

안녕하세요

유니티 클라이언트

최시호입니다.

---

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

## Defense

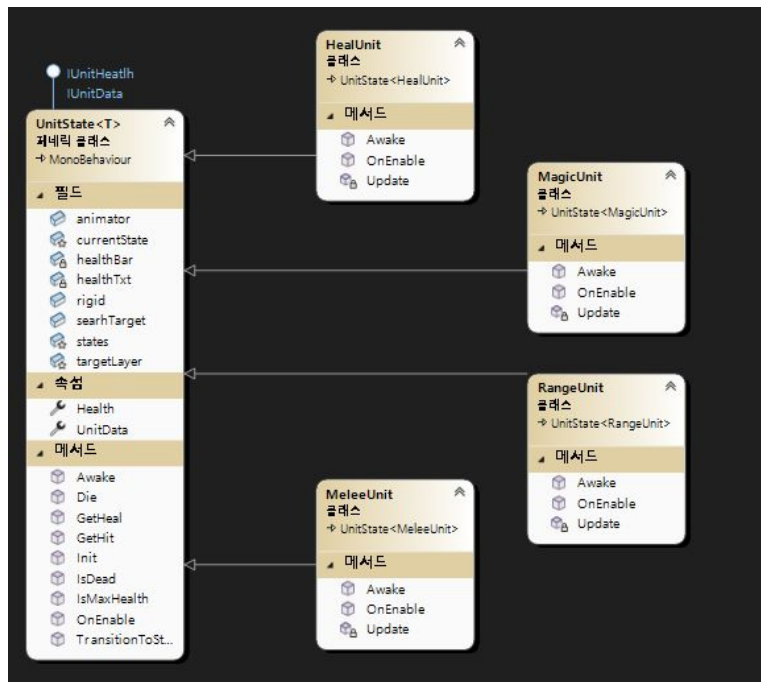
플랫폼	엔진	그래픽/시점	장르	개발 기간
PC	유니티	2D	Defense	10주
게임 설명	- 웨이브를 클리어하며 아군유닛 강화			
프로그래밍	- 1인			
주요 스크립트	<ul style="list-style-type: none"> <li>- UnitState, BaseState</li> <li>- AttackStrategy</li> <li>- Factory, HitScan</li> <li>- PoolManager, GameManager</li> <li>- Castle</li> <li>- BaseMenu</li> <li>- WaveDataSO, UnitData</li> <li>- GenerateWaveDataSO</li> <li>- UnitSpawn, EnemySpawn</li> </ul>			
깃 허브 주소	<a href="https://github.com/SIHO0903/DefenseGame">https://github.com/SIHO0903/DefenseGame</a>			
유튜브 링크	<a href="https://youtu.be/fjwn0BxfbkA">https://youtu.be/fjwn0BxfbkA</a>			



## 영상

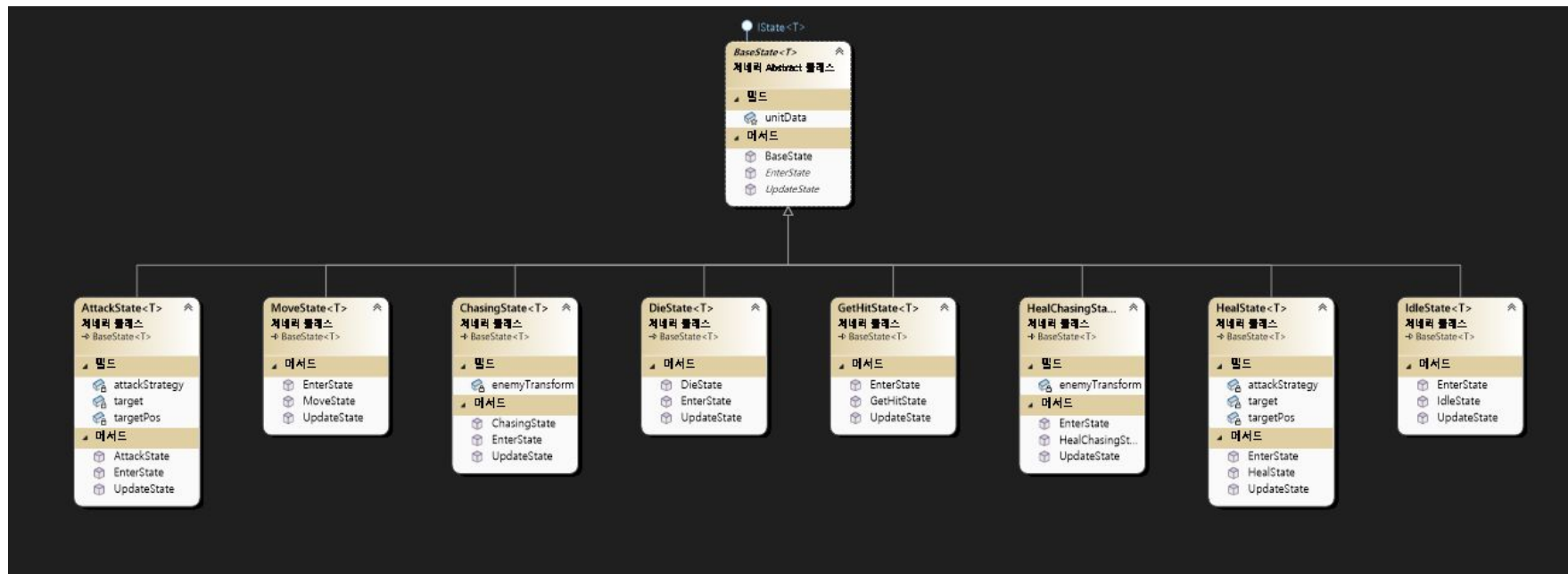


## UnitState



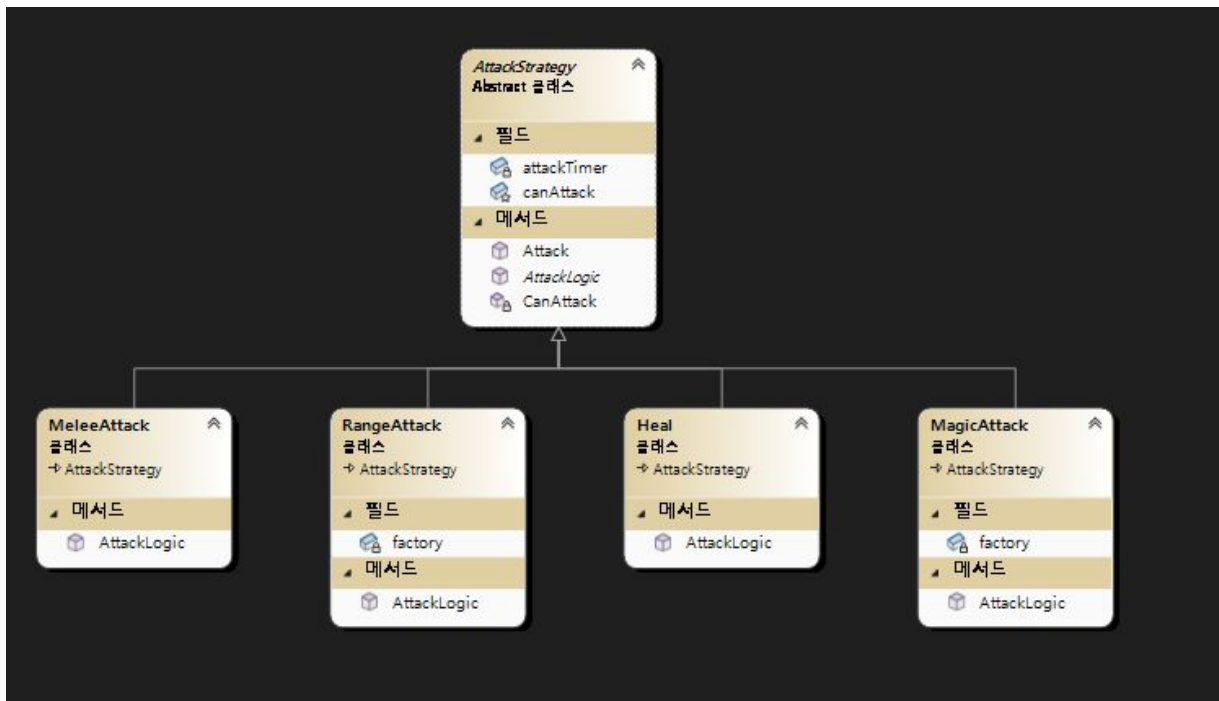
**UnitState<T>**가 기본 유닛의 상태 및 메소드를 정의하고, 하위클래스는 그 틀을 상속받아 근접 유닛의 특정 상태 및 동작을 구현하는 관계입니다.

## BaseState



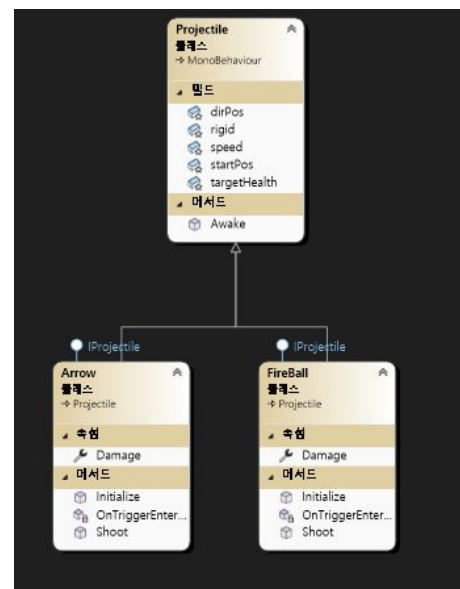
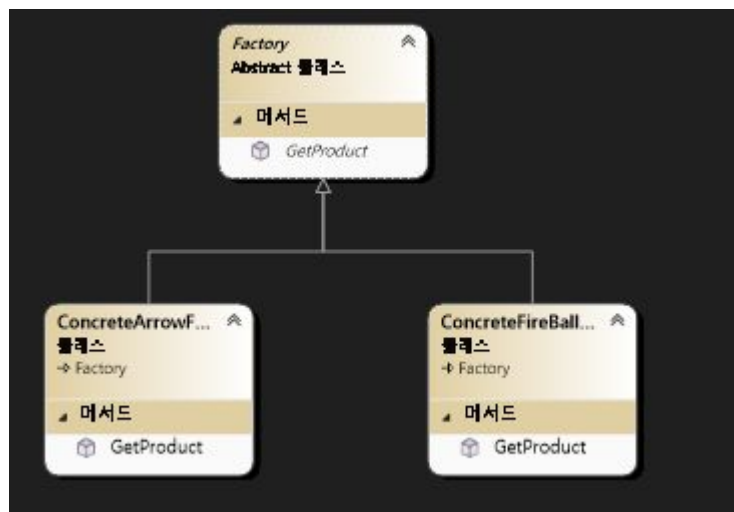
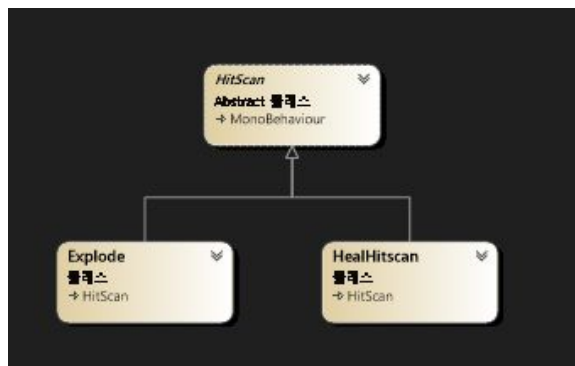
IState<T> 인터페이스를 통해 상태 패턴을 구현하며, BaseState<T>가 상속받아 공통 기능을 제공  
BaseState<T>를 상속받은 상태클래스들은 EnterState, UpdateState에서 해당상태에 맞는 로직을 구현

## AttackStrategy



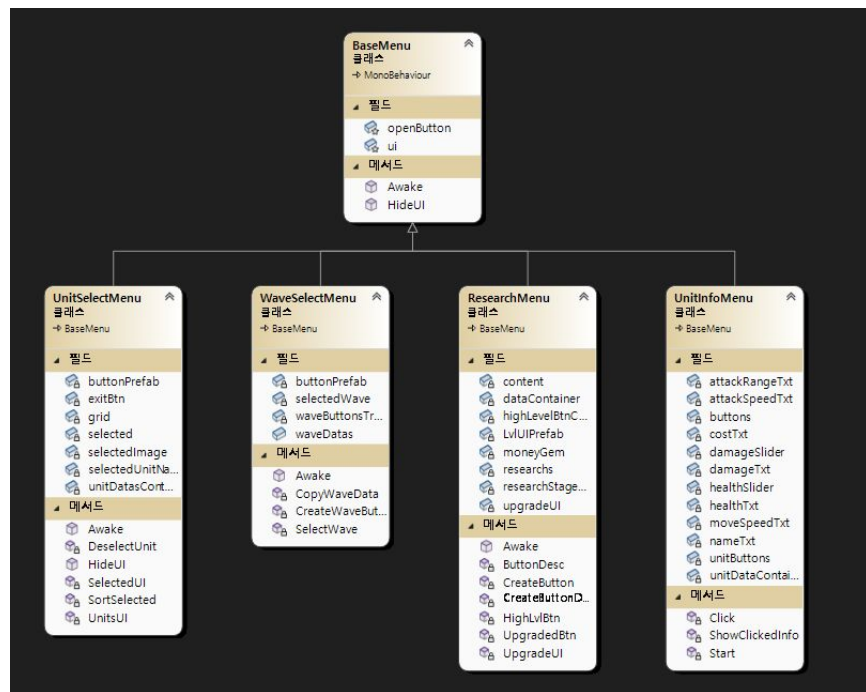
**AttackStrategy**는 공격 로직의 기본적인 틀을 제공하는 추상 클래스이며  
하위 클래스는 그 틀을 구체적으로 구현하는 자식 클래스  
공통 로직을 재사용하면서도 각각의 공격에 맞는 구체적인 기능을 추가하도록 설계  
코드의 재사용성과 유연성을 높이는 구조를 시각적으로 표현

## HitScan, Factory, Projectile



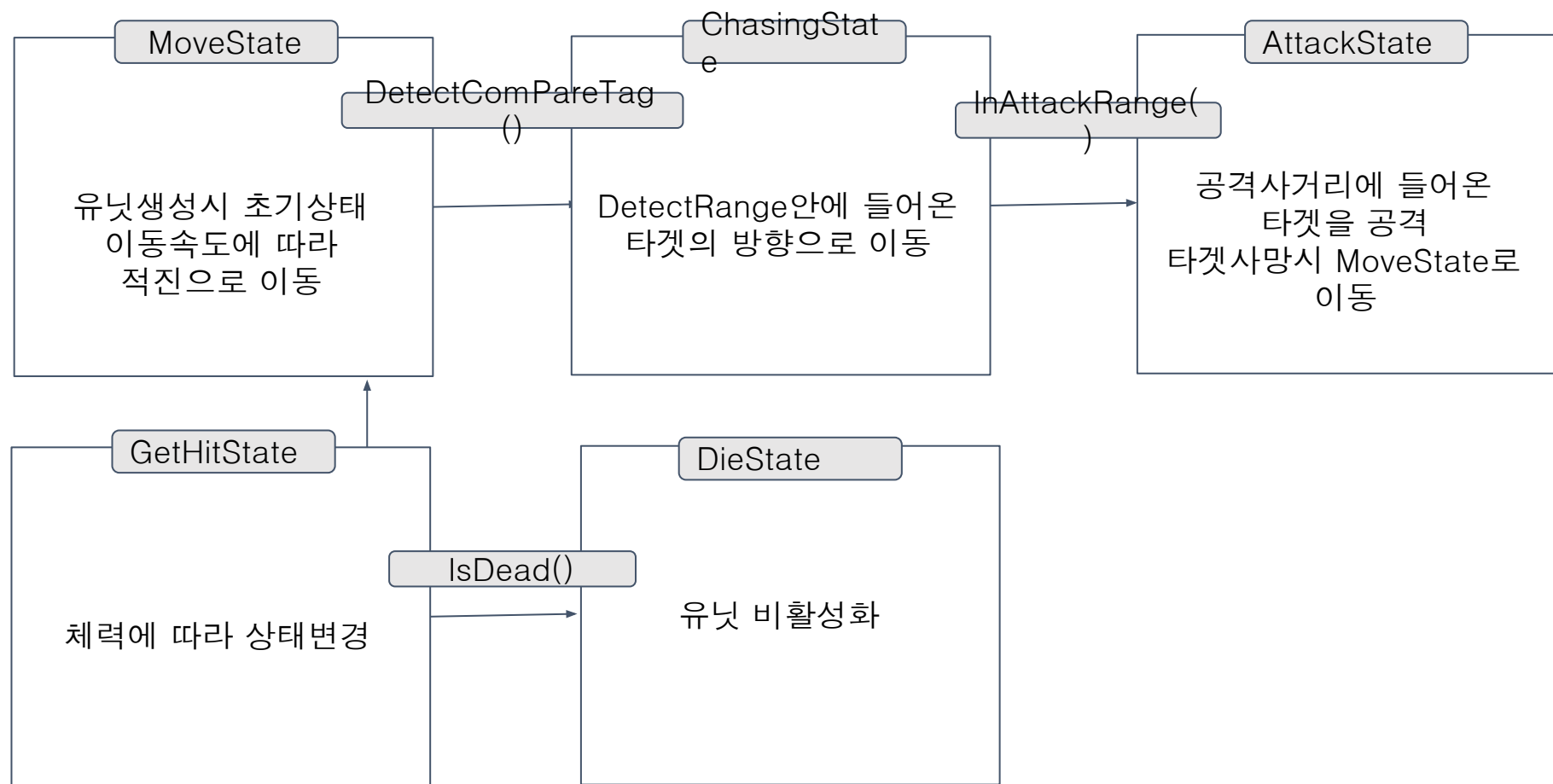
팩토리 패턴을 사용하여 투사체 인스턴스를 생성하는 기초 클래스

## Menu



메뉴의 기본 UI 요소와 상호작용을 처리하는 기능을 제공





## 01. UnitState

```

88 > public enum EUnit{...}
18
19 [RequireComponent(typeof(Rigidbody2D), typeof(CircleCollider2D))]
    // Unity 스크립트 | 참조: 12개
20 public class UnitState<T> : MonoBehaviour, IUnitHealth, IUnitData
21 {
22     protected IState<T> currentState;
23     protected Dictionary<EUnit, IState<T>> states = new Dictionary<EUnit, IState<T>>();
24     Slider healthBar;
25     TMP_Text healthTxt;
26
27     public SearchTarget searchTarget;
28     // 참조: 14개
29     public float Health { get; set; }
30     // 참조: 40개
31     [field: SerializeField] public UnitData UnitData { get; set; }
32     [SerializeField] protected LayerMask targetLayer;
33     [HideInInspector] public Rigidbody2D rigid;
34     [HideInInspector] public Animator animator;
35     // Unity 메시지 | 참조: 8개
36     public virtual void Awake(){}
37     // Unity 메시지 | 참조: 8개
41     public virtual void OnEnable(){}
42     // 참조: 3개
45     public void Init(){}
46     // 참조: 14개
51     public void TransitionToState(EUnit estate){}
52     // 참조: 3개
56     public void GetHit(float damage){}
57     // 참조: 2개
65     public bool IsMaxHealth(){}
66     // 참조: 2개
69     public void GetHeal(float amount){}
70     // 참조: 2개
77     public bool IsDead(){}
78     // 참조: 2개
81     public void Die(){}
82
87
88
89

```

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

## 02. SearchTarget

```

4  ~참조 3개
5  ~public class SearchTarget
6  {
7      Transform transform;
8      UnitData unitData;
9      LayerMask targetLayer;
10     ~참조 1개
11     ~public SearchTarget(Transform transform, UnitData unitData, LayerMask targetLayer)
12     {
13         this.transform = transform;
14         this.unitData = unitData;
15         this.targetLayer = targetLayer;
16     }
17
18     ~참조 6개
19     ~public Transform TargetTransform()...
20     ~참조 1개
21
22     ~참조 1개
23     ~public bool DetectComPareTag(string detectTag, out Action<float> targetHealth)...
24     ~참조 1개
25
26     ~참조 3개
27     ~public Transform HealTargetTransform()...
28     ~참조 1개
29
30     ~참조 3개
31     ~public bool DetectHealComPareTag(string detectTag, out Action<float> targetHealth)...
32     ~참조 3개
33
34     ~public string TargetTag()...
35 }
36
37 107

```

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

```
1  // Unity 스크립트(자산 참조 16개) | 참조 7개
2  public class MeleeUnit : UnitState<MeleeUnit>
3  {
4      // Unity 메시지 | 참조 6개
5      public override void Awake()
6      {
7          base.Awake();
8          states.Add(EUnit.Idle, new IdleState<MeleeUnit>(UnitData));
9          states.Add(EUnit.Move, new MoveState<MeleeUnit>(UnitData));
10         states.Add(EUnit.Chasing, new ChasingState<MeleeUnit>(UnitData));
11         states.Add(EUnit.Attack, new AttackState<MeleeUnit>(UnitData, new MeleeAttack()));
12         states.Add(EUnit.GetHit, new GetHitState<MeleeUnit>(UnitData));
13         states.Add(EUnit.Die, new DieState<MeleeUnit>(UnitData));
14     }
15     // Unity 메시지 | 참조 5개
16     public override void OnEnable()
17     {
18         base.OnEnable();
19         TransitionToState(EUnit.Move);
20     }
21     // Unity 메시지 | 참조 0개
22     private void Update()
23     {
24         currentState.UpdateState(this);
25     }
26 }
```

states : 딕셔너리에 유닛의 다양한 행동 상태 등록

## 04. MoveState

```
참조 5개 | Added by tigh0903@gmail.com on 2024년 8월 27일 화요일
4  > public class MoveState<T> : BaseState<T> where T : UnitState<T>
5  {
    참조 4개 | Added by tigh0903@gmail.com on 2024년 8월 27일 화요일
6  >   public MoveState(UnitData unitData) ...
    참조 3개 | Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
9  >   public override void EnterState() ...
    참조 6개 | Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
13 >   public override void UpdateState(T owner)
14 {
15     owner.rigid.velocity = Convert.ToInt32(unitData.MoveDir) * Vector2.right * MyUtil.MoveSpeed(unitData.MoveSpeed);
16     owner.animator.SetBool("IsMove", true);
17
18     if (owner.searchTarget.DetectComPareTag(owner.searchTarget.TargetTag()))
19         owner.TransitionToState(EUnit.Chasing);
20
21 }
22 }
23
```

이동, 애니메이션로직  
적유닛감지시 ChasingState로 전환

## 05. ChasingState

```
4 public class ChasingState<T> : BaseState<T> where T : UnitState<T>
5 {
6     Transform enemyTransform;
7     참조 3개 | Added by tlg903@gmail.com on 2024년 8월 20일 화요일
8     public ChasingState(UnitData unitData) { ... }
9
10    참조 3개 | Changed by tlg903@gmail.com on 2024년 10월 25일 금요일
11    public override void EnterState() { ... }
12    참조 6개 | Changed locally
13    public override void UpdateState(T owner)
14    {
15        enemyTransform = owner.searchTarget.TargetTransform();
16        Vector3 dir = Vector3.zero;
17        if (enemyTransform == null)
18            owner.TransitionToState(EUnit.Move);
19        else
20        {
21            dir = enemyTransform.position - owner.transform.position;
22            dir.Normalize();
23            InAttackRange(owner);
24        }
25        owner.rigid.velocity = dir * MyUtil.MoveSpeed(unitData.MoveSpeed);
26        owner animator.SetBool("IsMove", true);
27    }
28
29    참조 1개 | Added locally
30    private void InAttackRange(T owner)
31    {
32        float distance = Vector3.Distance(enemyTransform.position, owner.transform.position);
33        if (distance <= unitData.AttackRange)
34        {
35            owner.rigid.velocity = Vector3.zero;
36            owner animator.SetBool("IsMove", false);
37            owner.TransitionToState(EUnit.Attack);
38        }
39    }
40 }
```

### 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 t1gh0903@gmail.com on 2024년 8월 27일 화요일
4 public class AttackState<T> : BaseState<T> where T : UnitState<T>
5 {
6
7     private AttackStrategy attackStrategy;
8     string target;
9     Vector3 targetPos;
10 참조 3개 | Changed by t1gh0903@gmail.com on 2024년 9월 25일 수요일
11     public AttackState(UnitData unitData, AttackStrategy attackStrategy) : base(unitData)
12     {
13         this.attackStrategy = attackStrategy;
14     }
15 참조 3개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
16     public override void EnterState()
17     {
18
19     }
20 참조 6개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
21     public override void UpdateState(T owner)
22     {
23         if(target == "")
24         {
25             target = owner.searchTarget.TargetTag();
26             targetPos = owner.searchTarget.TargetTransform().position;
27         }
28         if (owner.searchTarget.DetectComPareTag(target, out Action<float> targetHealth))
29             attackStrategy.Attack(owner.animator,unitData, targetHealth, owner.transform.position, targetPos);
30         else
31             owner.TransitionToState(EUnit.Move);
32     }
33
34 }
35
36
37
```

EnterState : target,targetPos 초기화

UpdateState : 타겟태그, 타겟위치 설정 DetectComPareTag 메서드로 타겟의 체력을 가져와 attackStrategy의 Attack 메서드를 호출

## 07. GetHitState

```
참조 5개 | Added by t1gh0903@gmail.com on 2024년 9월 4일 수요일
2 public class GetHitState<T> : BaseState<T> where T : UnitState<T>
3
4 참조 4개 | Added by t1gh0903@gmail.com on 2024년 9월 4일 수요일
5 public GetHitState(UnitData unitData) : base(unitData)
6 {
7 }
8
9 참조 3개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
10 public override void EnterState()
11 {
12     SoundManager.Instance.PlaySFX(SoundType.GetHit);
13 }
14
15 참조 6개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
16 public override void UpdateState(T owner)
17 {
18     if (owner.IsDead())
19     {
20         owner.TransitionToState(EUnit.Die);
21     }
22     else
23     {
24         owner.animator.SetTrigger("GetHit");
25         if (owner.animator.GetCurrentAnimatorStateInfo(0).IsName("Hurt") &&
26             owner.animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.99f)
27             owner.TransitionToState(EUnit.Move);
28     }
29 }
```

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



## 08. DieState

```
2
3  참조 5개 | Added by tlg0903@gmail.com on 2024년 8월 27일 화요일
4  public class DieState<T> : BaseState<T> where T : UnitState<T>
5  {
6      참조 4개 | Added by tlg0903@gmail.com on 2024년 8월 27일 화요일
7      public DieState(UnitData unitData) { ... }
8
9      참조 3개 | Changed locally
10     public override void EnterState()
11     {
12         SoundManager.Instance.PlaySFX(SoundType.Die);
13         Debug.Log(unitData.UnitName + "사망");
14     }
15
16     참조 6개 | Changed locally
17     public override void UpdateState(T owner)
18     {
19         Debug.Log("사망 Update");
20         owner.Die();
21     }
22 }
```

유닛사망시 오브젝트 비활성화

## 09. AttackStrategy

```
참조 8개 | Added by tlg0903@gmail.com on 2024년 9월 25일 수요일
4 public abstract class AttackStrategy
5 {
6     float attackTimer;
7     protected bool canAttack;
8
9     참조 1개 | Renamed From TryAttack to CanAttack by tlg0903@gmail.com on 2024년 10월 29일 화요일
10    bool CanAttack(float attackSpeed)
11    {
12        if (attackTimer >= attackSpeed)
13        {
14            attackTimer = 0f;
15            return true;
16        }
17        else
18        {
19            attackTimer += Time.deltaTime;
20            return false;
21        }
22    }
23
24    참조 2개 | Changed by tlg0903@gmail.com on 2024년 10월 29일 화요일
25    public void Attack(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
26    {
27        if (CanAttack(unitData.AttackSpeed))
28        {
29            animator.SetTrigger("Attack");
30            canAttack = true;
31        }
32        AttackLogic(animator, unitData, targetHealth, position, targetPos);
33    }
34
35    참조 5개 | Changed by tlg0903@gmail.com on 2024년 10월 29일 화요일
36    public abstract void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos);
37 }
```

공격 행동의 기본 틀을 정의하는 추상 클래스

CanAttack : 공격 속도에 따라 공격 가능 여부

Attack : 애니메이션 재생 및 AttackLogic 호출

AttackLogic : 각 공격타입마다 구체적인 공격로직 구현

(공격로직이 애니메이션의 공격모션과 일치하게 적용되어야 해서 if구문 밖에 있음)

## 10. MeleeAttack

```
참조 1개 | Added by t1gh0903@gmail.com on 2024년 9월 25일 수요일
4 public class MeleeAttack : AttackStrategy
5 {
6     참조 2개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
7     public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
8     {
9         if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
10             animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
11         {
12             SoundManager.Instance.PlaySFX(SoundType.MeleeAttack);
13             targetHealth?.Invoke(unitData.Damage);
14
15             canAttack = false;
16         }
17     }
18 }
19
```

타겟에게 데미지를 입히고 사운드를 재생하는 근접 공격 로직

## 11. RangeAttack

```
3
4 참조 1개 | Added by tlg0903@gmail.com on 2024년 9월 25일 수요일
5 public class RangeAttack : AttackStrategy
6 {
7     Factory factory;
8     참조 2개 | Changed by tlg0903@gmail.com on 2024년 10월 25일 금요일
9     public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
10    {
11        if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
12            animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
13        {
14            SoundManager.Instance.PlaySFX(SoundType.RangeAttack);
15            factory = new ConcreteArrowFactory();
16
17            factory.GetProduct(unitData.Damage, position, targetHealth, targetPos);
18
19            canAttack = false;
20        }
21    }
```

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

## 12. MagicAttack

```
참조 1개 | Added by tlg0903@gmail.com on 2024년 9월 25일 수요일
4 public class MagicAttack : AttackStrategy
5 {
6     Factory factory;
7     참조 2개 | Changed by tlg0903@gmail.com on 2024년 10월 25일 금요일
8     public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
9     {
10         Debug.Log("MagicAttack");
11
12         if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
13             animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
14         {
15             SoundManager.Instance.PlaySFX(SoundType.MagicAttack);
16             factory = new ConcreteFireBallFactory();
17             factory.GetProduct(unitData.Damage, position, targetHealth, targetPos);
18
19             canAttack = false;
20         }
21     }
```

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

### 13. Heal

```
참조 1개 | Added by t1gh0903@gmail.com on 2024년 9월 25일 수요일
4 public class Heal : AttackStrategy
5 {
6
7     참조 2개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
8     public override void AttackLogic(Animator animator, UnitData unitData, Action<float> targetHealth, Vector3 position, Vector3 targetPos)
9     {
10         Debug.Log("힐");
11         if (animator.GetCurrentAnimatorStateInfo(0).IsName("Attack01") &&
12             animator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && canAttack)
13         {
14             SoundManager.Instance.PlaySFX(SoundType.Heal);
15             PoolManager.Instance.Get(PoolEnum.HitScan, "HealEffect", targetPos, Quaternion.identity);
16             targetHealth(unitData.Damage);
17             canAttack = false;
18         }
19     }
20 }
```

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

## 14. Factory

```
참조 4개 | Changed by t1gh0903@gmail.com on 2024년 10월 17일 목요일
4  public abstract class Factory
5  {
    참조 4개 | Added by t1gh0903@gmail.com on 2024년 9월 25일 수요일
6      public abstract IProjectile GetProduct(float damage, Vector3 position, Action<float> targetHealth, Vector3 targetPos);
7
8  }
```

GetProduct : 구체적인 팩토리 클래스에서 오버라이드되어, 지정된 위치에 투사체를 생성

## 15. ConcreteArrowFactory, ConcreteFireBallFactory

```
참조 1개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
4 public class ConcreteFireBallFactory : Factory
5 {
6     참조 3개 | Changed by tigh0903@gmail.com on 2024년 10월 17일 목요일
7     public override IProjectile GetProduct(float damage, Vector3 position, Action<float> targetHealth, Vector3 targetPos)
8     {
9         GameObject instance = PoolManager.Instance.Get(PoolEnum.Projectile, "FireBall", position, Quaternion.identity);
10        FireBall fireBall = instance.GetComponent<FireBall>();
11        fireBall.Initialize(damage, targetHealth);
12        fireBall.Shoot(targetPos);
13        return fireBall;
14    }
15 }
```

```
참조 1개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
4 public class ConcreteArrowFactory : Factory
5 {
6     참조 3개 | Changed locally
7     public override IProjectile GetProduct(float damage, Vector3 position, Action<float> targetHealth, Vector3 targetPos)
8     {
9         GameObject instance = PoolManager.Instance.Get(PoolEnum.Projectile, "Arrow01", position, Quaternion.identity);
10        Arrow arrow = instance.GetComponent<Arrow>();
11        arrow.Initialize(damage, targetHealth);
12        arrow.Shoot(targetPos);
13        return arrow;
14    }
15 }
```

GetProduct : PoolManager로부터 오브젝트를 가져와 초기화하고 targetPos 방향으로 발사



## 16. Arrow

```
5 public class Arrow : Projectile, IPProjectile
6 {
7     public float Damage { get; set; }
8     public void Initialize(float unitDamage, Action<float> targetHealth)
9     {
10         Damage = unitDamage;
11         startPos = transform.position;
12         this.targetHealth = targetHealth;
13     }
14
15     public void Shoot(Vector3 targetPosition)
16     {
17         dirPos = targetPosition - startPos;
18         dirPos.Normalize();
19
20         rigid.AddForce(dirPos * speed, ForceMode2D.Impulse);
21     }
22
23     private void OnTriggerEnter2D(Collider2D collision)
24     {
25         if (collision.CompareTag("Enemy"))
26         {
27             targetHealth.Invoke(Damage);
28             gameObject.SetActive(false);
29         }
30     }
31 }
32
```

Initialize : 화살의 데미지와 시작 위치를 초기화하며, 타겟의 체력을 조정하기 위한 **targetHealth** 델리게이트를 설정  
Shoot : **targetPosition** 방향으로 발사  
OnTriggerEnter2D : **targetHealth**를 호출하여 적의 체력을 감소

## 17. FireBall

```
5 public class FireBall : Projectile, IProjectile
6 {
7     참조 4개 | Changed by tigh0903@gmail.com on 2024년 10월 17일 목요일
8     [SerializeField] public float Damage { get; set; }
9
10    참조 2개 | Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
11    public void Initialize(float unitDamage, Action<float> targetHealth)
12    {
13        Damage = unitDamage;
14        startPos = transform.position;
15        this.targetHealth = targetHealth;
16    }
17    참조 2개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
18    public void Shoot(Vector3 targetPosition)
19    {
20        dirPos = targetPosition - startPos;
21        dirPos.Normalize();
22
23        rigid.AddForce(dirPos * speed, ForceMode2D.Impulse);
24    }
25    Unity 메시지 | 참조 0개 | Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
26    private void OnTriggerEnter2D(Collider2D collision)
27    {
28        if (collision.CompareTag("Enemy"))
29        {
30            targetHealth.Invoke(Damage);
31            GameObject explode = PoolManager.Instance.Get(PoolEnum.HitScan, "Explode", transform.position, Quaternion.identity);
32            SoundManager.Instance.PlaySFX(SoundType.Explode);
33            explode.GetComponent<Explode>().Damage = Mathf.Round(Damage/2);
34            gameObject.SetActive(false);
35        }
36    }
37 }
```

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  {
6      * Unity 메시지|참조 0개|Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
7      private void OnEnable()
8      {
9          StartCoroutine(SetActiveFalse());
10     }
11     * Unity 메시지|참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 25일 금요일
12     public override void OnTriggerEnter2D(Collider2D collision)
13     {
14         if (collision.CompareTag("Enemy"))
15         {
16             collision.GetComponent<IUnitHealth>().GetHit(Damage);
17             canAttack = false;
18         }
19     }
20 }
```

2차 폭발 오브젝트

적과 충돌시 충돌한적에게 피해를 입힘

의도하지않은 중복피해를 방지하고자 canAttack = false;

## 19. HitScan

```
5 public abstract class HitScan : MonoBehaviour
6 {
7     [field : SerializeField] public float Damage { get; set; }
8     protected Animator animator;
9     protected bool canAttack;
10    public virtual void Awake()
11    {
12        animator = GetComponent<Animator>();
13    }
14
15    public abstract void OnTriggerEnter2D(Collider2D collision);
16
17    protected IEnumerator SetActiveFalse()
18    {
19        while (true)
20        {
21            AnimatorStateInfo stateInfo = animator.GetCurrentAnimatorStateInfo(0);
22
23            if (stateInfo.normalizedTime >= 1f)
24            {
25                gameObject.SetActive(false);
26                break;
27            }
28            yield return null;
29        }
30    }
31 }
32
```

즉시 시전되는 오브젝트의 추상클래스

SetActiveFalse : 애니메이션이 끝나면 비활성화

## 20. HealHitScan

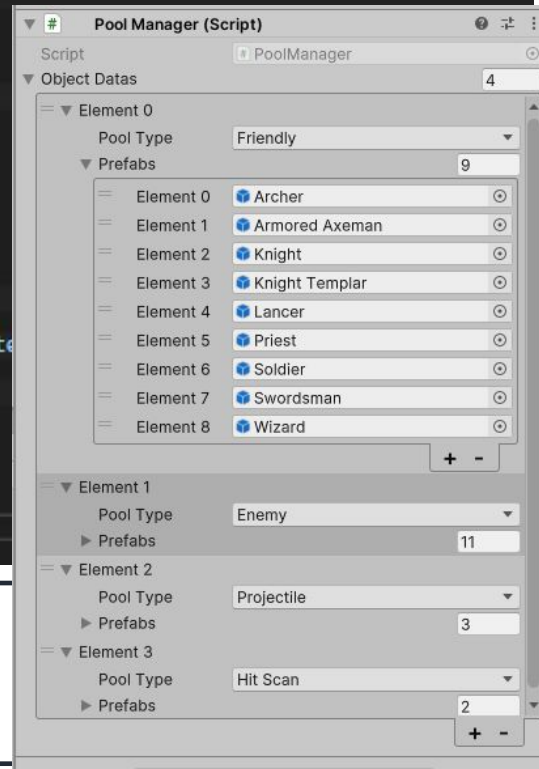
```
4 [RequireComponent(typeof(CircleCollider2D))]  
    * Unity 스크립트(자산 참조 1개)|참조 0개|Added by tlgh0903@gmail.com on 2024년 9월 25일 수요일  
5 public class HealHitscan : HitScan  
6 {  
    * Unity 메시지|참조 1개|Changed by tlgh0903@gmail.com on 2024년 10월 29일 화요일  
7     public override void OnTriggerEnter2D(Collider2D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

아군과 접촉시 애니메이션 재생후 비활성화

## 21. PoolManager

```
4 >public enum PoolEnum{...}
11
12 [System.Serializable]
13 >public class PoolType{...}
21
22 public class PoolManager : Singleton<PoolManager>
23 {
24     [SerializeField] List<PoolType> objectDatas = new List<PoolType>();
25
26     public override void Awake()...
46     public GameObject Get(PoolEnum prefabTypes, string name, Vector3 startPos, Quaternion quat)
73     public void DeactivateAllChildren()...
80
81
82
83
84
```

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



## 23. SoundManager

```

4 //사운드이름 enum타입으로 선언
  참조 18개 | Added by tlg0903@gmail.com on 2024년 10월 17일 목요일
5 >public enum SoundType{...
  참조 3개 | Added by tlg0903@gmail.com on 2024년 10월 17일 목요일
19 >public enum BGMTType{...
25 [RequireComponent(typeof(AudioSource))]
  Unity 스크립트(자산 참조 3개) | 참조 25개 | Changed by tlg0903@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
37     Unity 메시지 | 참조 0개 | Added by tlg0903@gmail.com on 2024년 10월 25일 금요일
    private void Start()...
38     Unity 메시지 | 참조 0개 | Changed by tlg0903@gmail.com on 2024년 10월 25일 금요일
    private void Update()
43 {
44     SoundUI();
45     BGMVolume();
46     SFXVolume();
47 }
48
49     참조 1개 | Changed by tlg0903@gmail.com on 2024년 10월 25일 금요일
    private void SoundUI()...
50     참조 17개 | Added by tlg0903@gmail.com on 2024년 10월 17일 목요일
    public void PlaySFX(SoundType sound){...
51     참조 2개 | Added by tlg0903@gmail.com on 2024년 10월 17일 목요일
52     public void PlayBGM(BGMTType sound){...
53     참조 1개 | Added by tlg0903@gmail.com on 2024년 10월 17일 목요일
54     void BGMVolume()...
55     참조 1개 | Added by tlg0903@gmail.com on 2024년 10월 17일 목요일
56     void SFXVolume()...
57 }
80

```

SoundType :

효과음을 구분하는 데 사용

BGMTType:

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:

BGMTType을 받아 AudioSource에 해당 배경음악을 재생하고 반복 재생

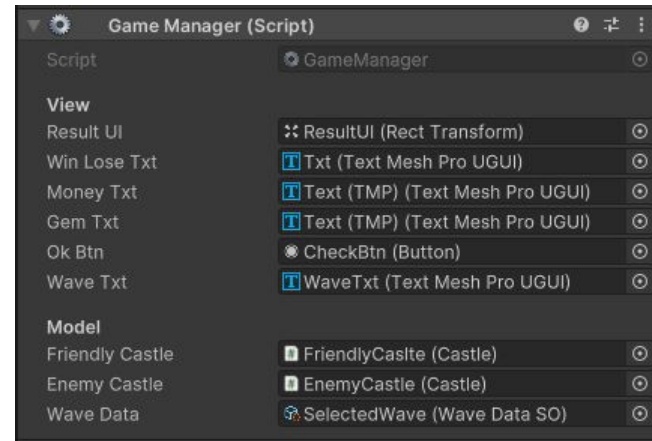


## 22. GameManager

```

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
24     참조 3개
25     public float Money { get; set; }
26     참조 3개
27     public float Gem { get; set; }
28
29     Unity 메시지 | 참조 0개
30     private void Awake()...
31     참조 3개
32     public void OnCastleDestroyed(Castle castle)...
33     참조 1개
34     private void Win()...
35     참조 1개
36     private void Lose()...
37
38 }

```



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



## 24. Castle

```

6  public class Castle : MonoBehaviour, IUnitHealth
7  {
8      public CastleData castleData;
9      참조 13개 | Added by tlg0903@gmail.com on 2024년 10월 11일 금요일
10     public float Health { get; set; }
11     Slider healthBar;
12     TMP_Text healthTxt;
13
14     public Action<Castle> CastleDestroy;
15     참조 0개 | Added by tlg0903@gmail.com on 2024년 10월 11일 금요일
16     public void Awake()
17     {
18     }
19     참조 0개 | Changed by tlg0903@gmail.com on 2024년 10월 29일 화요일
20     private void OnEnable()
21     {
22     }
23     참조 3개 | Added by tlg0903@gmail.com on 2024년 10월 11일 금요일
24     public void Init()
25     {
26     }
27     참조 3개 | Changed locally
28     public void GetHit(float damage)
29     {
30     }
31     Health -= damage;
32     Health = Mathf.Clamp(Health, 0, castleData.Health);
33     UpdateHealthBar();
34     if (IsDead())
35     {
36         Die();
37     }
38     참조 2개 | Added by tlg0903@gmail.com on 2024년 10월 11일 금요일
39     public bool IsMaxHealth()
40     {
41     }
42     참조 2개 | Added by tlg0903@gmail.com on 2024년 10월 11일 금요일
43     public void GetHeal(float amount)
44     {
45     }
46     참조 2개 | Added by tlg0903@gmail.com on 2024년 10월 11일 금요일
47     public bool IsDead()
48     {
49     }
50     참조 3개 | Moved up 2 positions by tlg0903@gmail.com on 2024년 10월 29일 화요일
51     private void UpdateHealthBar()
52     {
53     }
54     참조 2개 | Moved up 2 positions by tlg0903@gmail.com on 2024년 10월 29일 화요일
55     public void Die()
56     {
57     }
58
59     CastleDestroy?.Invoke(this);
60     Debug.Log(castleData.UnitName + " 가 파괴되었습니다");
61     CastleDestroy = null;
62 }
63

```

체력이 0이 되면 CastleDestroy 액션에 등록된 메서드가 호출

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

## 25. BaseMenu

```
4 public class BaseMenu : MonoBehaviour
5 {
6     [SerializeField] protected GameObject ui;
7     [SerializeField] protected Button openButton;
8     public virtual void Awake()
9     {
10         openButton.onClick.AddListener(() =>
11         {
12             ui.SetActive(true);
13             SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
14         });
15     }
16     public virtual void HideUI()
17     {
18         SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
19         ui.SetActive(false);
20     }
21 }
```

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;
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
37     void Click(string buttonName)...
38
39     void ShowClickedInfo(UnitData clickedUnit)...
40
41     void ShowClickedInfo(UnitData clickedUnit)...
42
43     void ShowClickedInfo(UnitData clickedUnit)...
44
45     void ShowClickedInfo(UnitData clickedUnit)...
46
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

```

5  Unity 스크립트(자산 참조 1개) 참조 0개 | Changed and Renamed from UnitSelectUI to UnitSelectMenu locally
6  public class UnitSelectMenu : BaseMenu
7  {
8      [Header("View")]
9      [SerializeField] GameObject grid;
10     [SerializeField] GameObject selected;
11     [SerializeField] Button exitBtn;
12
13     [Header("Model")]
14     [SerializeField] DataContainer unitDatasContainer;
15     [SerializeField] GameObject buttonPrefab;
16
17     Image[] selectedImage;
18     HashSet<string> selectedUnitNamesSet;
19     * Unity 메시지 | 참조 4개 | Changed by tigh0903@gmail.com on 2024년 10월 25일 금요일
20     public override void Awake()
21     {
22         base.Awake();
23         SelectedUI();
24         selectedUnitNamesSet = new HashSet<string>();
25
26         UnitsUI();
27         exitBtn.onClick.AddListener(() =>
28         {
29             SoundManager.Instance.PlaySFX(SoundType.ClickBtn);
30             HideUI();
31         });
32     }
33     참조 1개 | Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
34     private void SelectedUI()...
35     참조 1개 | Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
36     private void UnitsUI()...
37     참조 1개 | Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
38     void SortSelected(UnitData unitData)...
39     참조 1개 | Changed by tigh0903@gmail.com on 2024년 10월 29일 화요일
40     void DeselectUnit(Button button, int index)...
41     참조 3개 | Added by tigh0903@gmail.com on 2024년 9월 25일 수요일
42     public override void HideUI()...
43 }

```

**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 LvUIPrefab;
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[] researches;
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", "충량");
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(BGType.Prepare);
50     }
51     void CreateButton()...
52     void CreateButtonData(int index, UnitData unitData)...
53     void CreateButtonData(int index, string unitName, int upgradeLength, string unitKorName)...
54
55     void ButtonDesc(int researchIndex, IUpgradeableUnit unit, string growthTxt1, string growthTxt2 = null)...
56     void HighLvBtn(Button button)...
57     void UpgradedBtn(Button button, Image line)...
58     void UpgradeUI(IUpgradeableUnit unitData, int index, Button button, Button nextbutton, Image line, string growthTxt1, string growthTxt2=null)...
59 }

```

CreateButton :

성, 자원, 유닛업그레이드 버튼생성

ButtonDesc :

업그레이드 버튼의 내용 초기화 및 상태 설정

HighLvBtn :

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
6 public class WaveSelectMenu : BaseMenu
7 {
8     [Header("View")]
9     [SerializeField] Transform waveButtonsTransform;
10
11     [Header("Model")]
12     public WaveDataSO[] waveDatas;
13     [SerializeField] WaveDataSO selectedWave;
14     [SerializeField] GameObject buttonPrefab;
15
16     Unity 메시지|참조 4개|Changed by tish0903@gmail.com on 2024년 10월 25일 금요일
17     public override void Awake()
18     {
19         base.Awake();
20         CreateWaveButtons();
21         int waveindex = 0;
22         WaveClearData waveClearData = MyUtil.JsonLoad<WaveClearData>(MyUtil.JsonFileName.WaveClear);
23         waveindex = (waveClearData != null) ? waveClearData.WaveClear : 0;
24         int waveClear = waveindex + 1;
25         for (int i = 0; i < waveClear; i++)
26         {
27             waveButtonsTransform.GetChild(i).GetComponentInChildren<Button>().interactable = true;
28         }
29
30     참조 1개|Changed locally
31     private void CreateWaveButtons()...
32     참조 1개|Changed locally
49     private void SelectWave(int index)...
55     참조 1개|Changed by tish0903@gmail.com on 2024년 10월 25일 금요일
76     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

```
6  Unity 스크립트 | 참조 1개 | Added by t1gh0903@gmail.com on 2024년 10월 11일 금요일
7  public class GenerateWaveDataSO : EditorWindow
8  {
9      private int waveIndex = 1;
10     private int rewardMoney;
11     private int rewardGem;
12     private int enemyCastleHealth;
13
14     private List<EnemyData> enemies = new List<EnemyData>();
15
16     참조 3개 | Added by t1gh0903@gmail.com on 2024년 10월 11일 금요일
17     private class EnemyData
18     {
19         public EnemyName name;
20         public float multiplier;
21         public int spawnCount;
22         public int spawnTime;
23         public int spawnDelay;
24     }
25
26     [MenuItem("MyUtilities/Generate Wave SO")]
27     참조 0개 | Added by t1gh0903@gmail.com on 2024년 10월 11일 금요일
28     public static void ShowWindow()...
29     Unity 메시지 | 참조 0개 | Added by t1gh0903@gmail.com on 2024년 10월 11일 금요일
30     private void OnGUI()...
31     참조 1개 | Changed by t1gh0903@gmail.com on 2024년 10월 25일 금요일
85     private void GenerateSO()...
114 }
```

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

Wave Data Generator

Wave Data

Add Enemy

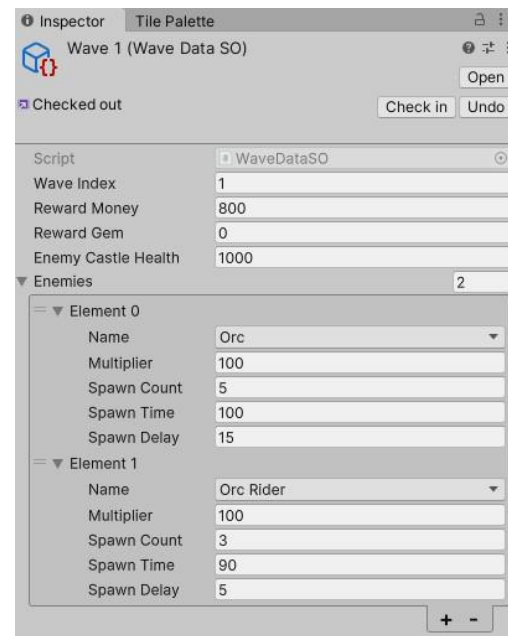
웨이브	보상 골드	보상 젬	적 성체력
1	0	0	0

적 캐릭터	배율(%)	출현수	출현 시기(성체력(%))	출현 딜레이(sec)
Orc	0	0	0	0
Orc	0	0	0	0

Generate SO

## 31. WaveDataSO

```
4
5 public class WaveDataSO : ScriptableObject
6 {
7     public int waveIndex;
8     public int rewardMoney;
9     public int rewardGem;
10    public float enemyCastleHealth;
11    public List<EnemyData> enemies = new List<EnemyData>();
12
13    [System.Serializable]
14    참조 7개 | Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
15    public class EnemyData
16    {
17        public EnemyName name;
18        public float multiplier;
19        public int spawnCount;
20        public int spawnTime;
21        public int spawnDelay;
22
23        참조 3개 | Added by tlgh0903@gmail.com on 2024년 10월 25일 금요일
24    }
25
26    > public enum EnemyName {
```



생성한 웨이브의 스크립트와 인스펙터창



## 32. EnemySpawn

```

6 public class EnemySpawn : MonoBehaviour
7 {
8     public WaveDataSO waveData;
9
10    public List<UnitData> unitDatas = new List<UnitData>();
11    float[] spawnTimer;
12    int[] curSpawnCount;
13    [SerializeField] Castle enemycastle;
14    int tempCount;
15    // Unity 메시지 | 참조 0개 | Changed and Moved down 4 positions and Renamed from Timer to Awake by t1gh0003@gmail.com on 2024년 10월 11일 월요일
16    private void Awake()
17    {
18        enemycastle.castleData.Health = waveData.enemyCastleHealth;
19        spawnTimer = new float[waveData.enemies.Count];
20        curSpawnCount = new int[waveData.enemies.Count];
21
22        // Unity 메시지 | 참조 0개 | Changed and Moved down 4 positions and Renamed from Timer to Awake by t1gh0003@gmail.com on 2024년 10월 11일 월요일
23        private void Start()
24        {
25            for (int i = 0; i < waveData.enemies.Count; i++)
26            {
27                GameObject unit = PoolManager.Instance.GetPoolEnum Enemy, waveData.enemies[i].name.ToString(), MyUtil.SpawnRandomPos(47f, 54f, -4f, -6.5f), Quaternion.identity);
28                UnitData unitData = unit.GetComponent<UnitData>().UnitData;
29
30                CopyUnitData(unitData, unitDatas[i]);
31                EnemyMultiplier unitDatas[i], waveData.enemies[i].multiplier;
32                unit.GetComponent<UnitData>().UnitData = unitDatas[i];
33                unit.SetActive(false);
34            }
35
36            // Unity 메시지 | 참조 0개 | Changed by t1gh0003@gmail.com on 2024년 10월 17일 목요일
37            private void Update()
38            {
39                for (int i = 0; i < waveData.enemies.Count; i++)
40                {
41                    Timer(i, waveData.enemies[i].spawnCount, waveData.enemies[i].spawnDelay);
42                }
43
44                참조 1개 | Changed and Moved down 5 positions and Renamed from UnitUpgrade to Update by t1gh0003@gmail.com on 2024년 10월 11일 월요일
45                void Timer(int index, int spawnCount, float spawnDelay)
46                {
47                    참조 1개 | Added by t1gh0003@gmail.com on 2024년 10월 11일 월요일
48                    void Spawn(int index)
49                    {
50                        참조 1개 | Changed by t1gh0003@gmail.com on 2024년 10월 29일 화요일
51                        private void CopyUnitData(UnitData origin, UnitData copy)
52                        {
53                            참조 1개 | Changed by t1gh0003@gmail.com on 2024년 10월 25일 금요일
54                            void EnemyMultiplier(UnitData unitData, float multiplier)
55                        }
56                    }
57                }
58            }
59        }
60    }
61 }

```

CreateWaveButtons :

웨이브 선택 버튼을 생성하고 각 버튼에  
웨이브 데이터 연결

SelectWave :

Battle씬으로 이동

CopyWaveData : 선택된 웨이브 데이터를  
복사

## 33. UnitSpawn

```

5  Unity 스크립트(자산 참조 1개) | 참조 0개 | Renamed from UnitTraining to UnitSpawn locally
6  public class UnitSpawn : MonoBehaviour
7  {
8      [Header("View")]
9      [SerializeField] Button[] buttons;
10     [SerializeField] TMP_Text supplyTxt;
11     [SerializeField] Button supplyUpBtn;
12
13     SelectedUnit unitNames;
14     UnitData[] selectedUnits;
15
16     [Header("Model")]
17     [SerializeField] SupplyResources supplyResourcesSO;
18     float curResources;
19     float curSpeed;
20     float curCapacity;
21     int curCapacityLvl = 1;
22     private void Awake()
23     {
24         JsonLoad();
25         Init();
26         ResourcesInit();
27     }
28     private void Update()
29     {
30         ResourceSupply();
31         for (int i = 0; i < buttons.Length; i++)
32         {
33             buttons[i].interactable = curResources >= selectedUnits[i].Cost;
34         }
35     }
36     private void ResourcesInit()
37     {
38     }
39     private void ResourceSupply()
40     {
41     }
42     private void JsonLoad()
43     {
44     }
45     private void Init()
46     {
47     }
48     void CreateUnit(int i)
49     {
50     }
51     public bool DecreaseResources(float amount)
52     {
53     }
54 }

```

### 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  
초기화하고 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

```

5  참조 43개 | Added by t1gh0903@gmail.com on 2024년 8월 16일 금요일
6  ~~~~~
7  public static class MyUtil
8  {
9      참조 3개 | Added by t1gh0903@gmail.com on 2024년 8월 16일 금요일
10     public static float MoveSpeed(float moveSpeed) {...}
11     참조 3개 | Added by t1gh0903@gmail.com on 2024년 8월 26일 수요일
12     public static Vector3 SpawnRandomPos(float minX, float maxX, float minY, float maxY) {...}
13     참조 5개 | Added by t1gh0903@gmail.com on 2024년 8월 26일 수요일
14     public static void JsonSave<T>(T data, JsonFileName fileName)
15     {
16         Debug.Log(data);
17         string json = JsonUtility.ToJson(data);
18         Debug.Log("JsonSave : " + json);
19         File.WriteAllText(Application.persistentDataPath + $"/{fileName}.json", json);
20         Debug.Log("Application.persistentDataPath : " + Application.persistentDataPath);
21     }
22     참조 5개 | Changed by t1gh0903@gmail.com on 2024년 10월 11일 금요일
23     public static T JsonLoad<T>(JsonFileName fileName)
24     {
25         string path = Application.persistentDataPath + $"/{fileName}.json";
26         if (File.Exists(path))
27         {
28             string json = File.ReadAllText(path);
29             T data = JsonUtility.FromJson<T>(json);
30             Debug.Log("JsonLoad : " + json);
31             return data;
32         }
33         else
34         {
35             Debug.Log("Statue_Save : null");
36             return default(T);
37         }
38     }
39     참조 17개 | Changed by t1gh0903@gmail.com on 2024년 10월 28일 화요일
40     public static string StringBuilderTxt(params object[] values)
41     {
42         StringBuilder sb = new();
43         sb.Clear();
44         foreach (var value in values)
45         {
46             sb.Append(value);
47         }
48         return sb.ToString();
49     }
50     참조 12개 | Changed by t1gh0903@gmail.com on 2024년 10월 17일 목요일
51     public enum JsonFileName {...}
52 }

```

**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")]
5  # Unity 스크립트 | 참조 36개
6  public class UnitData : ScriptableObject, IMove, IDamage, IUpgradeableUnit
7  {
8      참조 6개
9      [SerializeField] public Sprite Image { get; set; }
10     참조 8개
11     [SerializeField] public string UnitKorName { get; set; }
12     참조 14개
13     [SerializeField] public string UnitName { get; set; }
14     참조 10개
15     [SerializeField] public int Level { get; set; }
16     참조 20개
17     [SerializeField] public float Health { get; set; }
18     참조 16개
19     [SerializeField] public float Damage { get; set; }
20     참조 6개
21     [SerializeField] public float AttackSpeed { get; set; }
22     참조 8개
23     [SerializeField] public float MoveSpeed { get; set; }
24     참조 7개
25     [SerializeField] public float DetectRange { get; set; }
26     참조 6개
27     [SerializeField] public float AttackRange { get; set; }
28     참조 7개
29     [SerializeField] public int Cost { get; set; }
30     참조 6개
31     [SerializeField] public MoveDir MoveDir { get; set; }
32     참조 14개
33     [SerializeField] public UnitUpgradeDataSO unitUpgradeData { get; set; }
34     참조 3개
35     public float MaxHealth { get; set; }
36     참조 3개
37     public float MaxDamage { get; set; }
38     참조 1개
39     public void MaxHealthNDamage()...
40     참조 2개
41     public void LevelUP(float health, float damage)...
42     참조 2개
43     public float MoneyCost(int index) => unitUpgradeData.unitUpgrade[index].moneyCost;
44     참조 2개
45     public float GemCost(int index) => unitUpgradeData.unitUpgrade[index].gemCost;
46     참조 2개
47     public float CurGrowth1() => Health;
48     참조 2개
49     public float CurGrowth2() => Damage;
50     참조 2개
51     public float GetGrowth1(int index) => unitUpgradeData.unitUpgrade[index].growth1;
52     참조 2개
53     public float GetGrowth2(int index) => unitUpgradeData.unitUpgrade[index].growth2;

```

Inspector Tile Palette

Archer (Unit Data)

Open

Checkout

Script	UnitData
Image	Archer_Frame_0
Unit Kor Name	궁수
Unit Name	Archer
Level	1
Health	300
Damage	20
Attack Speed	3.5
Move Speed	11
Detect Range	8
Attack Range	5
Cost	150
Move Dir	Move Right
unit Upgrade Data	Archer (Unit Upgrade Data SO)

깃  
허브

<https://github.com/SlHO0903/DefenseGame>

유튜브  
링크

<https://youtu.be/fiwn0BxfbkA>

감사합니다

---