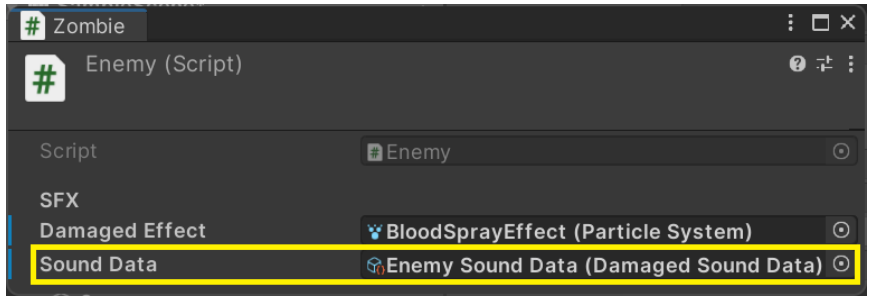
```
    ...
    IEnumerator ReloadRoutine()
    {
        state = State.Reloading;
        if (soundData) audioSource.PlayOneShot(soundData.Reload.audioClip, soundData.Reload.volume);
        ...
    }
} // class Gun

public class Player : LivingEntity
{
    ...
    private void Start()
    {
        anim.SetGun(gun);
        SetAudioSource();
        OnDieEvent += DieSoundPlay;
    }
    ...
}
```



Sounds

```
public class Enemy : LivingEntity
{
    ...
    private void Awake()
    {
        ...
        SetAudioSource();
    }
    ...
    public void Die()
    {
        ...
        DieSoundPlay();
    }
    ...
}
```





Object Pooling

- **ObjectPooling**
  - **많은 객체의 생성 후 삭제**할 때 메모리의 할당과 해제가 발생하며 **CPU에 많은 부담**을 주게된다
  - 수시로 생성 및 삭제하게 될 오브젝트를 미리 생성해두고 **비활성화**하여 숨겨 두었다가 필요할 경우 **활성화**하여 꺼내서 사용

```
public interface IPoolingObject
{
    void Initialize(object value);
    void SetPosition(Vector3 pos);
}
[System.Serializable]
public class ObjectPool<T> where T : MonoBehaviour, IPoolingObject
{
    [SerializeField] T targetObject;
    [SerializeField] [Range(1, 100)] int poolingAmount = 1;

    Transform containerObject;
    Queue<T> objectPool;
    object initData = null;
```

Object Pooling

```
public bool Initialize(object value)
{
    if (!targetObject || containerObject) return false;

    if (1 > poolingAmount) poolingAmount = 1;
    System.Text.StringBuilder sb = new System.Text.StringBuilder();
    sb.Append("Object Pool Container : ");
    sb.Append(targetObject.name);
    containerObject = new GameObject(sb.ToString()).transform;

    objectPool = new Queue<T>();
    initData = value;
    MakeAndPooling();
    return true;
}

bool MakeAndPooling()
{
    if (!containerObject) return false;

    T poolObject;
    for (int i = 0; poolingAmount > i; i++) {
        poolObject = MonoBehaviour.Instantiate(targetObject, containerObject);
        poolObject.name = targetObject.name;
        poolObject.Initialize(initData);
        poolObject.gameObject.SetActive(false);
        objectPool.Enqueue(poolObject);
    }
    return true;
}
```

Object Pooling

```
/// <summary> item 하나를 Pool에서 꺼내 활성화 시킨다. </summary>
public bool GetObject(out T item)
{
    item = null;

    if (!containerObject) return false;
    if (0 >= objectPool.Count)
    {
        if (!MakeAndPooling()) return false;
    }

    item = objectPool.Dequeue();
    item.gameObject.SetActive(true);
    return true;
}

/// <summary> 해당 풀이 가지는 아이템이 맞는지 확인. </summary>
public bool CheckItem(T item)
{
    if (!targetObject) return false;
    return targetObject.name.Equals(item.name);
}

/// <summary> item을 비활성화 시키고 Pool에 넣는다. </summary>
public bool PutInPool(T item)
{
    if (!(item && containerObject)) return false;

    item.gameObject.SetActive(false);
    objectPool.Enqueue(item);
    return true;
}
```

Object Pooling

```
    public bool Destroy()
    {
        if (!containerObject) return false;

        MonoBehaviour.Destroy(containerObject.gameObject);
        containerObject = null;

        objectPool.Clear();
        objectPool = null;
        return true;
    }

    public void ReturnBackPool()
    {
        if (containerObject)
        {
            // 모든 자식을 순회 한다.
            foreach (Transform child in containerObject)
            {
                if (child.gameObject.activeSelf)
                {
                    if (child.TryGetComponent(out T item)) PutInPool(item);
                }
            }
        }
    }
} // class ObjectPool
```

Object Pooling

```
using UnityEngine.AI;
public enum SpawnType { SELECT, RANDOM }
public class Spawner<T> : MonoBehaviour where T : MonoBehaviour, IPoolingObject
{
    [Header("Object Pool")]
    [SerializeField] ObjectPool<T>[] objectPools = null;
    [SerializeField] float maxSpawnDistance = 5.0f; // 랜덤 생성 시, 최대 반경.

    public int PoolCount => objectPools.Length;
    public int SpawnCount { get; private set; } = 0;

    protected void InitializeSpawner(object param = null)
    {
        int count = objectPools.Length;
        for (int i = 0; count > i; i++)
        {
            if (!objectPools[i].Initialize(param))
            {
                // 디버깅용 코드이기 때문에 string을 +연산자를 이용하여 합친 것, 실제 코드에서는 하면 안됨!!
                Debug.LogError(name + (i + 1) + "번째 Pool 생성 실패!!");
            }
        }
    }
```

Object Pooling

```
public T Spawn(int index, Vector3 center = default(Vector3), Vector3 offset = default(Vector3), SpawnType type = SpawnType.RANDOM)
{
    if (0 > index || (objectPools.Length - 1) < index) return null;

    if (objectPools[index].GetObject(out T item))
    {
        Vector3 pos = center;
        switch (type)
        {
            case SpawnType.SELECT: pos = (center + offset); break;
            case SpawnType.RANDOM: pos = (GetRandPositionInNeviMesh(center) + offset); break;
        }
        item.SetPosition(pos);
        SpawnCount++;
        return item;
    }
    return null;
}

Vector3 GetRandPositionInNeviMesh(Vector3 center)
{
    // center를 중심으로 반지름이 maxSpawnDistance인 구 안에서의 랜덤한 위치 하나를 지정.
    // Random.insideUnitCircle : 반지름이 1인 원 안에서의 랜덤한 위치 좌표를 반환.
    // Random.insideUnitSphere : 반지름이 1인 구체 안에서의 랜덤한 위치 좌표를 반환.
    // Random.onUnitSphere : 반지름이 1인 구체 표면상의 랜덤한 위치 좌표를 반환.
    Vector2 randomPos = Random.insideUnitCircle * maxSpawnDistance;
    // maxSpawnDistance 반경 안에서, randomPos에 가장 가까운 내비메시 위의 한 점을 찾는다.
    NavMesh.SamplePosition(center + new Vector3(randomPos.x, 0, randomPos.y), out NavMeshHit navHit, maxSpawnDistance, NavMesh.AllAreas);
    return navHit.position;
}
```

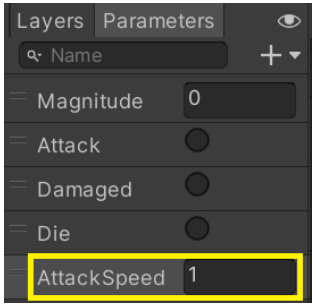
Object Pooling

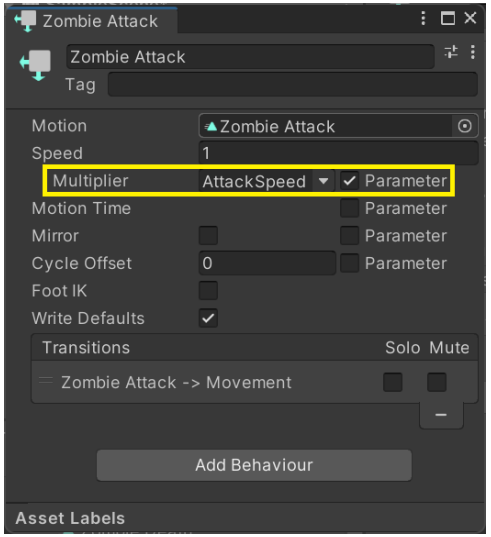
```
    public bool GiveBackItem(T item)
    {
        int count = objectPools.Length;
        for (int i = 0; count > i; i++) {
            if (objectPools[i].CheckItem(item)) {
                objectPools[i].PutInPool(item);
                SpawnCount--;
                return true;
            }
        }
        return false;
    }

    public void Clear()
    {
        int count = objectPools.Length;
        for (int i = 0; count > i; i++) {
            objectPools[i].Destroy();
        }
        SpawnCount = 0;
    }

    public void ReturnPool()
    {
        int count = objectPools.Length;
        for (int i = 0; count > i; i++) {
            objectPools[i].ReturnBackPool();
        }
        SpawnCount = 0;
    }
} // class Spawner
```

Enemies

- **Enemy 다양화**
  - Animator Controller(Zombie)의 **Parameters** AttackSpeed(float) **추가**
  - Animator Controller(Zombie)의 Zombie Attack(State)의 **Multiplier**를 AttackSpeed로 **설정**

```
public class Enemy : LivingEntity, IPoolingObject // IPoolingObject 인터페이스 추가.
{
    ...
    public void Initialize(object value) // 기존의 Initialize()를 IPoolingObject에 맞게 수정.
    {
        if (value is StateData data) { // is 연산자를 이용하여 변환 가능 여부 확인 후 값을 받는다.
            stateData = data;
        }
    }

    public bool SetData(EnemyData enemyData)
    {
        ...
        // Zombie Attack 애니메이션의 속도 제어 가능.
        anim.SetFloat("AttackSpeed", 2.3f / data.AttackDuration);
        anim.Play("Movement");
        StartCoroutine(OnUpdate());
        return true;
    }
    ...
    public void SetPosition(Vector3 pos) { transform.position = pos; }
}
```





Enemies

- **EnemyManager**
  - 많은 적 객체를 관리를 하기 위한 매니저
  - 씬(Scene)에 **EnemyManger(GameObject) 생성**
  - EnemyManger(GameObject)에 **EnemyMgr(Script) 생성 및 추가**

```
public class EnemyMgr : Spawner<Enemy>
{
    [Header("Enemy")]
    [SerializeField] EnemyData[] enemyDatas = null;
    [SerializeField] Transform centerTr; // Inspector 창에서 Level Art(GameObject) 연결.
    Vector3 center = Vector3.zero;

    public void Initialize()
    {
        Clear();

        if (centerTr) center = centerTr.position;

        IState idle = (IState)Resources.Load("EnemyData/Idle State");
        IState chase = (IState)Resources.Load("EnemyData/Chase State");
        IState attack = (IState)Resources.Load("EnemyData/Attack State");
        IState damaged = (IState)Resources.Load("EnemyData/Damaged State");
        IState die = (IState)Resources.Load("EnemyData/Die State");

        StateData data = ScriptableObject.CreateInstance<StateData>();
        data.SetData(idle, chase, attack, damaged, die);

        InitializeSpawner(data);
    }
```

Enemies

```
    public int GenerateEnemies(int count)
    {
        if (0 == count || 1 > enemyDatas.Length) return 0;

        Enemy enemy = null;
        int dataIndex = 0;
        for (int i = 0; count > i; i++)
        {
            dataIndex = Random.Range(0, enemyDatas.Length);
            enemy = Spawn(0, center);

            enemy.SetData(enemyDatas[dataIndex]);
            enemy.OnUnactiveEvent += (item) =>
            {
                if (!GiveBackItem(item)) Debug.LogError("Enemy : GiveBackItem Failed!!");
                GameMgr.Instance.UpdateGUIEnemiesCount(SpawnCount);
            };
        }
        return SpawnCount;
    }
} // class EnemyMgr
```

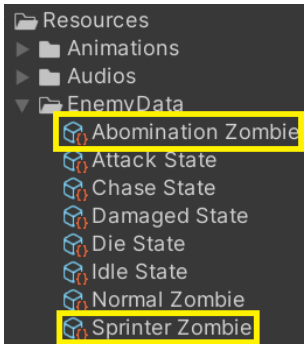
Enemies

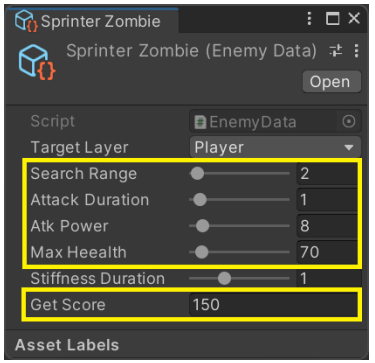
```
public sealed class GameMgr : MonoBehaviour
{
    ...
    [Header("Manager")]
    [SerializeField] GUIMgr uiMgr;
    [SerializeField] EnemyMgr enemyMgr; // Inspector 창에서 EnemyManager(GameObject) 연결.
    ...
    void StartGame()
    {
        ...
        enemyMgr.Initialize();
        UIMgr.UpdateScoreText(score);
        UpdateGUIEnemiesCount(0); // UIMgr.UpdateWaveText(wave, 0);에서 변경.
    }
    ...
    public void UpdateGUIEnemiesCount(int value)
    {
        UIMgr.UpdateWaveText(wave, value);
        if ((1 > value) && WaveUp()) {
            // TODO : Enemy Spawn Code
            int count = enemyMgr.GenerateEnemies(MaxEnemiesSpawnCount);
            UIMgr.UpdateWaveText(wave, count);
        }
    }
    ...
} // class GameMgr
```

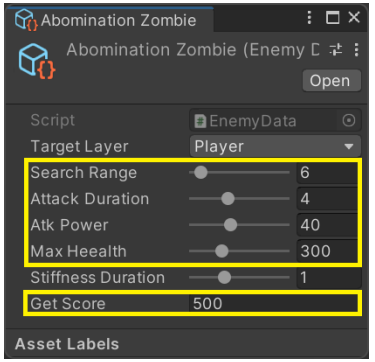
Enemies

Enemy Data

- Sprinter Zombie(Enemy Data) 생성
  - Search Range : 2
  - Attack Duration : 1
  - Atk Power : 8
  - Max Health : 70
  - Get Score : 150
- Abomination Zombie(Enemy Data) 생성
  - Search Range : 6
  - Attack Duration : 4
  - Atk Power : 40
  - Max Health : 300
  - Get Score : 500

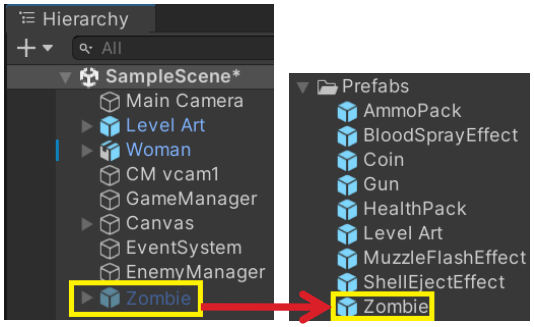






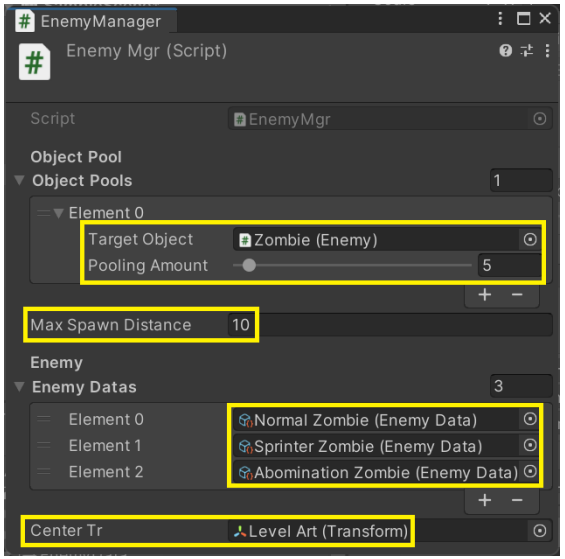
Zombie Prefab

- Hierarchy에 있는 Zombie(GameObject)를 Prefab으로 만들기
- Hierarchy에 있는 Zombie(GameObject) 삭제

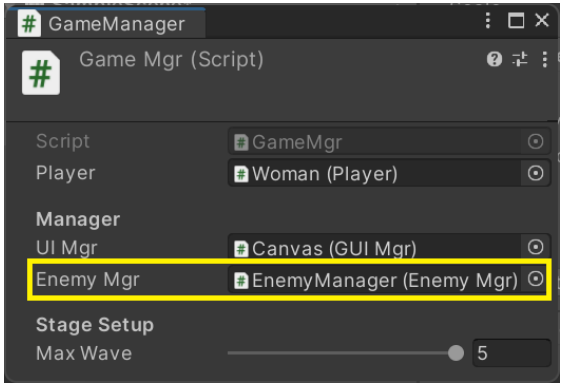


Enemies

- **Enemy Manager**
  - Object Pools : 1
    - Target Object : Resources/Prefabs/Zombie
    - Pooling Amount : 5
  - Max Spawn Distance : 10
  - Enemy Datas : 3
    - Resources/EnemyData/Normal Zombie(Enemy Data)
    - Resources/EnemyData/Sprinter Zombie(Enemy Data)
    - Resources/EnemyData/Abomination Zombie(Enemy Data)
  - Center Tr : Hierarchy에 있는 Level Art(GameObject)



- **Game Manager**
  - **Test Code 전부 삭제!!**
  - Enemy Mgr : EnemyManager(GameObject) 연결

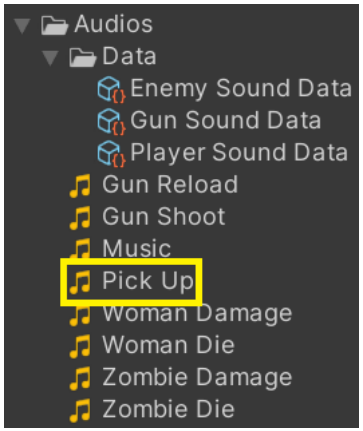


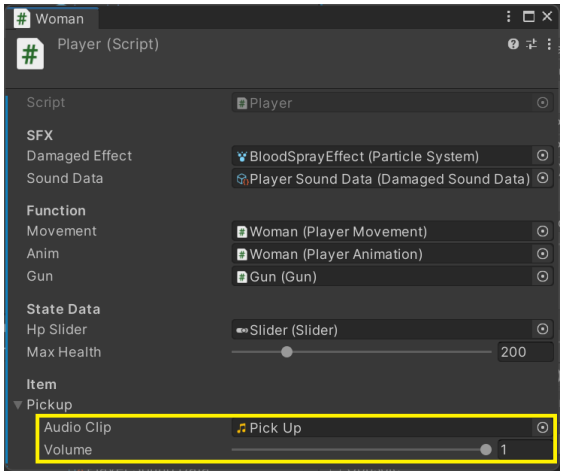
Items

- **Base Item Object class**
  - Item Object에 공통으로 적용되는 추상 클래스

```
public abstract class ItemObject : MonoBehaviour, IPoolingObject
{
    [SerializeField] LayerMask targetLayer;
    [SerializeField] protected int value;

    /// <summary> 호출 후 자동으로 delegate를 null로 비운다. </summary>
    public event System.Action<ItemObject> OnUseItemEvent = null;

    public void Initialize(object value) { }
    public void SetPosition(Vector3 pos) { transform.position = pos; }

    public virtual bool Use(GameObject target)
    {
        // *.layer는 index 값을 가지고,
        // targetLayer는 bit shift(2^index)된 값을 가진다.
        if (0 != ((1 << target.layer) & targetLayer))
        {
            gameObject.SetActive(false);
            OnUseItemEvent?.Invoke(this);
            OnUseItemEvent = null;
            return true;
        }
        return false;
    }
}
```

Items

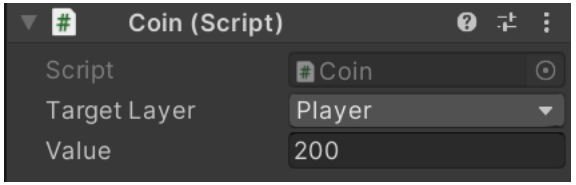





```
public class Player : LivingEntity
{
    ...
    [Header("Item")]
    [SerializeField] SoundData pickup; // Audio Clip : Resources/Audios/Pick Up(Audio Clip)
                                       // Volume : 1
    ...
    private void OnTriggerEnter(Collider other)
    {
        // 모든 Item(GameObject)들은 ItemObject 클래스를 상속 받기 때문에,
        // GetComponenet를 이용하여 ItemObject를 찾아 Use() 함수를 호출하여 사용 가능.
        if (other.TryGetComponent(out ItemObject item))
        {
            item.Use(gameObject);
            if(pickup.audioClip) audioSource.PlayOneShot(pickup.audioClip, pickup.volume);
        }
    }
}
```

Items

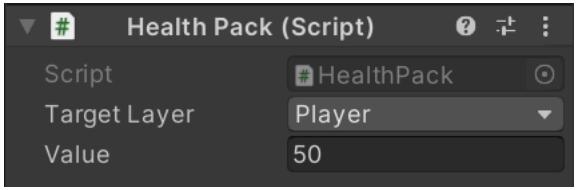
- **Coin**
  - Prefabs/**Coin(GameObject)**에 **Coin(Script)** **생성** **및** **추가**
  - Target Layer : Player
  - Value : 200




```
public class Coin : ItemObject
{
    public override bool Use(GameObject target)
    {
        if (base.Use(target))
        {
            GameMgr.Instance.AddScore(value);
            return true;
        }
        return false;
    }
}
```

Items

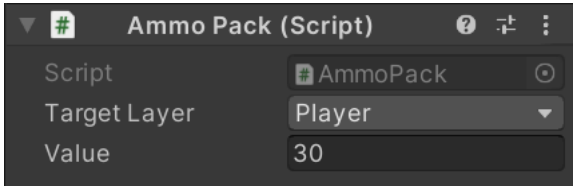
- **HealthPack**
  - Prefabs/**HealthPack(GameObject)**에 **HealthPack(Script)** 생성 및 추가
  - Target Layer : Player
  - Value : 50




```
public class HealthPack : ItemObject
{
    public override bool Use(GameObject target)
    {
        if (base.Use(target))
        {
            if (target.TryGetComponent(out IDamageable damageable))
            {
                damageable.RestoreHealth(value);
                return true;
            }
        }
        return false;
    }
}
```

Items

- **AmmoPack**
  - Prefabs/**AmmoPack(GameObject)**에 **AmmoPack(Script)** 생성 및 추가
  - Target Layer : Player
  - Value : 30




```
public class AmmoPack : ItemObject
{
    public override bool Use(GameObject target)
    {
        if (base.Use(target))
        {
            if (target.TryGetComponent(out Player player))
            {
                player.AddAmmo(value);
                return true;
            }
        }
        return false;
    }
}
```

Items

```
public class ItemSpawner : Spawner<ItemObject>
{
    [SerializeField] int maxSpawnCount = 5;
    [SerializeField] float timeBetSpawnMax = 7.0f; // 생성 최대 시간 간격.
    [SerializeField] float timeBetSpawnMin = 2.0f; // 생성 최소 시간 간격.
    float intervalTime = 0.0f;

    Transform spawnCenter;
    Coroutine coroutine = null;

    public void Initialize(Transform targetCenter)
    {
        spawnCenter = targetCenter;
        InitializeSpawner();
    }

    public bool Run()
    {
        if (spawnCenter)
        {
            Stop();
            coroutine = StartCoroutine(OnRun());
            return true;
        }
        return false;
    }
```

Items

```
    public void Stop()
    {
        if (null != coroutine)
        {
            StopCoroutine(coroutine);
            coroutine = null;
        }
    }

    IEnumerator OnRun()
    {
        while (true)
        {
            intervalTime = Random.Range(timeBetSpawnMin, timeBetSpawnMax);

            yield return new WaitForSeconds(intervalTime);

            if (maxSpawnCount > SpawnCount)
            {
                int index = Random.Range(0, PoolCount);
                ItemObject itemObject = Spawn(index, spawnCenter.position, Vector3.up * 0.5f);
                itemObject.OnUseItemEvent += (item) => GiveBackItem(item);
            }
        }
    }
} // class ItemSpawner
```

Items

```
public sealed class GameMgr : MonoBehaviour
{
    ...
    [Header("Manager")]
    [SerializeField] GUIMgr uIMgr;
    [SerializeField] EnemyMgr enemyMgr;
    [SerializeField] ItemSpawner itemSpawner; // Inspector 창에서 ItemSpawner(GameObject) 연결.
    ...
    void Initialize()
    {
        if (TryGetComponent(out input))
        {
            ...
            player.OnDieEvent += () =>
            {
                input.currentActionMap.Disable();
                UIMgr.GameOver();
                itemSpawner.Stop();
            };
```

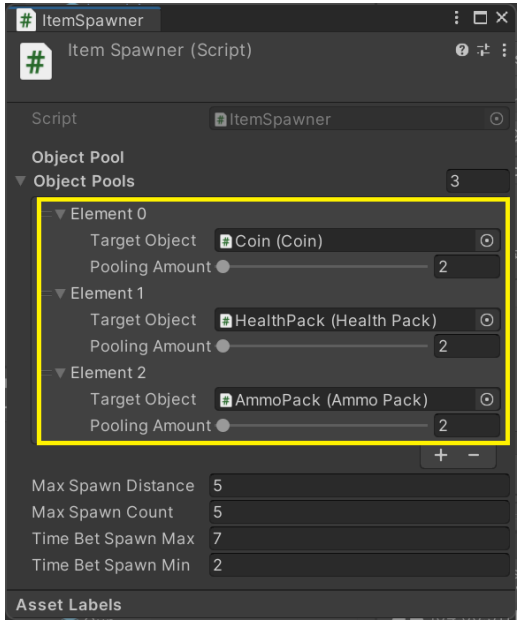
Items

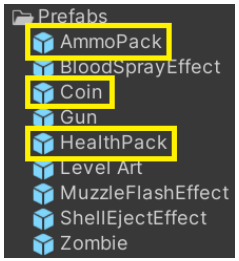
```
            itemSpawner.Initialize(player.transform);
            StartGame();
        }
    }

    void StartGame()
    {
        ...
        itemSpawner.ReturnPool();
        itemSpawner.Run();
    }
    ...
} // class GameMgr
```

Items

ItemSpawner

- 씬(Scene)에 **ItemSpawner(GameObject) 생성**
- ItemSpawner(GameObject)에 **ItemSpawner(Script) 생성 및 추가**



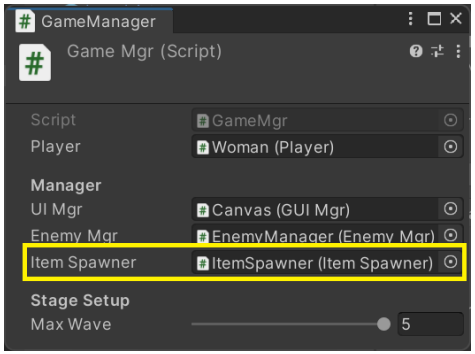


Enemy Manager

- Object Pools : 3
  - Target Object : Resources/Prefabs/Coin
  - Pooling Amount : 2
  - Target Object : Resources/Prefabs/HealthPack
  - Pooling Amount : 2
  - Target Object : Resources/Prefabs/AmmoPack
  - Pooling Amount : 2

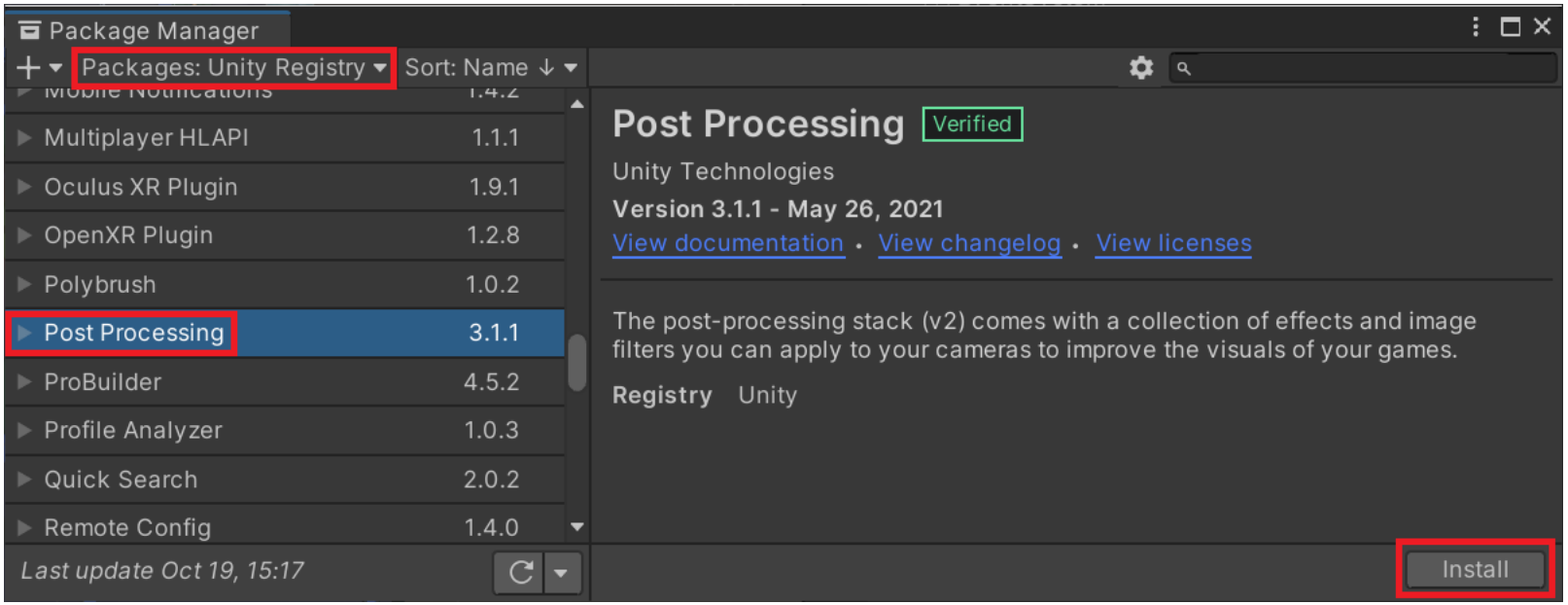
Game Manager

- Item Spawner : **ItemSpawner(GameObject) 연결**



포스트 프로세싱(Post Processing)

- **Post Processing**
  - 최종 렌더링 시에 추가 효과를 더하여 화면을 출력한다
  - Package Manager에서 Post Processing 설치

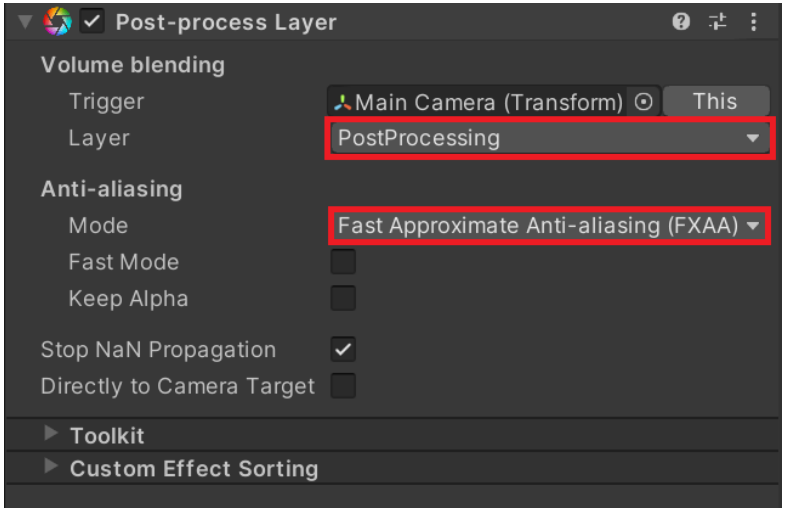


포스트 프로세싱(Post Processing)

- **Layer**
  - **PostProcessing 추가**

- **Camera**
  - **메인 카메라에 Post-process Layer Component 추가**

- **Post-process Layer**
  - Trigger : Main Camera
  - Layer : PostProcessing
  - Anti-aliasing Mode : FXAA



포스트 프로세싱(Post Processing)

- **Post-process Volume**
  - **새 오브젝트 생성**, Post-process Volume **컴포넌트 추가**
  - **Is Global** : **true**
  - **Profile** : Post-Process Profile/**Global Profile**