안녕하세요

유니티 클라이언트

최시호입니다.

최시호 I ChoiSiHo
tlgh10@naver.com
010-6778-4051

Defense

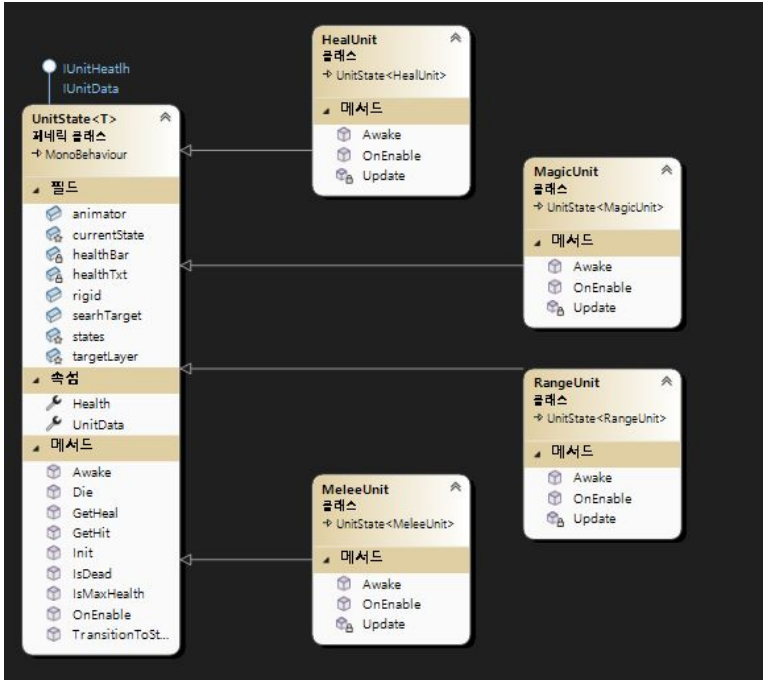
플랫폼	엔진	그래픽/시점	장르	개발 기간
PC	유니티	2D	Defense	10주

게임 설명	- 웨이브를 클리어하며 아군유닛 강화
프로그래밍	- 1인
주요 스크립트	- UnitState, BaseState - AttackStrategy - Factory, HitScan - PoolManager, GameManager - Castle - BaseMenu - WaveDataSO,UnitData - GenerateWaveDataSO - UnitSpawn, EnemySpawn
깃 허브 주소	https://github.com/SIHO0903/DefenseGame
유튜브 링크	https://youtu.be/fjwn0BxfbkA

영상

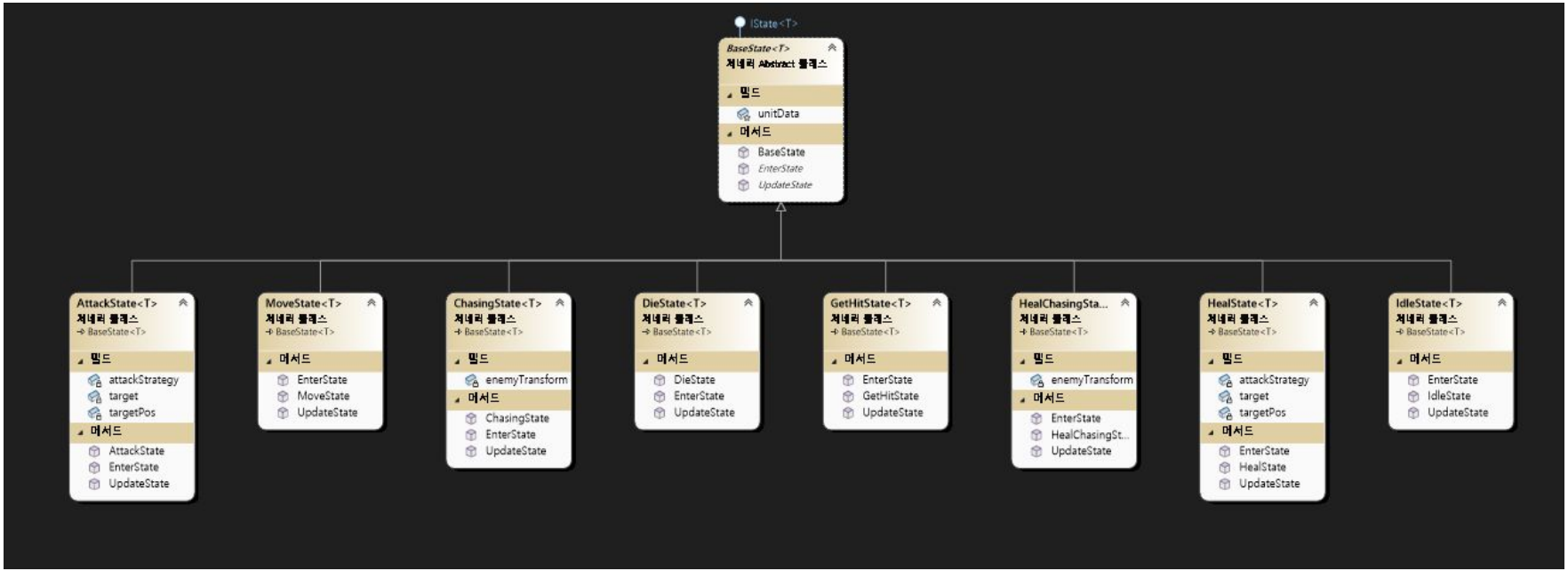

화살은 PoolManager클래스에 있는 프리팹을 사용해 만들고
화살오브젝트에 있는 Arrow클래스에서 어디로 발사되는지 결정합니다.
1200 / 1300

UnitState



UnitState<T>가 기본 유닛의 상태 및 메소드을 정의하고, 하위클래스는 그 틀을 상속받아 근접 유닛의 특정 상태 및 동작을 구현하는 관계입니다.

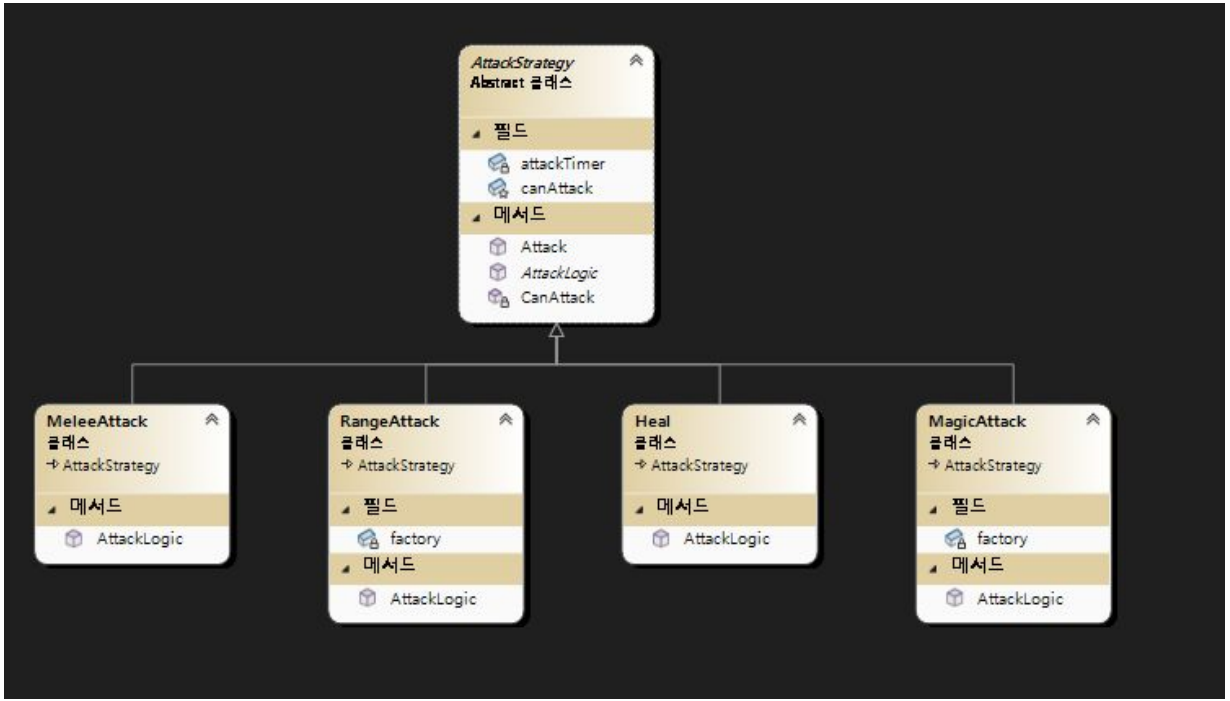
BaseState



IState<T> 인터페이스를 통해 상태 패턴을 구현하며, BaseState<T>가 상속받아 공통 기능을 제공
BaseState<T>를 상속받은 상태클래스들은 EnterState,UpdateState에서 해당상태에 맞는 로직을 구현

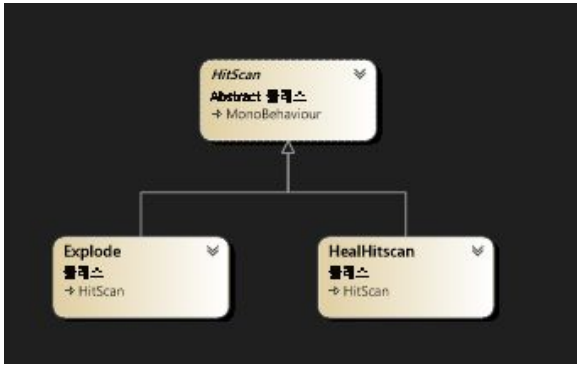
클래스 다이어그램

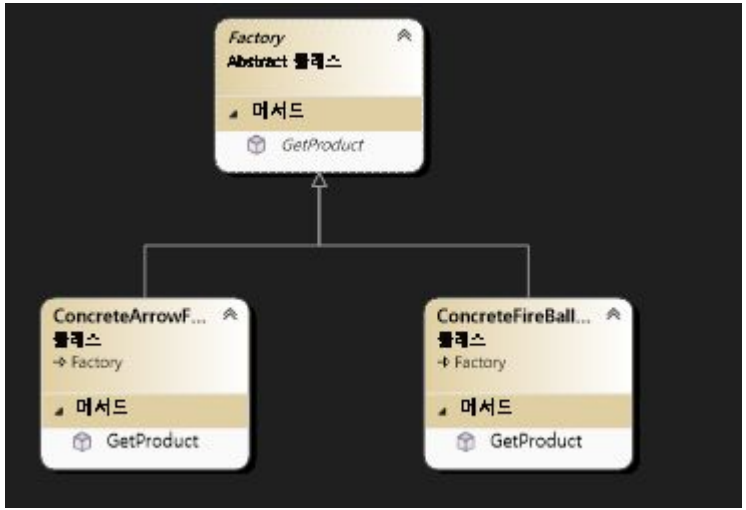
AttackStrategy

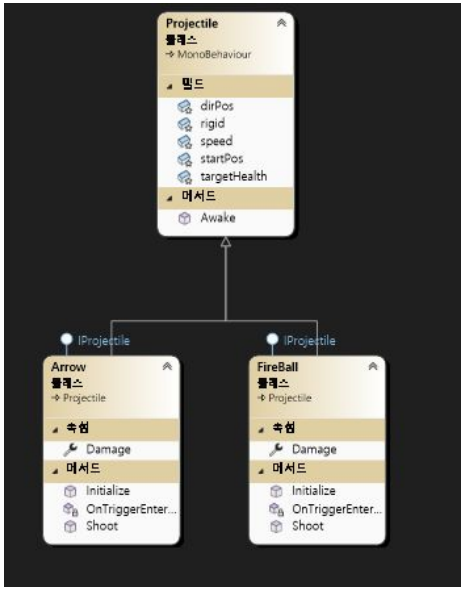


AttackStrategy는 공격 로직의 기본적인 틀을 제공하는 추상 클래스이며
하위 클래스는 그 틀을 구체적으로 구현하는 자식 클래스
공통 로직을 재사용하면서도 각각의 공격에 맞는 구체적인 기능을 추가하도록 설계
코드의 재사용성과 유연성을 높이는 구조를 시각적으로 표현

HitScan, Factory, Projectile

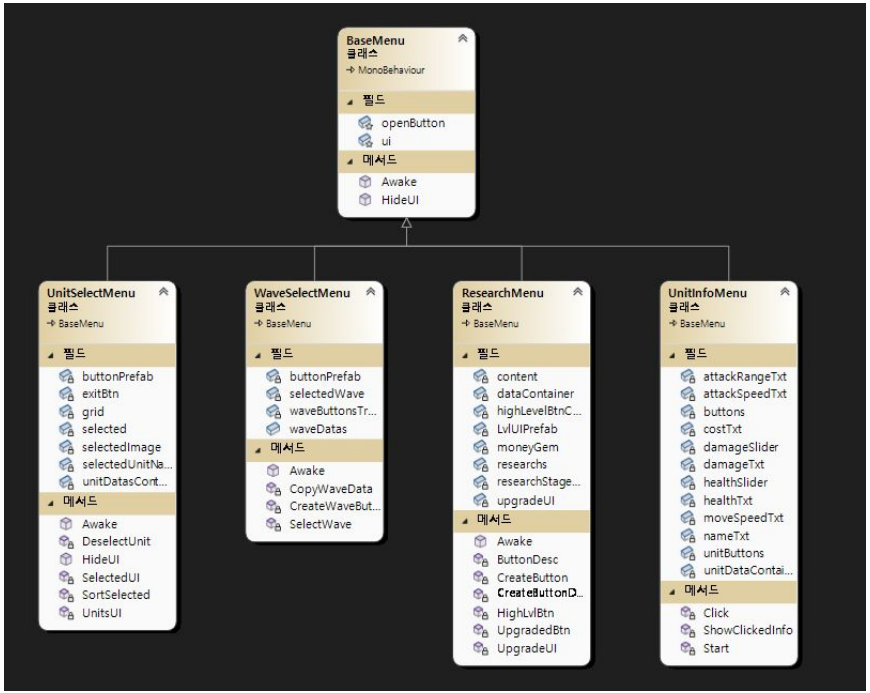






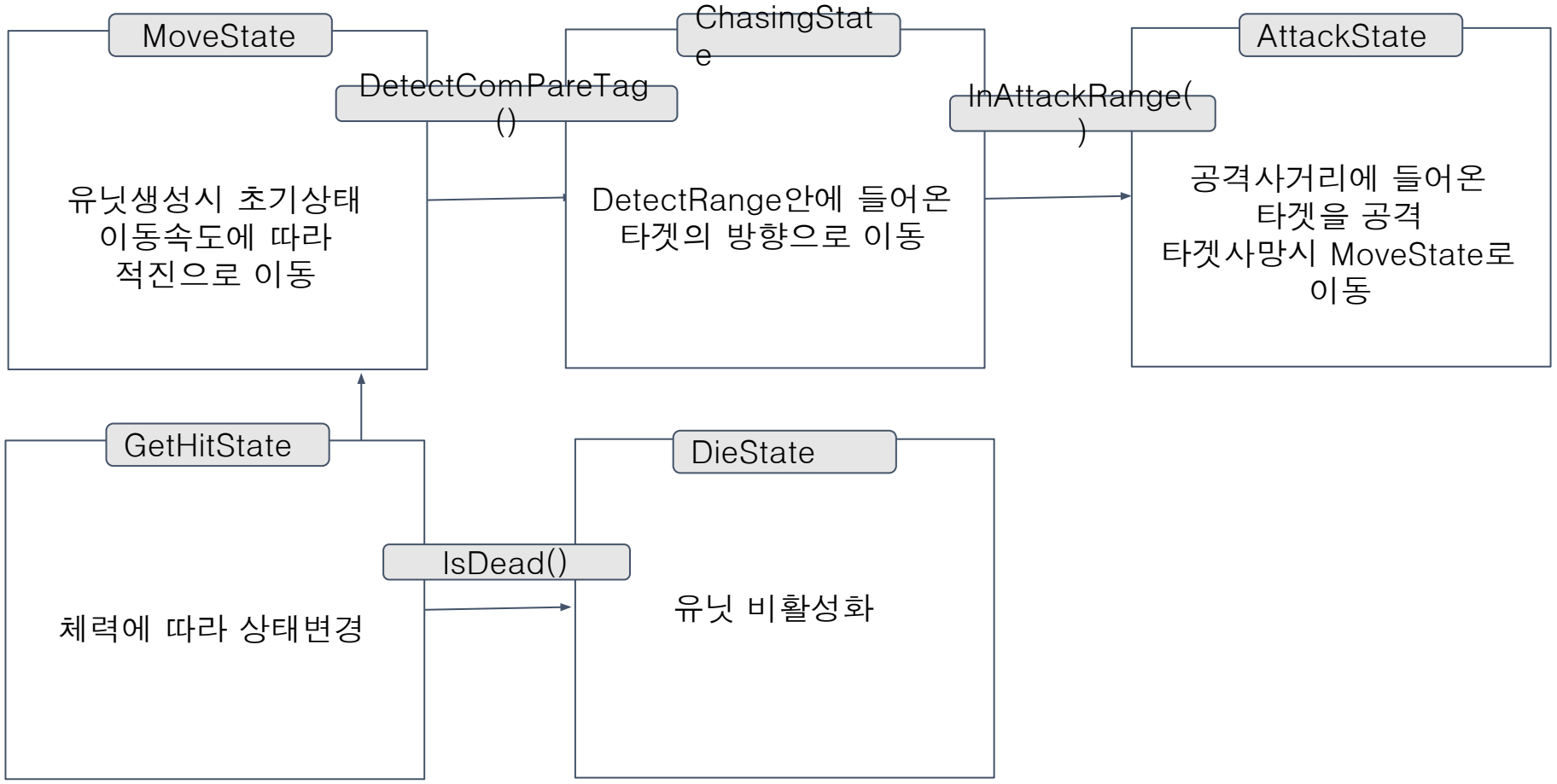
팩토리 패턴을 사용하여 투사체 인스턴스를 생성하는 기초 클래스

Menu



메뉴의 기본 UI 요소와 상호작용을 처리하는 기능을 제공

State패턴 흐름


MoveState
유닛생성시 초기상태
이동속도에 따라
적진으로 이동
DetectComPareTag
()
ChasingStat
e
DetectRange안에 들어온
타겟의 방향으로 이동
InAttackRange(
)
AttackState
공격사거리에 들어온
타겟을 공격
타겟사망시 MoveState로
이동
GetHitState
체력에 따라 상태변경
IsDead()
DieState
유닛 비활성화

01. UnitState

```
참조 41개
public enum EUnit...

[RequireComponent(typeof(Rigidbody2D), typeof(CircleCollider2D))]
Unity 스크립트 | 참조 12개
public class UnitState<T> : MonoBehaviour, IUnitHeatlh, IUnitData
{
    protected IState<T> currentState;
    protected Dictionary<EUnit, IState<T>> states = new Dictionary<EUnit, IState<T>>();
    Slider healthBar;
    TMP_Text healthTxt;

    public SearhTarget searhTarget;
    참조 14개
    public float Health { get; set; }
    참조 40개
    [field: SerializeField] public UnitData UnitData { get; set; }
    [SerializeField] protected LayerMask targetLayer;
    [HideInInspector] public Rigidbody2D rigid;
    [HideInInspector] public Animator animator;
    Unity 메시지 | 참조 8개
    public virtual void Awake()...
    Unity 메시지 | 참조 8개
    public virtual void OnEnable()...
    참조 3개
    public void Init()...
    참조 14개
    public void TransitionToState(EUnit estate)...
    참조 3개
    public void GetHit(float damage)...
    참조 2개
    public bool IsMaxHealth()...
    참조 2개
    public void GetHeal(float amount)...
    참조 2개
    public bool IsDead()...
    참조 2개
    public void Die()...

}
```

EUnit : 유닛의 다양한 상태를 정의
Health : 현재 체력
Awake : 애니메이터, 리지드바디 등을 초기화
GetHit : 피격시 GetHitState로 전환
TransitionToState : 상태전환, 전환시EnterState()호출

02. SearchTarget

```
참조 3개
public class SearhTarget
{
    Transform transform;
    UnitData unitData;
    LayerMask targetLayer;
    참조 1개
    public SearhTarget(Transform transform,UnitData unitData,LayerMask targetLayer)
    {
        this.transform = transform;
        this.unitData = unitData;
        this.targetLayer = targetLayer;
    }

    참조 6개
    public Transform TargetTransform()...
    참조 1개
    public bool DetectComPareTag(string detectTag, out Action<float> targetHealth)...
    참조 1개
    public bool DetectComPareTag(string detectTag)...
    참조 3개
    public Transform HealTargetTransform()...
    참조 1개
    public bool DetectHealComPareTag(string detectTag, out Action<float> targetHealth)...
    참조 3개
    public string TargetTag()...
}
```

TargetTransform :
범위 안의 가장 가까운 타겟을 찾아 반환

DetectComPareTag :
탐지된 타겟여부
HealTargetTransform :
탐지 범위 내에서 회복이 필요한 아군

DetectHealComPareTag :
회복이 필요한 아군여부

TargetTag :
탐지 레이어의 이름을 반환

03. MeleeUnit

```
Unity 스크립트(자산 참조 16개) | 참조 7개
public class MeleeUnit : UnitState<MeleeUnit>
{
    Unity 메시지 | 참조 6개
    public override void Awake()
    {
        base.Awake();
        states.Add(EUnit.Idle, new IdleState<MeleeUnit>(UnitData));
        states.Add(EUnit.Move, new MoveState<MeleeUnit>(UnitData));
        states.Add(EUnit.Chasing, new ChasingState<MeleeUnit>(UnitData));
        states.Add(EUnit.Attack, new AttackState<MeleeUnit>(UnitData, new MeleeAttack()));
        states.Add(EUnit.GetHit, new GetHitState<MeleeUnit>(UnitData));
        states.Add(EUnit.Die, new DieState<MeleeUnit>(UnitData));

    }
    Unity 메시지 | 참조 6개
    public override void OnEnable()
    {
        base.OnEnable();
        TransitionToState(EUnit.Move);
    }
    Unity 메시지 | 참조 0개
    private void Update()
    {
        currentState.UpdateState(this);
    }
}
```

states : 딕셔너리에 유닛의 다양한 행동 상태 등록

04. MoveState

```
참조 5개 | Added by tlgh0903@gmail.com on 2024년 8월 27일 화요일
public class MoveState<T> : BaseState<T> where T : UnitState<T>
{
    참조 4개 | Added by tlgh0903@gmail.com on 2024년 8월 27일 화요일
    public MoveState(UnitData unitData) ...
    참조 3개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void EnterState()...
    참조 6개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void UpdateState(T owner)
    {
        owner.rigid.velocity = Convert.ToInt32(unitData.MoveDir) * Vector2.right * MyUtil.MoveSpeed(unitData.MoveSpeed);
        owner.animator.SetBool("IsMove", true);

        if (owner.searhTarget.DetectComPareTag(owner.searhTarget.TargetTag()))
            owner.TransitionToState(EUnit.Chasing);

    }
}
```

이동, 애니메이션로직
적유닛감지시 ChasingState로 전환

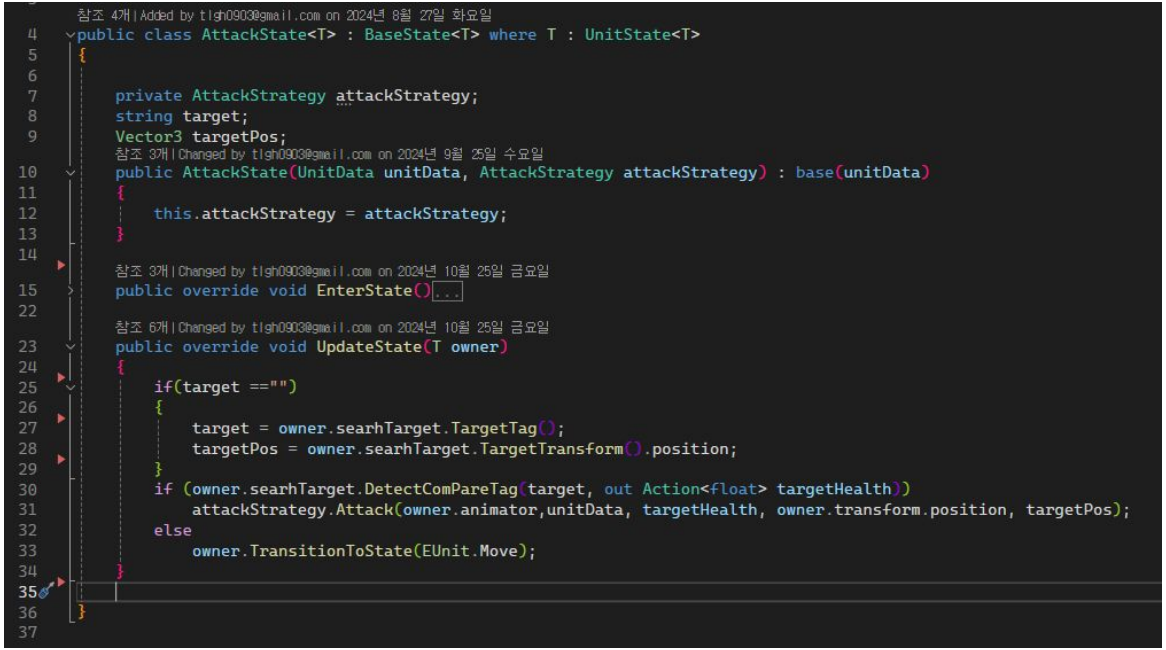
05. ChasingState

```
public class ChasingState<T> : BaseState<T> where T : UnitState<T>
{
    Transform enemyTransform;
    참조 3개 | Added by tlgh0903@gmail.com on 2024년 8월 20일 화요일
    public ChasingState(UnitData unitData) ...

    참조 3개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void EnterState()...
    참조 6개 | Changed locally
    public override void UpdateState(T owner)
    {
        enemyTransform = owner.searhTarget.TargetTransform();
        Vector3 dir = Vector3.zero;
        if (enemyTransform == null)
            owner.TransitionToState(EUnit.Move);
        else
        {
            dir = enemyTransform.position - owner.transform.position;
            dir.Normalize();
            InAttackRange(owner);
        }

        owner.rigid.velocity = dir * MyUtil.MoveSpeed(unitData.MoveSpeed);
        owner.animator.SetBool("IsMove", true);
    }

    참조 1개 | Added locally
    private void InAttackRange(T owner)
    {
        float distance = Vector3.Distance(enemyTransform.position, owner.transform.position);
        if (distance <= unitData.AttackRange)
        {
            owner.rigid.velocity = Vector3.zero;
            owner.animator.SetBool("IsMove", false);
            owner.TransitionToState(EUnit.Attack);
        }
    }
```

ChasingState :
추적 중인 적이 죽으면 즉시 MoveState로 전환하고, 적이 살아 있다면 계속해서 적의 위치로 이동

InAttackRange :
메서드는 적이 공격 범위 내에 들어오면 유닛의 이동을 멈추고 AttackState로 전환하여 공격 시작

06. AttackState



```
참조 4개 | Added by tlgh0903@gmail.com on 2024년 8월 27일 화요일
public class AttackState<T> : BaseState<T> where T : UnitState<T>
{

    private AttackStrategy attackStrategy;
    string target;
    Vector3 targetPos;
    참조 3개 | Changed by tlgh0903@gmail.com on 2024년 9월 25일 수요일
    public AttackState(UnitData unitData, AttackStrategy attackStrategy) : base(unitData)
    {
        this.attackStrategy = attackStrategy;
    }

    참조 3개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void EnterState()...

    참조 6개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void UpdateState(T owner)
    {
        if(target =="")
        {
            target = owner.searhTarget.TargetTag();
            targetPos = owner.searhTarget.TargetTransform().position;
        }
        if (owner.searhTarget.DetectComPareTag(target, out Action<float> targetHealth))
            attackStrategy.Attack(owner.animator,unitData, targetHealth, owner.transform.position, targetPos);
        else
            owner.TransitionToState(EUnit.Move);
    }

}
```

EnterState : target,targetPos 초기화
UpdateState : 타겟태그, 타겟위치 설정 DetectComPareTag 메서드로 타겟의 체력을 가져와 attackStrategy의 Attack 메서드를 호출

07. GetHitState

```
참조 5개 | Added by tlgh0903@gmail.com on 2024년 9월 4일 수요일
public class GetHitState<T> : BaseState<T> where T : UnitState<T>
{
    참조 4개 | Added by tlgh0903@gmail.com on 2024년 9월 4일 수요일
    public GetHitState(UnitData unitData) : base(unitData)
    {
    }

    참조 3개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void EnterState()
    {
        SoundManager.Instance.PlaySFX(SoundType.GetHit);

    }
    참조 6개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void UpdateState(T owner)
    {
        if (owner.IsDead())
        {
            owner.TransitionToState(EUnit.Die);
        }
        else
        {
            owner.animator.SetTrigger("GetHit");
            if (owner.animator.GetCurrentAnimatorStateInfo(0).IsName("Hurt") &&
                owner.animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.99f)
                owner.TransitionToState(EUnit.Move);
        }
    }
}
```

유닛이 사망했는지 확인하여 **DieState**로 전환하거나, 애니메이션이 끝나면 **MoveState**로 전환

08. DieState

```
참조 5개 | Added by tlgh0903@gmail.com on 2024년 8월 27일 화요일
public class DieState<T> : BaseState<T> where T : UnitState<T>
{
    참조 4개 | Added by tlgh0903@gmail.com on 2024년 8월 27일 화요일
    public DieState(UnitData unitData) ...

    참조 3개 | Changed locally
    public override void EnterState()
    {
        SoundManager.Instance.PlaySFX(SoundType.Die);
        Debug.Log(unitData.UnitName + "사망");
    }

    참조 6개 | Changed locally
    public override void UpdateState(T owner)
    {
        Debug.Log("사망 Update");
        owner.Die();
    }
}
```

유닛사망시 오브젝트 비활성화

09. AttackStrategy

```
참조 8개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
public abstract class AttackStrategy
{
    float attackTimer;
    protected bool canAttack;

    참조 1개 | Renamed from TryAttack to CanAttack by tigh0903@gmail.com on 2024년 10월 29일 화요일
    bool CanAttack(float attackSpeed)
    {
        if (attackTimer >= attackSpeed)
        {
            attackTimer = 0f;
            return true;
        }
        else
        {
            attackTimer += Time.deltaTime;
            return false;
        }
    }

    참조 2개 | Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    public void Attack(Animator animator,UnitData unitData, Action<float> targetHealth,Vector3 position,Vector3 targetPos)
    {
        if (CanAttack(unitData.AttackSpeed))
        {
            animator.SetTrigger("Attack");
            canAttack = true;
        }
        AttackLogic(animator,unitData, targetHealth, position,targetPos);
    }
    참조 5개 | Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    public abstract void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth,Vector3 position, Vector3 targetPos);

}
```

공격 행동의 기본 틀을 정의하는 추상 클래스
CanAttack : 공격 속도에 따라 공격 가능 여부
Attack : 애니메이션 재생 및 AttackLogic호출
AttackLogic : 각 공격타입마다 구체적인 공격로직 구현
(공격로직이 애니메이션의 공격모션과 일치하게 적용되야해서 if구문 밖에 있음)

10. MeleeAttack

```
참조 1개 | Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
public class MeleeAttack : AttackStrategy
{
    참조 2개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
    {
        if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
            animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
        {
            SoundManager.Instance.PlaySFX(SoundType.MeleeAttack);
            targetHealth?.Invoke(unitData.Damage);


            canAttack = false;
        }
    }
}
```

타겟에게 데미지를 입히고 사운드를 재생하는 근접 공격 로직

11. RangeAttack

```
참조 1개 | Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
public class RangeAttack : AttackStrategy
{
    Factory factory;
    참조 2개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
    {
        if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
            animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
        {
            SoundManager.Instance.PlaySFX(SoundType.RangeAttack);
            factory = new ConcreteArrowFactory();

            factory.GetProduct(unitData.Damage,position,targetHealth,targetPos);

            canAttack = false;
        }
    }
}
```

ConcreteArrowFactory를 사용하여 화살을 생성
GetProduct 메서드를 호출하여 공격력, 위치, 타겟 정보를 전달

12. MagicAttack

```
참조 1개 | Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
public class MagicAttack : AttackStrategy
{
    Factory factory;
    참조 2개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
    {
        Debug.Log("MagicAttack");

        if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
            animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
        {
            SoundManager.Instance.PlaySFX(SoundType.MagicAttack);
            factory = new ConcreteFireBallFactory();
            factory.GetProduct(unitData.Damage, position, targetHealth, targetPos);

            canAttack = false;
        }
    }
}
```

ConcreteFireBallFactory를 사용하여 불덩이를 생성
GetProduct메소드를 호출하여 공격력, 위치, 타겟 정보를 전달

13. Heal

```
참조 1개 | Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
public class Heal : AttackStrategy
{

    참조 2개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
    {
        Debug.Log("힐");
        if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
            animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
        {
            SoundManager.Instance.PlaySFX(SoundType.Heal);
            PoolManager.Instance.Get(PoolEnum.HitScan, "HealEffect", targetPos, Quaternion.identity);
            targetHealth(unitData.Damage);
            canAttack = false;
        }
    }
}
```

PoolManager.Instance.Get : "HealEffect" 프리팹을 targetPos에 생성
공격력으로 설정된 값을 타겟의 체력 회복을 위해 targetHealth 액션에 전달

14. Factory

```
참조 4개 | Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
public abstract class Factory
{
    참조 4개 | Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
    public abstract IProjectile GetProduct(float damage,Vector3 position, Action<float> targetHealth, Vector3 targetPos);

}
```

GetProduct : 구체적인 팩토리 클래스에서 오버라이드되어, 지정된 위치에 투사체를 생성

15. ConcreteArrowFactory, ConcreteFireBallFacroty

```
참조 1개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
public class ConcreteFireBallFactory : Factory
{
    참조 3개 | Changed by tigh0903@gmail.com on 2024년 10월 17일 목요일
    public override IProjectile GetProduct(float damage, Vector3 position, Action<float> targetHealth, Vector3 targetPos)
    {
        GameObject instance = PoolManager.Instance.Get(PoolEnum.Projectile, "FireBall", position, Quaternion.identity);
        FireBall fireBall = instance.GetComponent<FireBall>();
        fireBall.Initialize(damage, targetHealth);
        fireBall.Shoot(targetPos);
        return fireBall;
    }
}
```

```
참조 1개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
public class ConcreteArrowFactory : Factory
{
    참조 3개 | Changed locally
    public override IProjectile GetProduct(float damage,Vector3 position, Action<float> targetHealth, Vector3 targetPos)
    {
        GameObject instance = PoolManager.Instance.Get(PoolEnum.Projectile, "Arrow01", position, Quaternion.identity);
        Arrow arrow = instance.GetComponent<Arrow>();
        arrow.Initialize(damage, targetHealth);
        arrow.Shoot(targetPos);

        return arrow;
    }
}
```

GetProduct : PoolManager로부터 오브젝트를 가져와 초기화하고 targetPos방향으로 발사

16. Arrow

```
Unity 스크립트(자산 참조 1개)|참조 2개|Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
public class Arrow : Projectile, IProjectile
{
    참조 3개|Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
    public float Damage { get; set; }
    참조 2개|Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
    public void Initialize(float unitDamage, Action<float> targetHealth)
    {
        Damage = unitDamage;
        startPos = transform.position;
        this.targetHealth = targetHealth;
    }

    참조 2개|Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
    public void Shoot(Vector3 targetPosition)
    {
        dirPos = targetPosition - startPos;
        dirPos.Normalize();

        rigid.AddForce(dirPos * speed, ForceMode2D.Impulse);
    }

    Unity 메시지|참조 0개|Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Enemy"))
        {
            targetHealth.Invoke(Damage);
            gameObject.SetActive(false);
        }
    }
}
```

Initialize : 화살의 데미지와 시작 위치를 초기화하며, 타겟의 체력을 조정하기 위한 targetHealth 델리게이트를 설정
Shoot : targetPosition방향으로 발사
OnTriggerEnter2D : targetHealth를 호출하여 적의 체력을 감소

17. FireBall

```
Unity 스크립트(자산 참조 1개)|참조 2개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
public class FireBall : Projectile, IProjectile
{
    참조 4개|Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
    [field:SerializeField] public float Damage { get; set; }

    참조 2개|Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public void Initialize(float unitDamage, Action<float> targetHealth)
    {
        Damage = unitDamage;
        startPos = transform.position;
        this.targetHealth = targetHealth;
    }
    참조 2개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
    public void Shoot(Vector3 targetPosition)
    {
        dirPos = targetPosition - startPos;
        dirPos.Normalize();

        rigid.AddForce(dirPos * speed, ForceMode2D.Impulse);
    }
    Unity 메시지|참조 0개|Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Enemy"))
        {
            targetHealth.Invoke(Damage);
            GameObject explode = PoolManager.Instance.Get(PoolEnum.HitScan, "Explode", transform.position, Quaternion.identity);
            SoundManager.Instance.PlaySFX(SoundType.Explode);
            explode.GetComponent<Explode>().Damage = Mathf.Round(Damage/2);
            gameObject.SetActive(false);
        }
    }
}
```

적과 충돌시 데미지를 입힘
2차폭발오브젝트를 생성하고 발사체는 비활성화

18. Explode

```
3   [RequireComponent(typeof(CircleCollider2D))]
    Unity 스크립트(자산 참조 1개)|참조 1개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
4   public class Explode : HitScan
5   {
        Unity 메시지|참조 0개|Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
6       private void OnEnable()
7       {
8           StartCoroutine(SetActiveFalse());
9       }
        Unity 메시지|참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
10      public override void OnTriggerEnter2D(Collider2D collision)
11      {
12          if (collision.CompareTag("Enemy"))
13          {
14              collision.GetComponent<IUnitHeatlh>().GetHit(Damage);
15
16              canAttack = false;
17          }
18      }
19  }
20
```

2차 폭발 오브젝트
적과 충돌시 충돌한적에게 피해를 입힘
의도하지않은 중복피해를 방지하고자 canAttack = false;

19. HitScan

```
Unity 스크립트|참조 2개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
public abstract class HitScan : MonoBehaviour
{
    참조 2개|Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
    [field : SerializeField]public float Damage { get; set; }
    protected Animator animator;
    protected bool canAttack;
    Unity 메시지|참조 0개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
    public virtual void Awake()
    {
        animator = GetComponent<Animator>();
    }

    Unity 메시지|참조 2개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
    public abstract void OnTriggerEnter2D(Collider2D collision);

    참조 3개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
    protected IEnumerator SetActiveFalse()
    {
        while (true)
        {
            AnimatorStateInfo stateInfo = animator.GetCurrentAnimatorStateInfo(0);

            if (stateInfo.normalizedTime >= 1f)
            {
                gameObject.SetActive(false);
                break;
            }
            yield return null;
        }
    }

}
```

즉시 시전되는 오브젝트의 추상클래스
SetActiveFalse : 애니메이션이 끝나면 비활성화

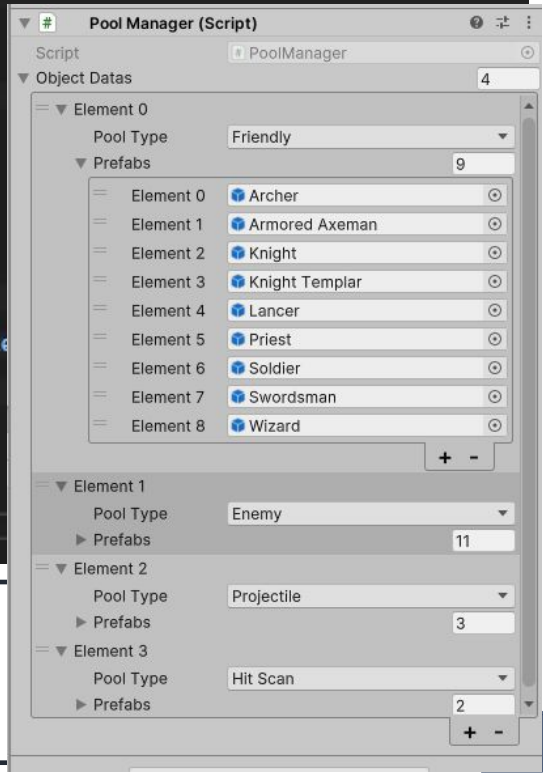
20. HealHitScan

```
4   [RequireComponent(typeof(CircleCollider2D))]
    Unity 스크립트(자산 참조 1개)|참조 0개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일
5   public class HealHitscan : HitScan
6   {
        Unity 메시지|참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 29일 화요일
7       public override void OnTriggerEnter2D(Collider2D collision)
8       {
9           if (collision.CompareTag("Friendly"))
10          {
11              StartCoroutine(SetActiveFalse());
12              canAttack = false;
13          }
14      }
15  }
```

아군과 접촉시 애니메이션 재생후 비활성화

21. PoolManager

```
참조 10개
4   public enum PoolEnum...

11
12  [System.Serializable]
    참조 3개
13  public class PoolType...
21
    Unity 스크립트(자산 참조 1개)|참조 10개
22  public class PoolManager : Singleton<PoolManager>
23  {
24      [SerializeField] List<PoolType> objectDatas = new List<PoolType>();
25
        Unity 메시지|참조 1개
26      public override void Awake()...
        참조 8개
46      public GameObject Get(PoolEnum prefabTypes, string name, Vector3 startPos, Quaternion quate
        참조 1개
73      public void DeactivateAllChildren()...
80
81
82  }
83
84
```



Awake : 프리팹리스트 초기화
Get : 오브젝트 생성
DeactiveAllChildren : 생성한오브젝트 비활성화 (게임 종료시 사용)

23. SoundManager

```
4   //사운드이름 enum타입으로 선언
    참조 18개 | Added by tlgh0903@gmail.com on 2024년 10월 17일 목요일
5   public enum SoundType...
    참조 3개 | Added by tlgh0903@gmail.com on 2024년 10월 17일 목요일
19  public enum BGMType...
25  [RequireComponent(typeof(AudioSource))]
    Unity 스크립트(자산 참조 3개) | 참조 25개 | Changed by tlgh0903@gmail.com on 2024년 10월 17일
26  public class SoundManager : Singleton<SoundManager>
27  {
28      [SerializeField] AudioSource sfxAudioSource;
29      [SerializeField] AudioSource bgmAudioSource;
30      [SerializeField] GameObject soundUI;
31      [SerializeField] Slider BGMSlider;
32      [SerializeField] Slider SFXSlider;
33
34      public AudioClip[] BGM;
35      public AudioClip[] SFX;
36
        Unity 메시지 | 참조 0개 | Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
37      private void Start()...
        Unity 메시지 | 참조 0개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
43      private void Update()
44      {
45          SoundUI();
46          BGMVolume();
47          SFXVolume();
48      }
        참조 1개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
49      private void SoundUI()...
        참조 17개 | Added by tlgh0903@gmail.com on 2024년 10월 17일 목요일
61      public void PlaySFX(SoundType sound)...
        참조 2개 | Added by tlgh0903@gmail.com on 2024년 10월 17일 목요일
65      public void PlayBGM(BGMType sound)...
        참조 1개 | Added by tlgh0903@gmail.com on 2024년 10월 17일 목요일
72      void BGMVolume()...
        참조 1개 | Added by tlgh0903@gmail.com on 2024년 10월 17일 목요일
76      void SFXVolume()...
80  }
```

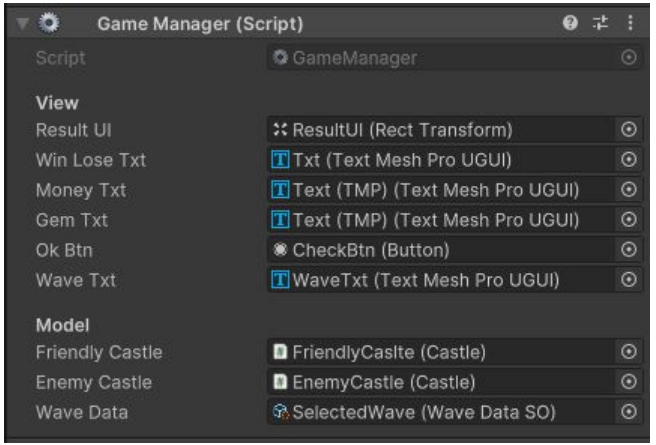
SoundType :
효과음을 구분하는 데 사용
BGMType:
배경음악 종류를 정의

Start :
JsonLoad 메서드를 통해 JSON 파일에 저장된 값을 로드후 BGMSlider와 SFXSlider에 적용

PlaySFX :
인덱스를 활용해 미리 정의된 SFX 배열에서 해당하는 효과음을 PlayOneShot으로 재생
PlayBGM:
BGMType을 받아 AudioSource에 해당 배경음을 재생하고 반복 재생

22. GameManager

```
9   public class GameManager : MonoBehaviour, ICastleObserver
10  {
11      [Header("View")]
12      [SerializeField] Transform resultUI;
13      [SerializeField] TMP_Text winLoseTxt;
14      [SerializeField] TMP_Text moneyTxt;
15      [SerializeField] TMP_Text gemTxt;
16      [SerializeField] Button okBtn;
17      [SerializeField] TMP_Text waveTxt;
18
19      [Header("Model")]
20      [SerializeField] Castle friendlyCastle;
21      [SerializeField] Castle enemyCastle;
22      [SerializeField] WaveDataSO waveData;
23
        참조 3개
24      public float Money { get; set; }
        참조 3개
25      public float Gem { get; set; }
26
        Unity 메시지 | 참조 0개
27      private void Awake()...
        참조 3개
39      public void OnCastleDestroyed(Castle castle)...
        참조 1개
52      private void Win()...
        참조 1개
71      private void Lose()...
77
78  }
```



Awake :	캐슬 파괴 이벤트 구독, 보상UI의 버튼리스너 추가
OnCastleDestroyed :	캐슬 파괴(게임종료) 시 호출, 보상UI 활성화
Win :	보상 JSON으로 저장
Lose :	패배화면 출력

24. Castle

```
Unity 스크립트(자산 참조 2개)|참조 6개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
public class Castle : MonoBehaviour, IUnitHeatlh
{
    public CastleData castleData;
    참조 13개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
    public float Health { get; set; }
    Slider healthBar;
    TMP_Text healthTxt;

    public Action<Castle> CastleDestroy;
    Unity 메시지|참조 0개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
    public void Awake()...
    Unity 메시지|참조 0개|Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    private void OnEnable()...
    참조 3개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
    public void Init()...
    참조 3개|Changed locally
    public void GetHit(float damage)
    {
        Health -= damage;
        Health = Mathf.Clamp(Health,0,castleData.Health);
        UpdateHealthBar();
        if (IsDead())
            Die();
    }
    참조 2개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
    public bool IsMaxHealth()...
    참조 2개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
    public void GetHeal(float amount)...
    참조 2개|Added by tigh0903@gmail.com on 2024년 10월 11일 금요일
    public bool IsDead()...
    참조 3개|Moved up 2 positions by tigh0903@gmail.com on 2024년 10월 29일 화요일
    private void UpdateHealthBar()...
    참조 2개|Moved up 2 positions by tigh0903@gmail.com on 2024년 10월 29일 화요일
    public void Die()
    {
        CastleDestroy?.Invoke(this);
        Debug.Log(castleData.UnitName + " 가 파괴되었습니다");
        CastleDestroy = null;
    }
```

체력이 0이 되면 CastleDestroy 액션에 등록된 메서드가 호출

의도치 않은 중복 호출을 방지하기 위해 CastleDestroy를 null로 설정하여 모든 이벤트 구독을 제거

25. BaseMenu

```
Unity 스크립트 | 참조 4개
public class BaseMenu : MonoBehaviour
{
    [SerializeField] protected GameObject ui;
    [SerializeField] protected Button openButton;
    Unity 메시지 | 참조 6개
    public virtual void Awake()
    {
        openButton.onClick.AddListener(() =>
        {
            ui.SetActive(true);
            SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
        });
    }
    참조 3개
    public virtual void HideUI()
    {
        SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
        ui.SetActive(false);
    }
}
```

UI의 기본적인 열기와 닫기 기능제공

26. UnitInfoMenu

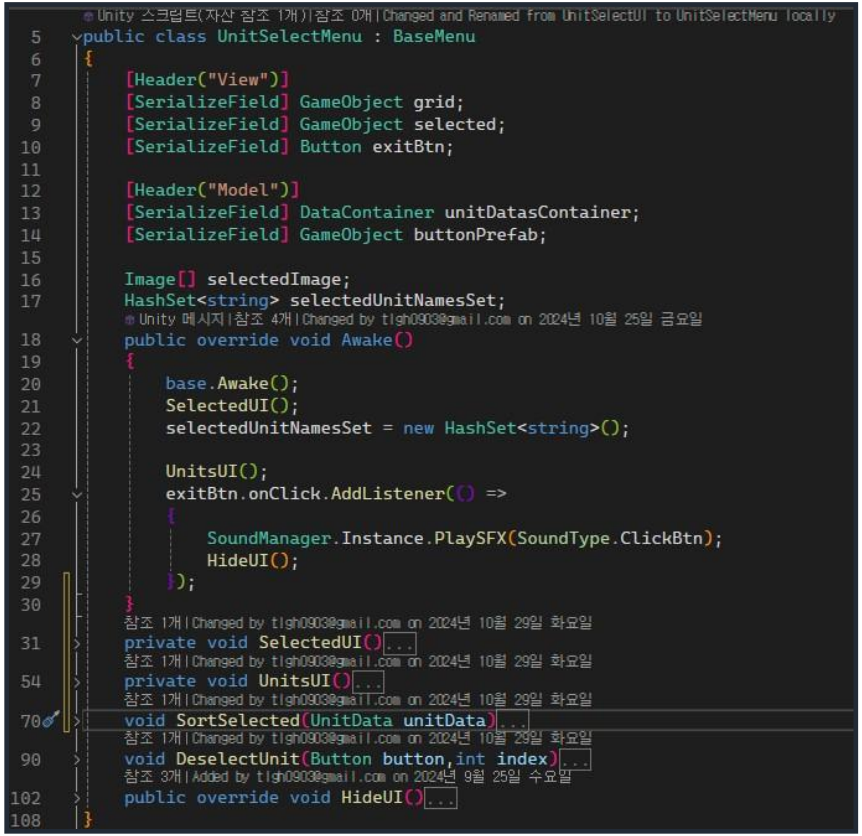
```
5   public class UnitInfoMenu : BaseMenu
6   {
7       [Header("View")]
8       [SerializeField] GameObject unitButtons;
9       [SerializeField] TMP_Text nameTxt;
10      [SerializeField] TMP_Text healthTxt;
11      [SerializeField] TMP_Text damageTxt;
12      [SerializeField] Slider healthSlider;
13      [SerializeField] Slider damageSlider;
14      [SerializeField] TMP_Text attackSpeedTxt;
15      [SerializeField] TMP_Text attackRangeTxt;
16      [SerializeField] TMP_Text moveSpeedTxt;
17      [SerializeField] TMP_Text costTxt;
18      Button[] buttons;
19
20      [Header("Model")]
21      [SerializeField] DataContainer unitDataContainer;
        @ Unity 메시지|참조 0개|Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
22      void Start()
23      {
24          buttons = unitButtons.GetComponentsInChildren<Button>();
25
26          foreach (var button in buttons)
27          {
28              button.onClick.AddListener(() =>
29              {
30                  SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
31                  Click(button.name);
32              });
33          }
34          foreach (UnitData unitData in unitDataContainer.UnitDatas)
35              unitData.MaxHealthNDamage();
36      }
        참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
37      void Click(string buttonName)...
        참조 1개|Changed locally
47      void ShowClickedInfo(UnitData clickedUnit)...
```

Start :
unitButtons의 모든버튼에 클릭 이벤트를 설정

Click :
button.Name과 일치하는 유닛 데이터를 찾아
ShowClickedInfo 메서드를 호출

ShowClickedInfo :
텍스트 필드를 갱신하여, 유닛의 이름, 레벨,
체력, 공격력 등 다양한 정보표시

27. UnitSelectMenu



```
Unity 스크립트(자산 참조 1개)|참조 0개|Changed and Renamed from UnitSelectUI to UnitSelectMenu locally
public class UnitSelectMenu : BaseMenu
{
    [Header("View")]
    [SerializeField] GameObject grid;
    [SerializeField] GameObject selected;
    [SerializeField] Button exitBtn;

    [Header("Model")]
    [SerializeField] DataContainer unitDatasContainer;
    [SerializeField] GameObject buttonPrefab;

    Image[] selectedImage;
    HashSet<string> selectedUnitNamesSet;
    Unity 메시지|참조 4개|Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
    public override void Awake()
    {
        base.Awake();
        SelectedUI();
        selectedUnitNamesSet = new HashSet<string>();

        UnitsUI();
        exitBtn.onClick.AddListener(() =>
        {
            SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
            HideUI();
        });
    }
    참조 1개|Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    private void SelectedUI()...
    참조 1개|Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    private void UnitsUI()...
    참조 1개|Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    void SortSelected(UnitData unitData)...
    참조 1개|Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
    void DeselectUnit(Button button,int index)...
    참조 3개|Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
    public override void HideUI()...
}
```

SelectedUI :
선택된 유닛을 표시하는 UI를 초기화
UnitsUI :
선택 가능한 유닛 리스트를 표시하는 UI를 초기화
SortSelected :
유닛 선택, 빈 슬롯에 배치
DeselectUnit :
유닛 선택 해제, 슬롯에서 제거

28. ResearchMenu

```
22  public class ResearchMenu : BaseMenu
23  {
24      [Header("View")]
25      [SerializeField] UpgradeUI upgradeUI;
26      [SerializeField] Transform content;
27      [SerializeField] GameObject researchStageUIPrefab;
28      [SerializeField] GameObject LvlUIPrefab;
29
30      [Header("Model")]
31      [SerializeField] DataContainer dataContainer;
32      [SerializeField] MoneyGem moneyGem;
33
34      Color highLevelBtnColor = new Color(255 / 255, 112/ 255, 98/ 255, 1);
35      UpgradeLevels[] researchs;
        # Unity 메시지 | 참조 4개 | Moved down 3 positions by tlgh0903@gmail.com on 2024년 10월 29일 화요일
36      public override void Awake()
37      {
38          base.Awake();
39          moneyGem.LoadResource();
40          CreateButton();
41          ButtonDesc(0,dataContainer.castleData, "체력");
42          ButtonDesc(1,dataContainer.supplyResources, "속도","충탄");
43          int index = 2;
44          foreach (UnitData unitData in dataContainer.UnitDatas)
45          {
46              ButtonDesc(index, unitData, "체력","공격력");
47              index++;
48          }
49          SoundManager.Instance.PlayBGM(BGMType.Prepare);
50      }
        참조 1개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
51      void CreateButton()...
        참조 1개 | Changed by tlgh0903@gmail.com on 2024년 10월 28일 화요일
65      void CreateButtonData(int index, UnitData unitData)...
        참조 2개 | Changed by tlgh0903@gmail.com on 2024년 10월 28일 화요일
82      void CreateButtonData(int index, string unitName, int upgradeLength, string unitKorName)...
97
        참조 3개 | Changed locally
98      void ButtonDesc(int researchIndex,IUpgradeableUnit unit, string growthTxt1, string growthTxt2 = null)...
        참조 1개 | Changed locally
153     void HighLvlBtn(Button button)...
        참조 2개 | Changed locally
160     void UpgradedBtn(Button button,Image line)...
        참조 1개 | Changed locally
168     void UpgradeUI(IUpgradeableUnit unitData, int index, Button button, Button nextbutton, Image line,string growthTxt1,string growthTxt2=null)...
217 }
```

CreateButton :
성, 자원, 유닛업그레이드 버튼생성

ButtonDesc :
업그레이드 버튼의 내용 초기화 및 상태 설정

HighLvlBtn :
고레벨 버튼의 색상과 상태를 비활성화 상태로 설정

UpgradedBtn :
업그레이드된 버튼과 진행 라인의 상태를 설정

UpgradeUI :
업그레이드 UI를 초기화하고 버튼의 상호작용을 설정

29. WaveSelectMenu

```
Unity 스크립트(자산 참조 1개)|참조 4개|Changed and Renamed from WaveSelect to WaveSelectMenu locally
public class WaveSelectMenu : BaseMenu
{
    [Header("View")]
    [SerializeField] Transform waveButtonsTransform;

    [Header("Model")]
    public WaveDataSO[] waveDatas;
    [SerializeField] WaveDataSO selectedWave;
    [SerializeField] GameObject buttonPrefab;

    Unity 메시지|참조 4개|Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
    public override void Awake()
    {
        base.Awake();
        CreateWaveButtons();
        int waveindex = 0;
        WaveClearData waveClearData = MyUtil.JsonLoad<WaveClearData>(MyUtil.JsonFileName.WaveClear);
        waveindex = (waveClearData != null) ? waveClearData.WaveClear : 0;
        int waveClear = waveindex + 1;
        for (int i = 0; i < waveClear; i++)
        {
            waveButtonsTransform.GetChild(i).GetComponentInChildren<Button>().interactable = true;
        }
    }

    참조 1개|Changed locally
    private void CreateWaveButtons()...
    참조 1개|Changed locally
    private void SelectWave(int index)...
    참조 1개|Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
    private void CopyWaveData(WaveDataSO origin, WaveDataSO copy)...
}
```

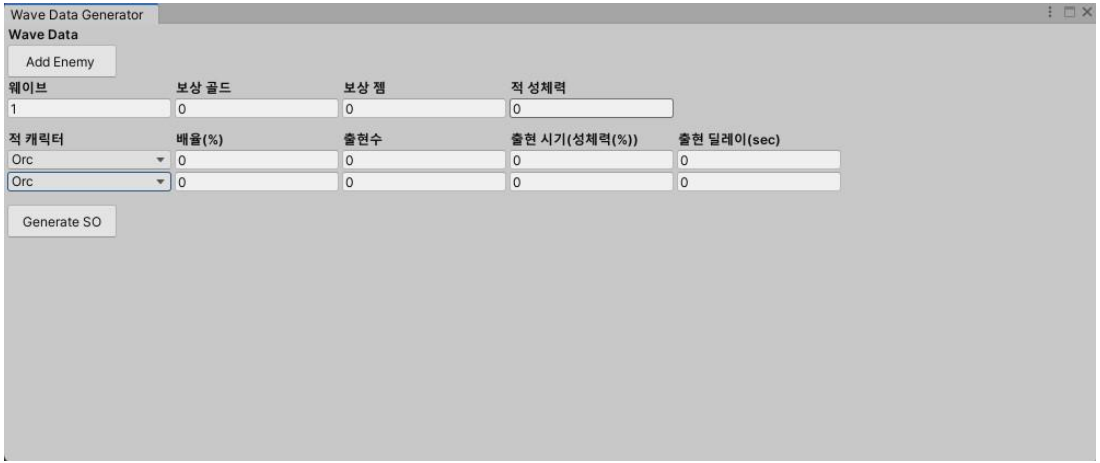
CreateWaveButtons :
웨이브 선택 버튼을 생성하고 각 버튼에
웨이브 데이터 연결

SelectWave :
Battle씬으로 이동
CopyWaveData :
선택된 웨이브 데이터를 복사

30. GenerateWaveDataSO

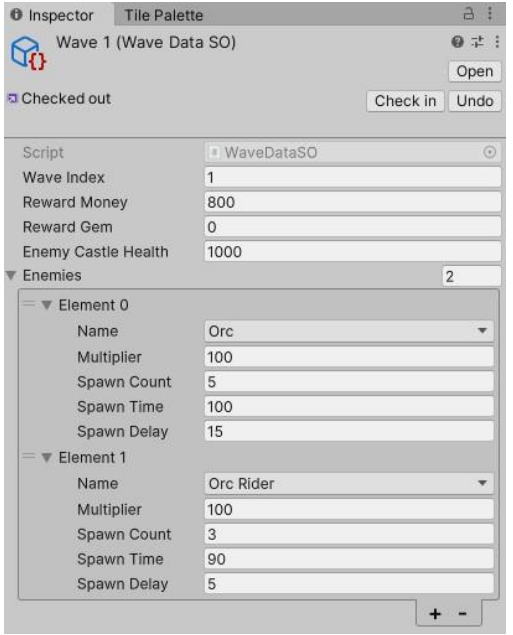
```
Unity 스크립트|참조 1개|Added by tigh09030@gmail.com on 2024년 10월 11일 금요일
public class GenerateWaveDataSO : EditorWindow
{
    private int waveIndex = 1;
    private int rewardMoney;
    private int rewardGem;
    private int enemyCastleHealth;

    private List<EnemyData> enemies = new List<EnemyData>();

    참조 3개|Added by tigh09030@gmail.com on 2024년 10월 11일 금요일
    private class EnemyData
    {
        public EnemyName name;
        public float multiplier;
        public int spawnCount;
        public int spawnTime;
        public int spawnDelay;
    }
    [MenuItem("MyUtilities/Generate Wave SO")]
    참조 0개|Added by tigh09030@gmail.com on 2024년 10월 11일 금요일
    public static void ShowWindow()...
    Unity 메시지|참조 0개|Added by tigh09030@gmail.com on 2024년 10월 11일 금요일
    private void OnGUI()...
    참조 1개|Changed by tigh09030@gmail.com on 2024년 10월 25일 금요일
    private void GenerateSO()...
}
```

웨이브 데이터를 입력받아 WaveDataSO 스크립터블 오브젝트로 생성하는 에디터 창
적 캐릭터의 세부 정보를 추가할 수 있으며, 생성된 데이터는 에셋으로 저장



31. WaveDataSO

```
Unity 스크립트|참조 15개|Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
public class WaveDataSO : ScriptableObject
{
    public int waveIndex;
    public int rewardMoney;
    public int rewardGem;
    public float enemyCastleHealth;
    public List<EnemyData> enemies = new List<EnemyData>();

    [System.Serializable]
    참조 7개|Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    public class EnemyData
    {
        public EnemyName name;
        public float multiplier;
        public int spawnCount;
        public int spawnTime;
        public int spawnDelay;
    }
}
참조 3개|Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
public enum EnemyName...
```



생성한 웨이브의 스크립트와 인스펙터창

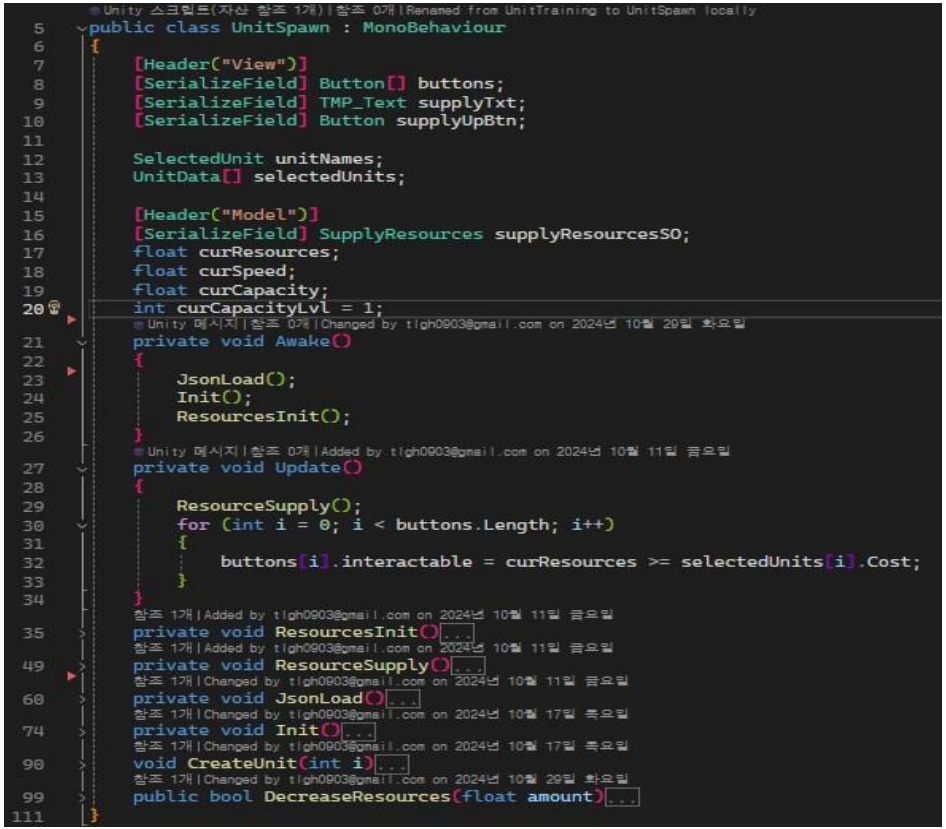
32. EnemySpawn

```
public class EnemySpawn : MonoBehaviour
{
    public WaveDataSO waveData;

    public List<UnitData> unitDatas = new List<UnitData>();
    float[] spawnTimer;
    int[] curSpawnCount;
    [SerializeField] Castle enemycastle;
    int tempCount;
    Unity 메시지 | 참조 0개 | Changed and Moved down 4 positions and Renamed from Timer to Awake by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    private void Awake()
    {
        enemycastle.castleData.Health = waveData.enemyCastleHealth;
        spawnTimer = new float[waveData.enemies.Count];
        curSpawnCount = new int[waveData.enemies.Count];
    }
    Unity 메시지 | 참조 0개 | Changed and Moved down 4 positions and Renamed from Timer to Awake by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    private void Start()
    {
        for (int i = 0; i < waveData.enemies.Count; i++)
        {
            GameObject unit = PoolManager.Instance.Get(PoolEnum.Enemy, waveData.enemies[i].name.ToString(), MyUtil.SpawnRandomPos(47f, 54f, -4f, -6.5f), Quaternion.identity);
            UnitData unitData = unit.GetComponent<IUnitData>().UnitData;

            CopyUnitData(unitData, unitDatas[i]);
            EnemyMultiplier(unitDatas[i], waveData.enemies[i].multiplier);
            unit.GetComponent<IUnitData>().UnitData = unitDatas[i];
            unit.SetActive(false);
        }
    }

    Unity 메시지 | 참조 0개 | Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
    private void Update()
    {
        for (int i = 0; i < waveData.enemies.Count; i++)
            Timer(i, waveData.enemies[i].spawnCount, waveData.enemies[i].spawnDelay);
    }
    참조 1개 | Changed and Moved down 5 positions and Renamed from UnitUpgrade to Update by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    void Timer(int index,int spawnCount,float spawnDelay)...
    참조 1개 | Added by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    void Spawn(int index)...
    참조 1개 | Changed by tlgh0903@gmail.com on 2024년 10월 28일 화요일
    private void CopyUnitData(UnitData origin, UnitData copy)...
    참조 1개 | Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
    void EnemyMultiplier(UnitData unitData, float multiplier)...
}
```

CreateWaveButtons :
웨이브 선택 버튼을 생성하고 각 버튼에 웨이브 데이터 연결

SelectWave :
Battle씬으로 이동
CopyWaveData : 선택된 웨이브 데이터를 복사

33. UnitSpawn



```
Unity 스크립트(자산 참조 1개)|참조 0개|Renamed from UnitTraining to UnitSpawn locally
public class UnitSpawn : MonoBehaviour
{
    [Header("View")]
    [SerializeField] Button[] buttons;
    [SerializeField] TMP_Text supplyTxt;
    [SerializeField] Button supplyUpBtn;

    SelectedUnit unitNames;
    UnitData[] selectedUnits;

    [Header("Model")]
    [SerializeField] SupplyResources supplyResourcesSO;
    float curResources;
    float curSpeed;
    float curCapacity;
    int curCapacityLvl = 1;
    Unity 메시지|참조 0개|Changed by tlgh0903@gmail.com on 2024년 10월 29일 화요일
    private void Awake()
    {
        JsonLoad();
        Init();
        ResourcesInit();
    }
    Unity 메시지|참조 0개|Added by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    private void Update()
    {
        ResourceSupply();
        for (int i = 0; i < buttons.Length; i++)
        {
            buttons[i].interactable = curResources >= selectedUnits[i].Cost;
        }
    }
    참조 1개|Added by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    private void ResourcesInit()...
    참조 1개|Added by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    private void ResourceSupply()...
    참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    private void JsonLoad()...
    참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
    private void Init()...
    참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
    void CreateUnit(int i)...
    참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 29일 화요일
    public bool DecreaseResources(float amount)...
}
```

Update :
자원이 충분할 경우 유닛 생성 버튼을 활성화
ResourcesInit :
초기 자원 속도와 용량을 설정하고,
supplyUpBtn 버튼에 레벨 증가 로직을 추가해 용량을 증가
ResourceSupply :
자원 공급 속도에 따라 현재 자원을 증가시킵니다. 자원 업그레이드 버튼 활성화 여부를 결정
JsonLoad :
선택된 유닛 정보를 JSON에서 불러와 초기화고 유닛을 준비
Init :
유닛 버튼을 설정하고, 각 버튼에 유닛 생성 버튼 클릭 이벤트를 추가
CreateUnit :
자원이 충분할 경우 해당 유닛을 생성 위치에 스폰하고 자원을 차감
DecreaseResources :
지정된 양의 자원을 차감하며 자원이 충분한지 확인

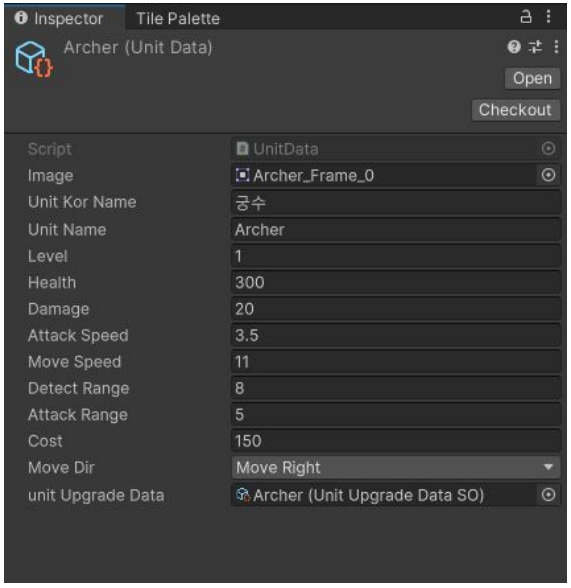
34. MyUtil

```
참조 43개 | Added by tlgh0903@gmail.com on 2024년 8월 16일 금요일
public static class MyUtil
{
    참조 3개 | Added by tlgh0903@gmail.com on 2024년 8월 16일 금요일
    public static float MoveSpeed(float moveSpeed)...
    참조 3개 | Added by tlgh0903@gmail.com on 2024년 9월 26일 수요일
    public static Vector3 SpawnRandomPos(float minX, float maxX, float minY, float maxY)...
    참조 5개 | Added by tlgh0903@gmail.com on 2024년 9월 26일 수요일
    public static void JsonSave<T>(T data,JsonFileName fileName)
    {
        Debug.Log(data);
        string json = JsonUtility.ToJson(data);
        Debug.Log("JsonSave : " + json);
        File.WriteAllText(Application.persistentDataPath + $"/{fileName}.json", json);
        Debug.Log("Application.persistentDataPath : " + Application.persistentDataPath);
    }
    참조 6개 | Changed by tlgh0903@gmail.com on 2024년 10월 11일 금요일
    public static T JsonLoad<T>(JsonFileName fileName)
    {
        string path = Application.persistentDataPath + $"/{fileName}.json";
        if (File.Exists(path))
        {
            string json = File.ReadAllText(path);
            T data = JsonUtility.FromJson<T>(json);
            Debug.Log("JsonLoad : " + json);
            return data;
        }
        else
        {
            Debug.Log("Statue_Save : null");
            return default(T);
        }
    }
    참조 17개 | Changed by tlgh0903@gmail.com on 2024년 10월 29일 화요일
    public static string StringBuilderTxt(params object[] values)
    {
        StringBuilder sb = new();
        sb.Clear();
        foreach (var value in values)
        {
            sb.Append(value);
        }
        return sb.ToString();
    }
    참조 12개 | Changed by tlgh0903@gmail.com on 2024년 10월 17일 목요일
    public enum JsonFileName...
}
```

MoveSpeed:
이동 속도 값에 Time.deltaTime을 곱해 반환
SpawnRandomPos :
X와 Y 좌표 범위를 지정해 무작위 스폰 위치를 생성하고 반환
JsonSave<T> :
데이터를 JSON 형식으로 변환하여 지정된 파일명으로 저장
JsonLoad<T> :
지정된 파일명에서 JSON 파일을 로드해 객체로 변환 후 반환합니다. 파일이 없으면 기본값을 반환
StringBuilderTxt :
전달된 여러 값들을 문자열로 연결해 하나의 텍스트로 반환

35. UnitData

```
4   [CreateAssetMenu(menuName = "Unit/UnitData", fileName = "UnitData")]
    Unity 스크립트 | 참조 36개
5   public class UnitData : ScriptableObject, IMove, IDamage, IUpgradeableUnit
6   {
        참조 67개
7       [field: SerializeField] public Sprite Image { get; set; }
        참조 87개
8       [field: SerializeField] public string UnitKorName { get; set; }
        참조 147개
9       [field: SerializeField] public string UnitName { get; set; }
        참조 107개
10      [field: SerializeField] public int Level { get; set; }
        참조 207개
11      [field: SerializeField] public float Health { get; set; }
        참조 167개
12      [field: SerializeField] public float Damage { get; set; }
        참조 67개
13      [field: SerializeField] public float AttackSpeed { get; set; }
        참조 87개
14      [field: SerializeField] public float MoveSpeed { get; set; }
        참조 77개
15      [field: SerializeField] public float DetectRange { get; set; }
        참조 67개
16      [field: SerializeField] public float AttackRange { get; set; }
        참조 77개
17      [field: SerializeField] public int Cost { get; set; }
        참조 67개
18      [field: SerializeField] public MoveDir MoveDir { get; set; }
        참조 147개
19      [field: SerializeField] public UnitUpgradeDataSO unitUpgradeData { get; set; }
        참조 37개
20      public float MaxHealth { get; set; }
        참조 37개
21      public float MaxDamage { get; set; }
        참조 17개
22      public void MaxHealthNDamage()...
        참조 27개
32      public void LevelUP(float health, float damage)...
        참조 27개
38      public float MoneyCost(int index) => unitUpgradeData.unitUpgrade[index].moneyCost;
        참조 27개
39      public float GemCost(int index) => unitUpgradeData.unitUpgrade[index].gemCost;
        참조 27개
40      public float CurGrowth1() => Health;
        참조 27개
41      public float CurGrowth2() => Damage;
        참조 27개
42      public float GetGrowth1(int index) => unitUpgradeData.unitUpgrade[index].growth1;
        참조 27개
43      public float GetGrowth2(int index) => unitUpgradeData.unitUpgrade[index].growth2;
```



깃 허브

https://github.com/SIHO0903/DefenseGame

유튜브 링크

https://youtu.be/fjwn0BxfbkA

감사합니다