

뽕뽕 특공대

탕탕특공대 게임 모작

Shooting Game with Unity

Name 김나영
Tel 010-3902-6074
E-mail nayeong0121@gmail.com

목차

1. 게임 개요
2. 게임 화면
3. 구조도
4. 기능 구현 및 주요 기능 구현 함수

1. 게임 개요

- 언어 : C#
- 도구 : Unity
- 플랫폼 : Windows
- [게임 영상 링크](#)
- [깃 허브 링크](#)
- 개발 기간 : 2025.02 ~ 2025.03



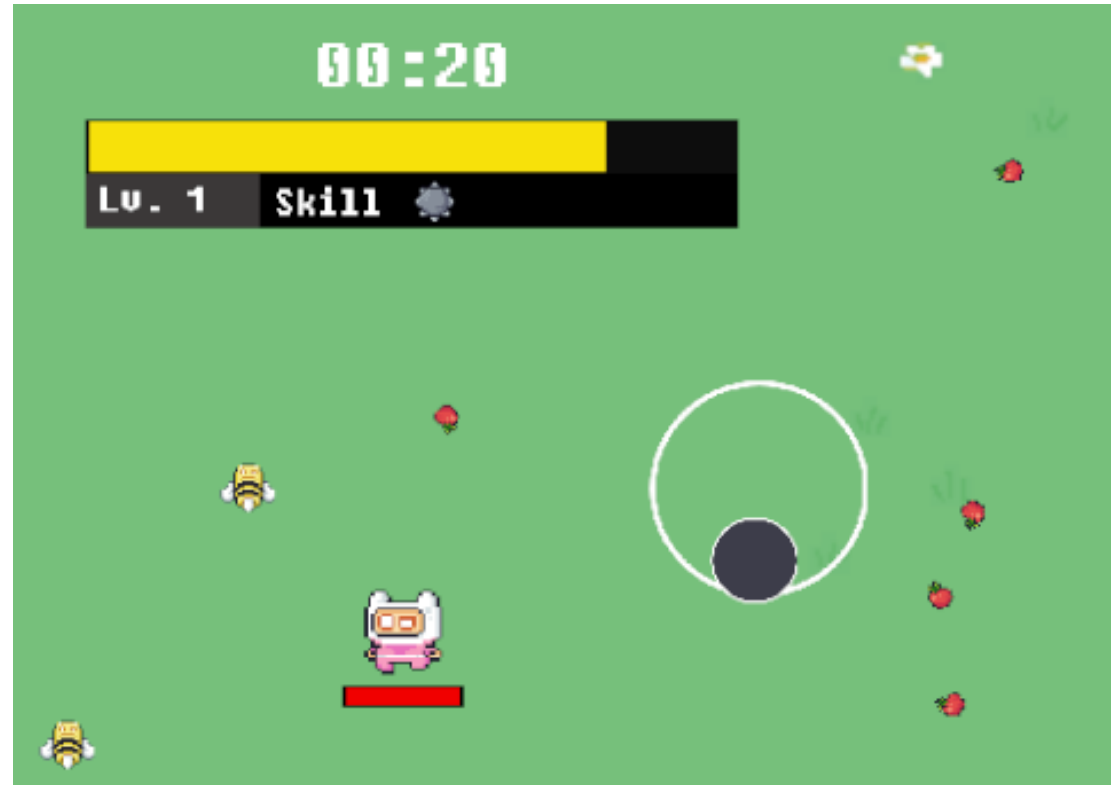
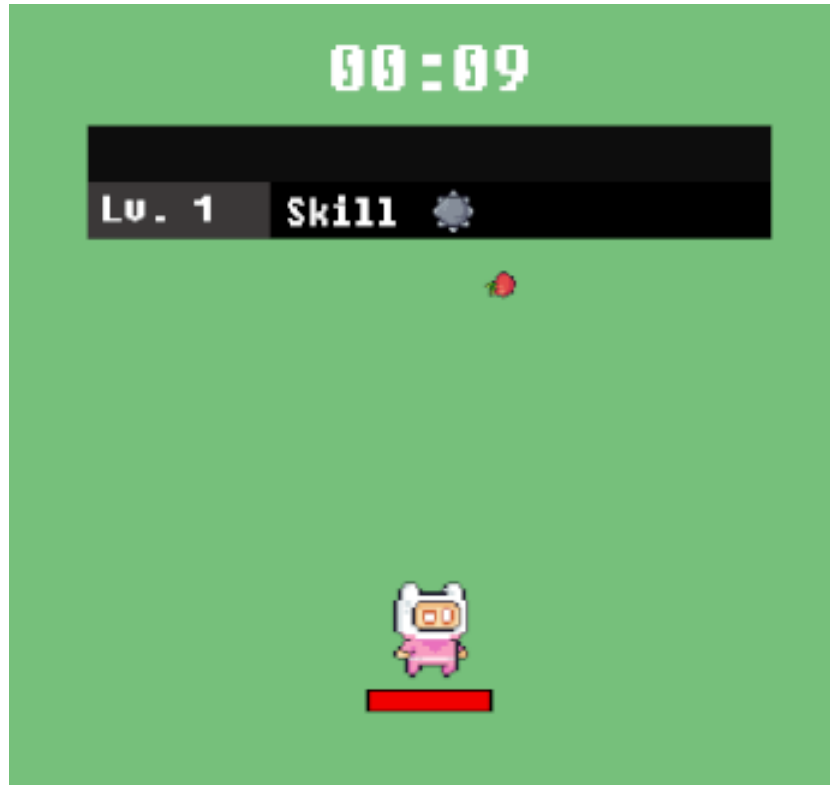
- 빵빵 특공대는 Habby의 탕탕 특공대를 모작한 슈팅 게임입니다.
- 몰려오는 몬스터를 피하며 과일을 먹어 스킬을 획득해 몬스터를 없애는 것이 목표입니다.

2. 게임 화면 – 메인 화면



메인 화면에서 화면을 클릭하면 게임 화면으로 넘어가게 됩니다.

2. 게임 화면 – 인게임 화면



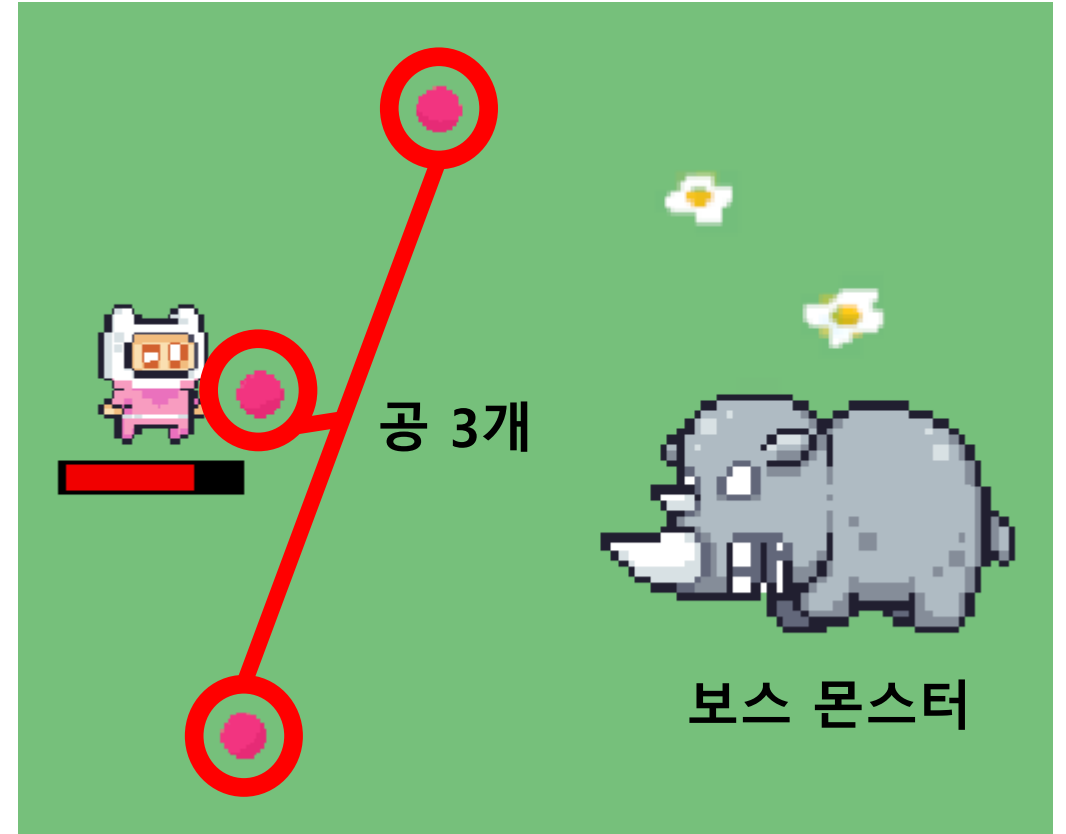
게임 초기 화면에서 레벨은 1이며, 플레이어의 스킬은 스파이크 공 한 개입니다.
화면을 클릭해서 나타나는 조이스틱으로 플레이어가 움직이고, 과일을 먹으면 과일 게이지가 차오릅니다.

2. 게임 화면 - 스킬 선택 화면



과일 게이지를 다 채우면 나타나는 스킬 선택 화면입니다.

2. 게임 화면 - 보스 등장 화면



보스가 등장하기 전 플레이어 주위로 철창이 생기며, 몇 초 뒤에 보스가 등장합니다.
보스는 일정 시간마다 플레이어를 향해 돌진하고, 3개의 총알을 발사합니다.
보스가 죽거나 플레이어가 죽으면 게임이 끝나게 됩니다.

2. 게임 화면 – 일시 정지 화면



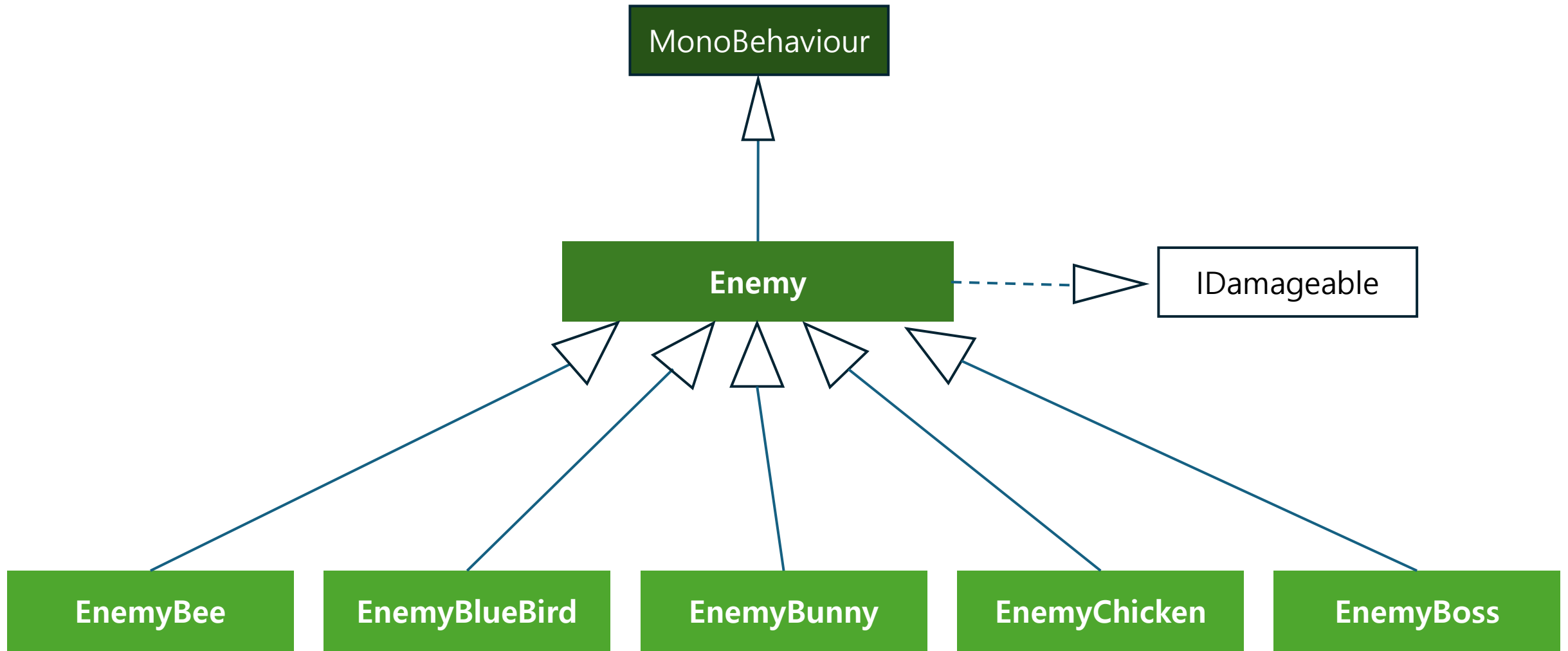
게임 화면에서 우측 상단의 일시 정지 버튼을 클릭하면 나오는 화면입니다.

2. 게임 오버 화면

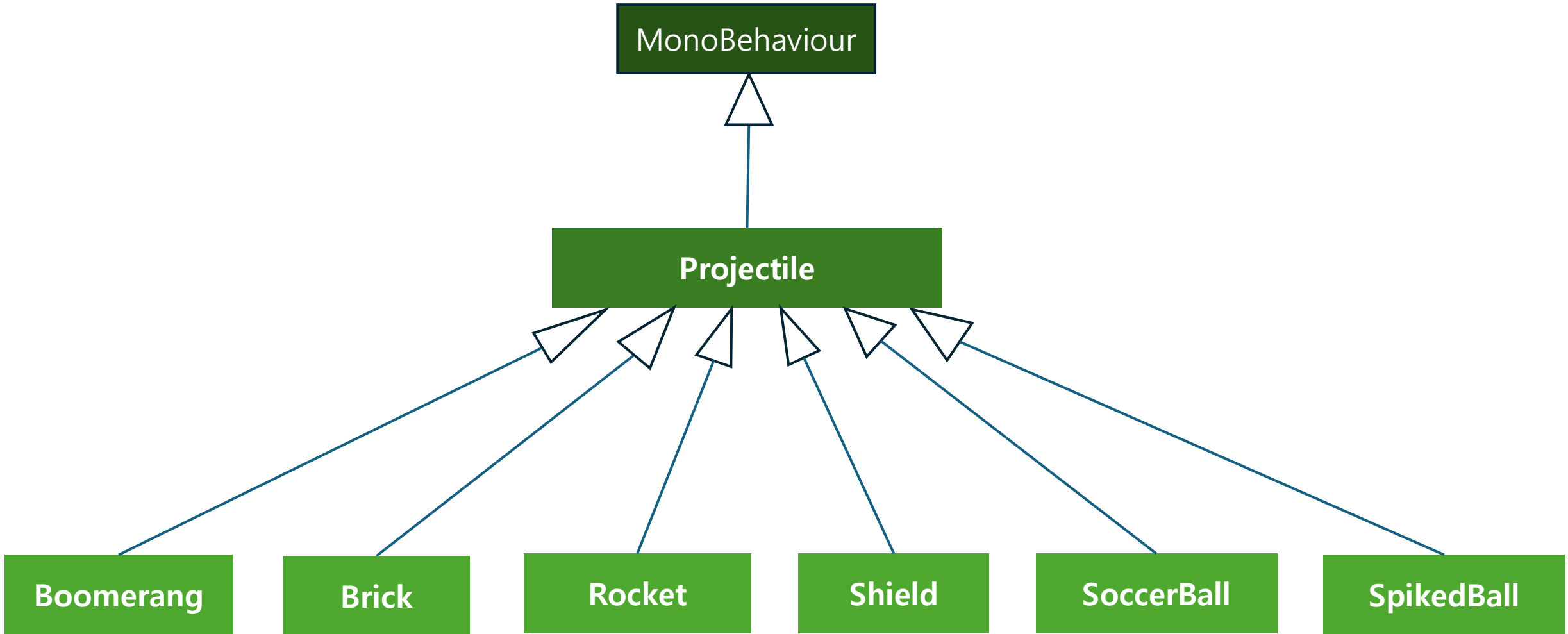


플레이어가 죽거나 보스가 죽으면 나오는 게임 오버 화면입니다.

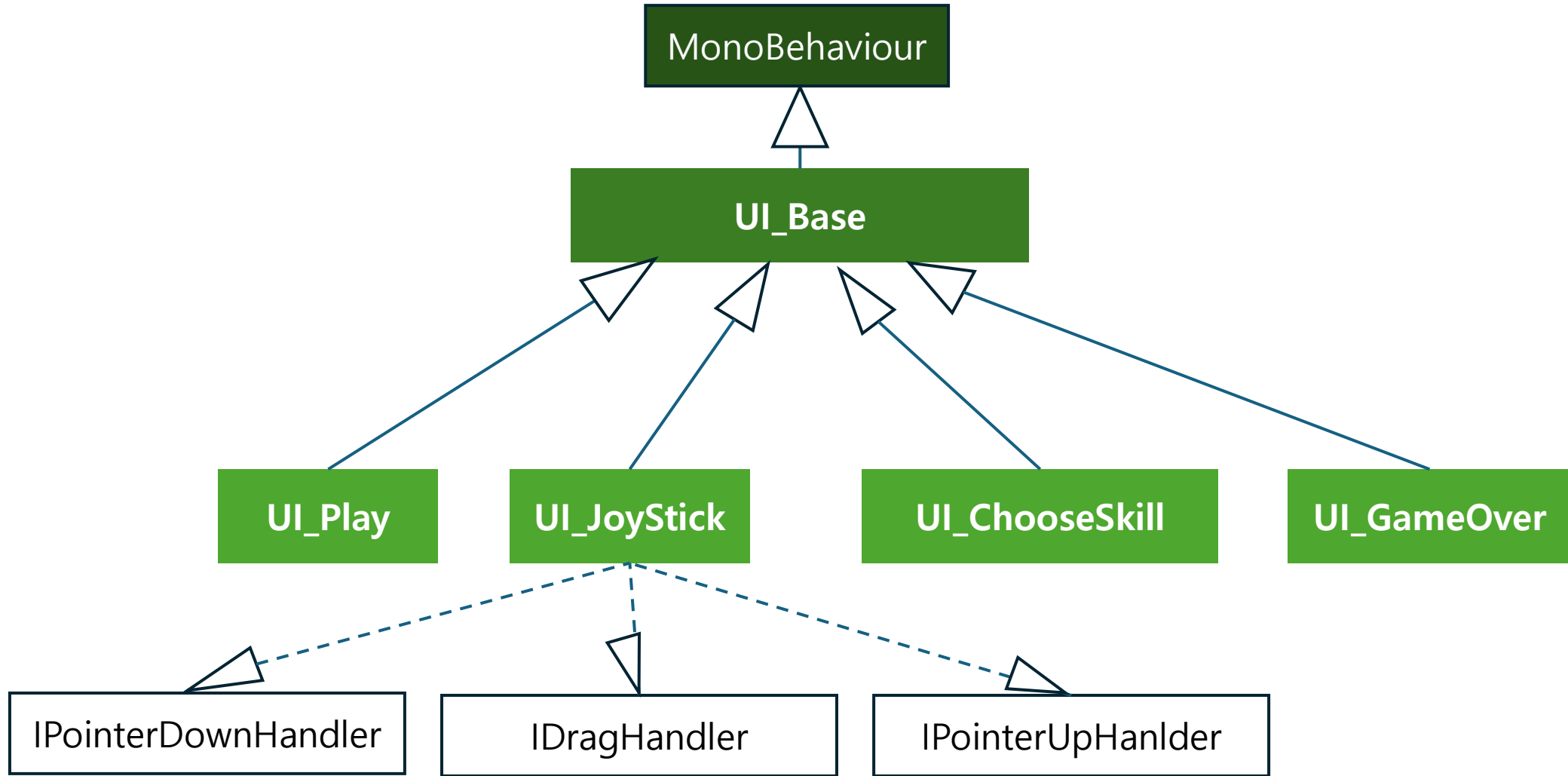
3. 구조도 - 적 관련 상속 구조



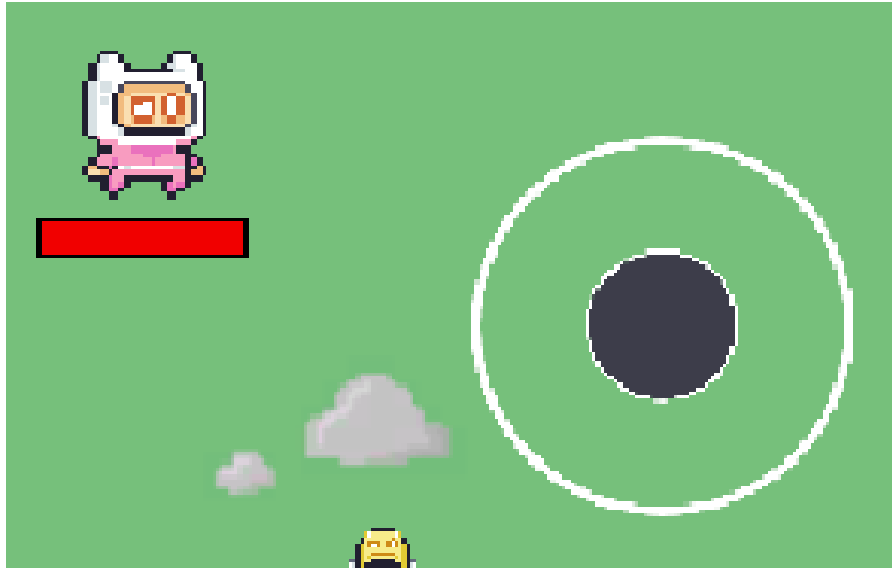
3. 구조도 - 무기 관련 상속 구조



3. 구조도 - UI 상속 구조



4. 기능 구현 - 조이스틱 조작



화면에 마우스를 클릭하면 나타나는 조이스틱



마우스 방향을 오른쪽 위로 향하게 하면
플레이어는 오른쪽 위로 움직인다.

화면에 마우스를 클릭하면 플레이어를 움직일 수 있는 조이스틱이 나타나고,
마우스 방향대로 플레이어가 움직이게 됩니다.
플레이어가 움직이는 방향에 따라 카메라도 따라가게 됩니다.

4. 주요 기능 구현 함수 - 조이스틱 조작

```
public void OnPointerDown(PointerEventData eventData)
{
    SetActiveJoystick(true);
    _touchPos = Input.mousePosition;
    Joystick.transform.position = Input.mousePosition;
    Handler.transform.position = Input.mousePosition;
}
```

마우스 클릭 시 실행되는 함수

```
//마우스를 드래그 했을 때 실행되는 함수
참조 0개
public void OnDrag(PointerEventData eventData)
{
    Vector2 dragPos = eventData.position;
    _moveDir = (dragPos - _touchPos).normalized;
    float distance = (dragPos - _touchPos).sqrMagnitude;

    Vector3 newPos = (distance < _radius) ? _touchPos + (_moveDir * distance) :
        _touchPos + (_moveDir * _radius);
    Handler.transform.position = newPos;
    GameManager.Instance.MoveDir = _moveDir;
}
```

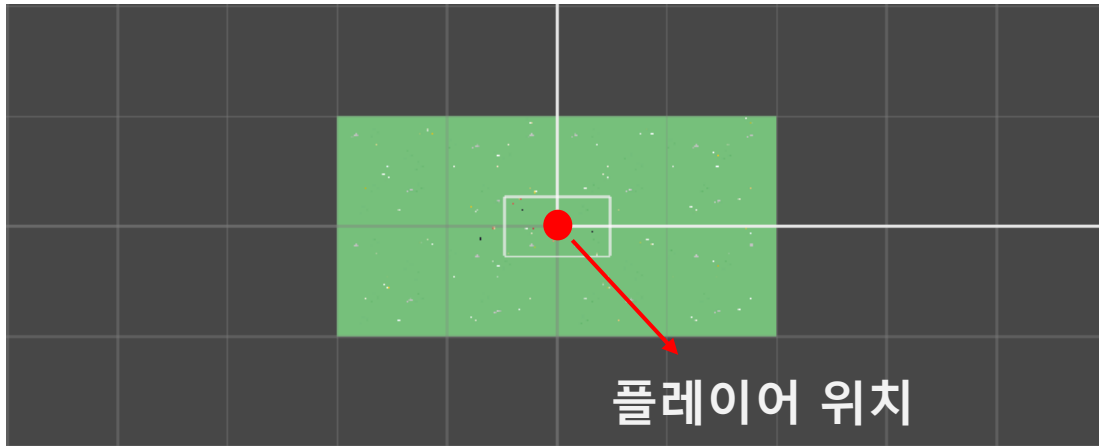
마우스 드래그 시 실행되는 함수

4. 주요 기능 구현 함수 – 조이스틱 조작

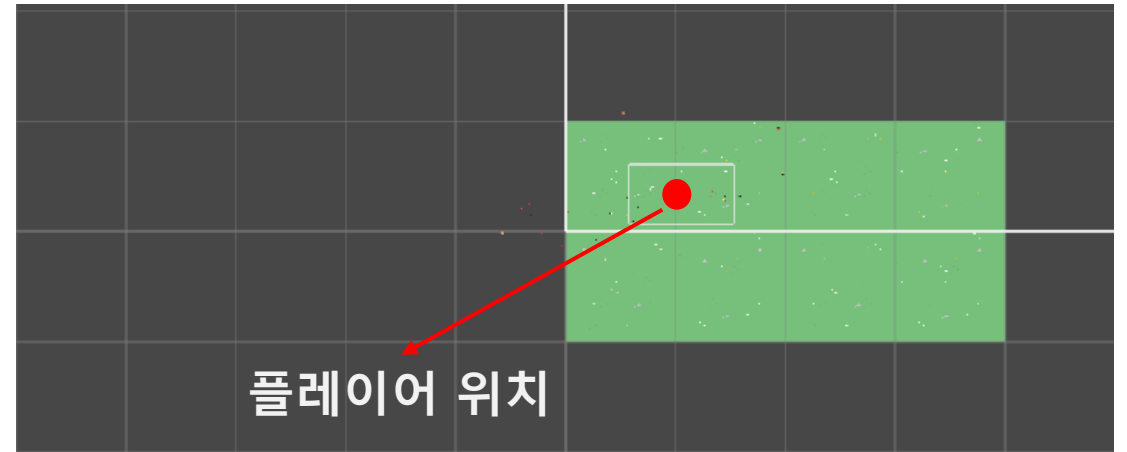
```
public void OnPointerUp(PointerEventData eventData)
{
    _moveDir = Vector2.zero;
    Handler.transform.position = _originPos;
    JoyStick.transform.position = _originPos;
    SetActiveJoyStick(false);
    GameManager.Instance.MoveDir = _moveDir;
}
```

마우스 클릭을 뗐을 때 실행되는 함수

4. 기능 구현 - 무한맵



원점을 기준으로 위치한 타일맵 4개



플레이어가 이동함에 따라 변경된 타일맵 위치

- 화면에 원점을 기준으로 4개의 타일 맵을 배치
- 플레이어의 이동 방향에 따라 타일 맵의 위치 변경

4. 주요 기능 구현 함수 - 맵 조작

```
private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.tag != Define.AreaTag) return;
    if (GameManager.Instance != null && GameManager.Instance.Player != null)
    {
        Vector3 playerPos = GameManager.Instance.Player.transform.position;
        Vector3 myPos = transform.position;

        float dirX = playerPos.x - myPos.x;
        float dirY = playerPos.y - myPos.y;

        float diffX = Mathf.Abs(dirX);
        float diffY = Mathf.Abs(dirY);

        dirX = dirX > 0 ? 1 : -1;
        dirY = dirY > 0 ? 1 : -1;

        if (transform.tag == Define.GroundTag)
        {
            if (diffX > 20)
            {
                transform.Translate(Vector3.right * dirX * 40);
            }
            else if (diffY > 10)
            {
                transform.Translate(Vector3.up * dirY * 20);
            }
        }
    }
}
```

맵의 위치를 변경하는 함수

4. 기능 구현 - UI



타이머 기능

- 플레이어가 게임을 시작한 시간부터 시간 측정
- 시간에 따라 등장하는 몬스터의 패턴 예측 가능

일시 정지 기능

- 플레이 화면 오른쪽 상단에 일시 정지 버튼 클릭 시 일시 정지
- 일시 정지 화면에서는 플레이하기 버튼,
- 다시 시도하기 버튼, 메인으로 가는 버튼 표시

플레이어의 스킬 보여주는 기능

- 선택한 스킬이 과일 게이지 아래에 표시
- 처음에는 스파이크 공이 기본으로 보여집니다.

4. 기능 구현- 몬스터

몬스터 스폰

일정 시간마다 새로운 몬스터가 출현하게 되며, 타이머가 5분이 지나면 보스 몬스터가 등장합니다.

몬스터 종류

몬스터 이미지 및 이름	 Bee	 Blue Bird	 Bunny	 Chicken
최초 등장 시각	게임 시작 시 등장	타이머 1분 후	타이머 2분 후	타이머 3분 후
특징	플레이어를 향해 날아든다.	일정 사정거리 접근 시 속도 증가 및 플레이어를 향해 날아든다.	플레이어를 향해 빠르게 돌진한다.	플레이어를 향해 돌진하며 썩은 달걀 투척한다.

4. 주요 기능 구현 함수 - 적

```
void SpawnEnemyByTime()
{
    if (GameManager.Instance.isBossSpawn) return;
    for(int i = 0; i < _enemyTime.Length; i++)
    {
        if (!IsEnemySpawn[i])
            continue;
        _enemyTime[i] += Time.deltaTime;
        if (_enemyTime[i] > _enemyIntervalTime[i])
        {
            _enemyTime[i] -= _enemyIntervalTime[i];
            switch (i)
            {
                case 0:
                    SpawnEnemy<EnemyBee>(i); break;
                case 1:
                    SpawnEnemy<EnemyBlueBird>(i); break;
                case 2:
                    SpawnEnemy<EnemyBunny>(i); break;
                case 3:
                    SpawnEnemy<EnemyChicken>(i); break;
            }
        }
    }
}
```

시간에 따라 적들을 스폰하는 함수

```
public bool AnyDamage(float damage, GameObject damageCauser,
    int projectile = (int)Define.EProjectile.SpikedBall)
{
    EnemyInfo.CurrentHp -= damage;
    if (EnemyInfo.CurrentHp <= 0)
    {
        gameObject.SetActive(false);
        if (gameObject.tag == Define.BossTag)
        {
            GameManager.Instance.isBossSpawn = false;
            GameManager.Instance.isGameOver = true;
        }
        UI_Play.Instance.DeadEnemyCount++;
        SoundManager.Instance.EnemyDeadSound.Play();
    }
    if (gameObject.tag == Define.BossTag)
    {
        UI_Play.Instance.BossHpBar.fillAmount = EnemyInfo.CurrentHp / EnemyInfo.MaxHp;
    }
    if(projectile==(int)Define.EProjectile.SpikedBall && gameObject.tag!=Define.BossTag)
        OnKnockBack(damageCauser);
    return true;
}
```

몬스터가 데미지를 입었을 시 실행되는 함수

4. 주요 기능 구현 함수 - 적

```
public bool AnyDamage(float damage, GameObject damageCauser,
    int projectile = (int)Define.EProjectile.SpikedBall)
{
    EnemyInfo.CurrentHp -= damage;
    if (EnemyInfo.CurrentHp <= 0)
    {
        gameObject.SetActive(false);
        if (gameObject.tag == Define.BossTag)
        {
            GameManager.Instance.isBossSpawn = false;
            GameManager.Instance.isGameOver = true;
        }
        UI_Play.Instance.DeadEnemyCount++;
        SoundManager.Instance.EnemyDeadSound.Play();
    }
    if (gameObject.tag == Define.BossTag)
    {
        UI_Play.Instance.BossHpBar.fillAmount = EnemyInfo.CurrentHp / EnemyInfo.MaxHp;
    }
    if(projectile==(int)Define.EProjectile.SpikedBall && gameObject.tag!=Define.BossTag)
        OnKnockBack(damageCauser);
    return true;
}
```

몬스터가 데미지를 입었을 때
실행되는 함수

4. 주요 기능 구현 함수 - 적

```
public GameObject GetNearestTarget(float distance=12)
{
    if (isBossSpawn)
    {
        _boss = GameObject.FindWithTag(Define.BossTag);
        return _boss;
    }
    EnemyController = FindAnyObjectByType<EnemyController>();
    List<Enemy> targetList = new();
    int enemyGroupKey = EnemyController.enemyGroup.Count;
    if (enemyGroupKey <= 0)
    {
        return null;
    }

    for(int i=0; i < 4; i++)
    {
        if (!EnemyController.enemyGroup.ContainsKey(types[i]))
            continue;

        targetList.AddRange(EnemyController.enemyGroup[types[i]].
            Where(enemy => enemy.gameObject.activeSelf));
    }
}
```

```
if (Player == null)
{
    return null;
}
var target = targetList.OrderBy(enemy =>
    (Player.Center - enemy.transform.position).sqrMagnitude).FirstOrDefault();

if (target == null || (target.transform.position - Player.Center).sqrMagnitude > distance)
{
    return null;
}
return target.gameObject;
}
```

플레이어에게 가장 가까운 적을 찾는 함수

4. 주요 기능 구현 함수 - 적 : Boss

```
void DashTime()
{
    if (!_isDash)
    {
        BossInfo.Speed = 0.5f;
        _dashTime += Time.deltaTime;
        if (_dashTime > _dashCoolTime)
        {
            _isDash = true;
            _dashTime = 1.5f;
        }
    }
    else
    {
        if (_dashTime > 0)
        {
            BossInfo.Speed = 2.2f;
            _bossRb.mass = 1;
            _dashTime -= Time.deltaTime;
        }
        else
        {
            _isDash = false;
        }
    }
}
```

일정 시간마다 플레이어에게 돌진하는 함수

```
void FireBullet()
{
    _bulletTime += Time.deltaTime;
    if (_bulletTime > _bulletCoolTime)
    {
        _bulletTime -= _bulletTime;
        _bulletSpawnPos = _bossRend.flipX ? RAttackPos : LAttackPos;
        for(int i=-1; i<2; i++)
        {
            Instantiate(BossBulletPrefab, _bulletSpawnPos.position, Quaternion.Euler(0, 0, i*30));
        }
    }
}
```

Boss가 총알을 발사하는 함수

4. 주요 기능 구현 함수 – 적 : Boss

```
void FireBullet()
{
    _bulletTime += Time.deltaTime;
    if (_bulletTime > _bulletCoolTime)
    {
        _bulletTime -= _bulletTime;
        _bulletSpawnPos = _bossRend.flipX ? RAttackPos : LAttackPos;
        for(int i=-1; i<2; i++)
        {
            Instantiate(BossBulletPrefab, _bulletSpawnPos.position, Quaternion.Euler(0, 0, i*30));
        }
    }
}
```

일정 시간마다 플레이어에게 3개의 총알을 발사하는 함수입니다.

4. 기능 구현 - 과일

과일 스폰

일정 시간마다 새로운 과일이 등장하며, 과일에 따라 레벨이 달라집니다.
과일 게이지를 다 채울 경우 스킬을 선택할 수 있고, 레벨이 올라갑니다.
레벨이 올라감에 따라 더 많은 과일을 먹어야 과일 게이지를 다 채울 수 있게 됩니다.

과일 종류

과일 이미지 및 과일 이름	 Apple	 Orange	 Kiwi
최초 등장 시각	게임 시작 시 등장	타이머 1분 후	타이머 2분 후
레벨	레벨 1	레벨 2	레벨 3

4. 주요 기능 구현 함수 - 과일

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag(Define.PlayerTag))
    {
        gameObject.SetActive(false);
        UI_Play.Instance.FruitsCount += (_fruitLv + 1) * 4;
    }
}
```

플레이어와 과일이 부딪히면 과일 게이지가 올라가는 함수



4. 기능 구현 - 아이템

아이템 박스 스폰

일정 시간마다 아이템 박스가 플레이어 주변에 생성됩니다.

아이템 박스를 깨면 회복 아이템, 자석 아이템, 코인이 랜덤으로 나오게 됩니다.

아이템 박스 및 아이템 종류

아이템 박스 및 아이템	 Item Box	 Health Item	 Magnet Item	 Coin Item
특징	플레이어가 직접 깨거나 특정 무기로 깰 수 있다.	아이템을 먹으면 일정량의 체력이 회복된다.	주위에 있는 과일들이 플레이어에게 모여 순식간에 과일 게이지가 차오른다.	코인을 획득할 수 있다.

4. 주요 기능 구현 함수 – 아이템 박스

```
public class ItemBox : MonoBehaviour
{
    public Action<GameObject> OnItemBoxCrashed;

    * Unity 메시지 | 참조 0개
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag(Define.ProjectileTag) || collision.CompareTag(Define.PlayerTag))
        {
            OnItemBoxCrashed?.Invoke(gameObject);
            SoundManager.Instance.ItemGainSound.Play();
            gameObject.SetActive(false);
        }
    }
}
```

플레이어나 특정 무기가 아이템 박스와 부딪히면 실행되는 함수

4. 주요 기능 구현 함수 – 아이템 : 자석

```
void MagnetFieldActive()
{
    _time += Time.deltaTime;
    if (_time < _magnetTime)
    {
        Collider2D[] fruits = Physics2D.OverlapCircleAll(
            _player.transform.position,
            20f,
            LayerMask.GetMask("Fruit"));
        foreach (Collider2D fruit in fruits)
        {
            MoveFruitsTowardPlayer(fruit.transform);
        }
    }
    else
    {
        isMagnetActive = false;
        Destroy(gameObject);
    }
}
```

아이템 박스에서 나온 자석 아이템을 먹었을 경우 실행되는 함수

4. 주요 기능 구현 함수 – 아이템 : 자석

```
private void MoveFruitsTowardPlayer(Transform fruit)
{
    Vector3 dir = (_player.transform.position - fruit.position).normalized;
    fruit.position = Vector3.MoveTowards(fruit.position,
        _player.transform.position,
        magnetSpeed * Time.deltaTime);
}
```

자석 아이템을 먹었을 경우 플레이어를 향해 과일들이 움직이는 함수

4. 기능 구현 - 무기

무기(스킬)

게임 시작 시 플레이어가 쓰는 첫 무기는 스파이크 공으로, 가장 가까운 적에게 스파이크 공을 날립니다.
무기는 과일 게이지를 모아 스킬을 선택할 때 추가할 수 있습니다.

무기 종류 - (1)

무기 이미지 및 무기 이름			
	Spiked Ball	Soccer Ball	Brick
특징	스파이크 공은 가장 가까운 적에게 날아가며, 스파이크 공을 맞은 적은 넉백 된다.	축구공은 적에게 맞으면 튕겨진다. 축구공도 가장 가까운 적을 향해 날아간다.	벽돌은 위로 튀어 올랐다가 떨어지면서 적에게 피해를 준다.

4. 기능 구현 - 무기

무기 종류 - (2)

무기 이미지 및 무기 이름	 Boomerang	 Rocket	 Shield
특징	랜덤 방향으로 부메랑을 던지며, 다시 플레이어 방향으로 되돌아온다.	랜덤 방향으로 로켓이 날아가며, 적과 부딪히면 폭발 하며 적에게 데미지를 준다.	플레이어 주변에 생성되는 방어막으로 들어온 적에게 도트 데미지를 준다.

4. 주요 기능 구현 함수 - 무기

```
private T GetProjectile<T>(int projectileType) where T : Projectile
{
    Type type = typeof(T);
    if (projectileGroup.ContainsKey(type))
    {
        for(int i=0; i < projectileGroup[type].Count; i++)
        {
            if (!projectileGroup[type][i].gameObject.activeSelf)
            {
                projectileGroup[type][i].gameObject.SetActive(true);
                return projectileGroup[type][i] as T;
            }
        }
        GameObject newProjectile = Instantiate(projectilePrefabs[projectileType]);
        newProjectile.transform.parent = _projectilePool.transform;
        T controller = newProjectile.GetComponent<T>();
        projectileGroup[type].Add(controller);
        return controller as T;
    }
    else
    {
        GameObject newProjectile = Instantiate(projectilePrefabs[projectileType]);
        newProjectile.transform.parent = _projectilePool.transform;
        T controller = newProjectile.GetComponent<T>();
        projectileGroup[type] = new List<Projectile>();
        projectileGroup[type].Add(controller);
        return controller as T;
    }
}
```

무기 타입에 맞는 무기를 가져오는 함수입니다.

4. 주요 기능 구현 함수 - 무기 : 부메랑

```
void FireBoomerang()
{
    if (_time > _changeDirTime && !_isDirChange)
    {
        _moveDir = -_moveDir;
        _isDirChange = true;
    }
    _rb.AddForce(_moveDir * BoomerangInfo.Speed, ForceMode2D.Force);
    _time += Time.deltaTime;
    transform.Rotate(0, 0, 200 * Time.deltaTime, Space.World);
}
```

랜덤 방향으로 날아갔다가 반대 방향으로 날아가는 부메랑 관련 함수입니다.

4. 주요 기능 구현 함수 - 무기 : 로켓

```
private void OnTriggerStay2D(Collider2D collision)
{
    Enemy enemy = collision.GetComponent<Enemy>();

    IDamageable damageable = collision.GetComponent<IDamageable>();
    if (damageable!=null && collision.CompareTag(Define.EnemyTag) ||
        collision.CompareTag(Define.BossTag))
    {
        damageable.AnyDamage(_atk + _playerController.playerInfo.Atk,
            _player, (int)Define.EProjectile.Rocket);
    }
}
```




적과 충돌하면 폭발이 일어나며 폭발 반경으로 적이 데미지를 입는 함수

4. 기능 구현 - 스킬

스킬

- 과일 게이지를 다 채우면 게임이 멈추게 되고 스킬을 선택할 수 있는 창이 나타납니다.
6개의 무기 스킬과 4개의 기타 스킬 중 3개의 스킬이 랜덤으로 나타나 1개의 스킬을 선택할 수 있습니다.
- 스킬은 기본 무기 스킬인 스파이크 공을 포함하여 5개를 가질 수 있으며,
5개의 스킬을 선택했을 경우 스킬을 선택하는 화면에서는 선택한 스킬만 보여집니다.
- 플레이어가 가진 스킬이 스킬 선택 화면에 나타나 그 스킬을 선택하면 스킬 레벨이 올라가고,
최대 3레벨까지 올릴 수 있습니다.

무기 스킬 외의 기타 스킬 종류

스킬 이미지 및 스킬 이름				
	Hp Up	Power Up	Speed Up	Magnet Up
특징	플레이어의 최대 체력 증가	플레이어의 공격력 증가	플레이어의 이동 속도 증가	과일 획득 범위 증가

4. 주요 기능 구현 함수 - 스킬

```
void GetSkillList()
{
    int rndNum;
    if (_skills.PlayerSkillList.Count < 5)
    {
        rndNum = Random.Range(0, 10);
        for(int i=0; i < 3;)
        {
            if (RandomSkillList.Contains(rndNum) ||
                _skills.SkillList[rndNum].SkillLevel >= 4)
            {
                rndNum = UnityEngine.Random.Range(0, 10);
            }
            else
            {
                RandomSkillList.Add(rndNum);
                i++;
            }
        }
    }
}
```

```
else
{
    int[] playerSkillNums = new int[5];
    for(int i=0; i < 5; i++)
    {
        playerSkillNums[i] = _skills.PlayerSkillList[i].SkillNumber;
    }
    rndNum = Random.Range(0, 5);
    for(int i=0; i < 3;)
    {
        if (RandomSkillList.Contains(playerSkillNums[rndNum]) ||
            _skills.SkillList[playerSkillNums[rndNum]].SkillLevel >= 4)
        {
            rndNum = UnityEngine.Random.Range(0, 5);
        }
        else
        {
            RandomSkillList.Add(playerSkillNums[rndNum]);
            i++;
        }
    }
}
```

10개의 스킬 리스트 중 3개를 랜덤으로 뽑는 함수입니다.
플레이어의 스킬이 5개가 되면 5개 중에 레벨이 3을 넘지 않는 스킬만 랜덤으로 뽑습니다.

4. 주요 기능 구현 함수 - 스킬

```
public void OnChooseSkillButtonClick(int idx)
{
    int skillIdx = RandomSkillList[idx];
    int skillLevel = _skills.SkillList[skillIdx].SkillLevel;
    Skills.Skill mySkill = _skills.SkillList[skillIdx];
    if (_projectileController != null && skillLevel <= 3)
    {
        switch (skillIdx)
        {
            case 0:
                GameObject projectilePool = GameObject.Find("ProjectilePool");
                SpikedBall[] spikedBalls = projectilePool.
                    GetComponentsInChildren<SpikedBall>(true);
                for(int i=0; i < spikedBalls.Length; i++)
                {
                    spikedBalls[i].SpikedBallInfo.Atk *= 1.1f;
                }
                break;
            case 1: case 2: case 3: case 4: case 5:
                _projectileController.IsProjectileActive[RandomSkillList[idx]] = true; break;
            case 6:
                _playerController.playerInfo.MaxHp *= (1 + 0.1f * skillLevel); break;
            case 7:
                _playerController.playerInfo.Atk += skillLevel; break;
            case 8:
                _playerController.playerInfo.Speed *= (1 + 0.1f * skillLevel); break;
            case 9:
                _playerController.MagnetRange += 0.3f * skillLevel; break;
        }
        gameObject.SetActive(false);
    }
}
```

```

        gameObject.SetActive(false);
        if (_skills.PlayerSkillList.Count <= 5)
        {
            bool isPlayerSkill = false;
            for(int i=0; i < _skills.PlayerSkillList.Count; i++)
            {
                if (_skills.PlayerSkillList[i].SkillNumber ==
                    _skills.SkillList[skillIdx].SkillNumber)
                {
                    isPlayerSkill = true; break;
                }
            }
            if (!isPlayerSkill)
            {
                _skills.PlayerSkillList.Add(_skills.SkillList[skillIdx]);
            }
            mySkill.SkillLevel = ++skillLevel;
            mySkill.SkillExplain = Define.skillExplain2[skillIdx];
            _skills.SkillList[skillIdx] = mySkill;
            OnSkillChosed?.Invoke(skillIdx);
        }
        Time.timeScale = 1;
    }
}
```

3개의 스킬 중 1개를 선택할 때 실행되는 함수입니다.

4. 주요 기능 구현 함수 - 타이머

```
IEnumerator StartTimer()
{
    while (CurrentTime < 600)
    {
        CurrentTime += Time.deltaTime;
        DisplayTimer();
        yield return null;
        if (CurrentTime >= 480)
        {
            CurrentTime = 0;
            yield break;
        }
    }
}
```

게임이 시작되면 실행되는 타이머 함수

```
void DisplayTimer()
{
    _minute = (int)CurrentTime / 60;
    _second = (int)CurrentTime % 60;
    _timerText.text = _minute.ToString("00") +
        ":" + _second.ToString("00");
}
```

시간을 화면에 출력하는 함수

감사합니다.