욕망과 도파민이 만나면?

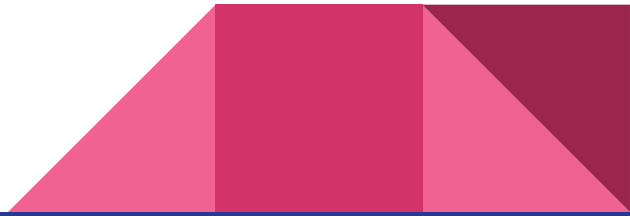
목차

- 게임 소개
- 기획 과정
- 개발
  - 배틀 필드
  - 플레이어
  - 몬스터
- 게임 화면
- 회고

게임 소개

게임 장르

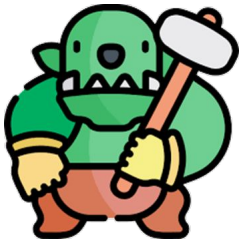
- 1 인칭
- 슈팅
- 디펜스
- 이세계 (!?)
- 자체 더빙 (??!?)

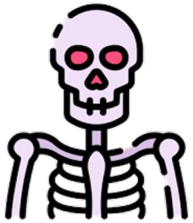


게임 이름 선정

게임 코어 루프

특이사항

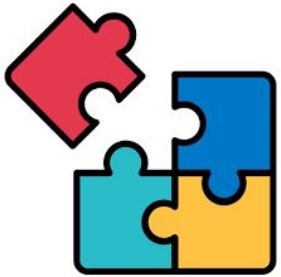




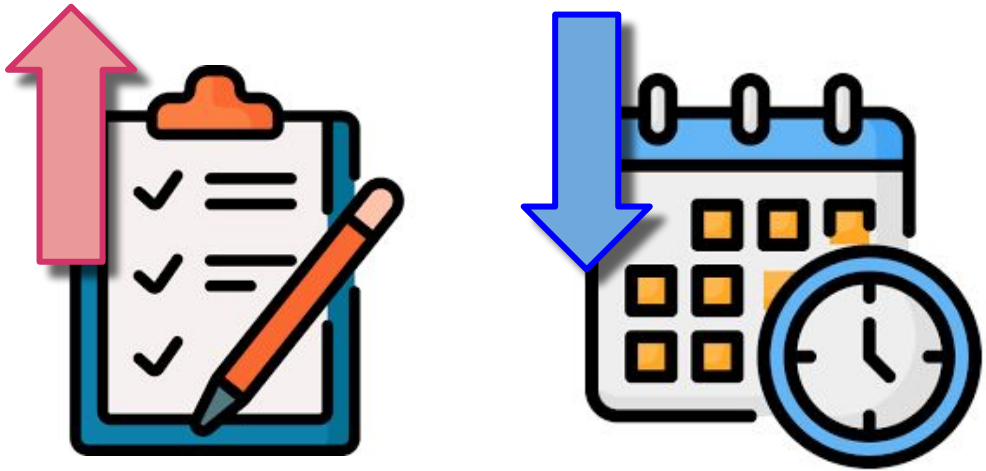
Demonstration

기획 과정

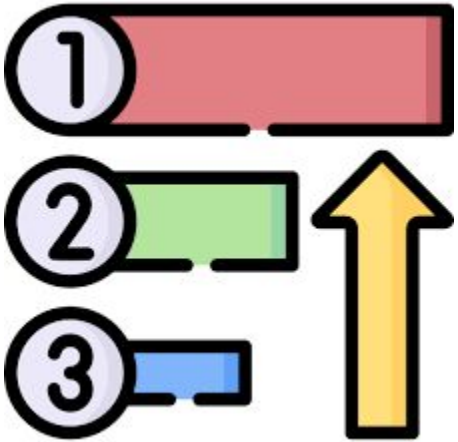
욕망의 시작



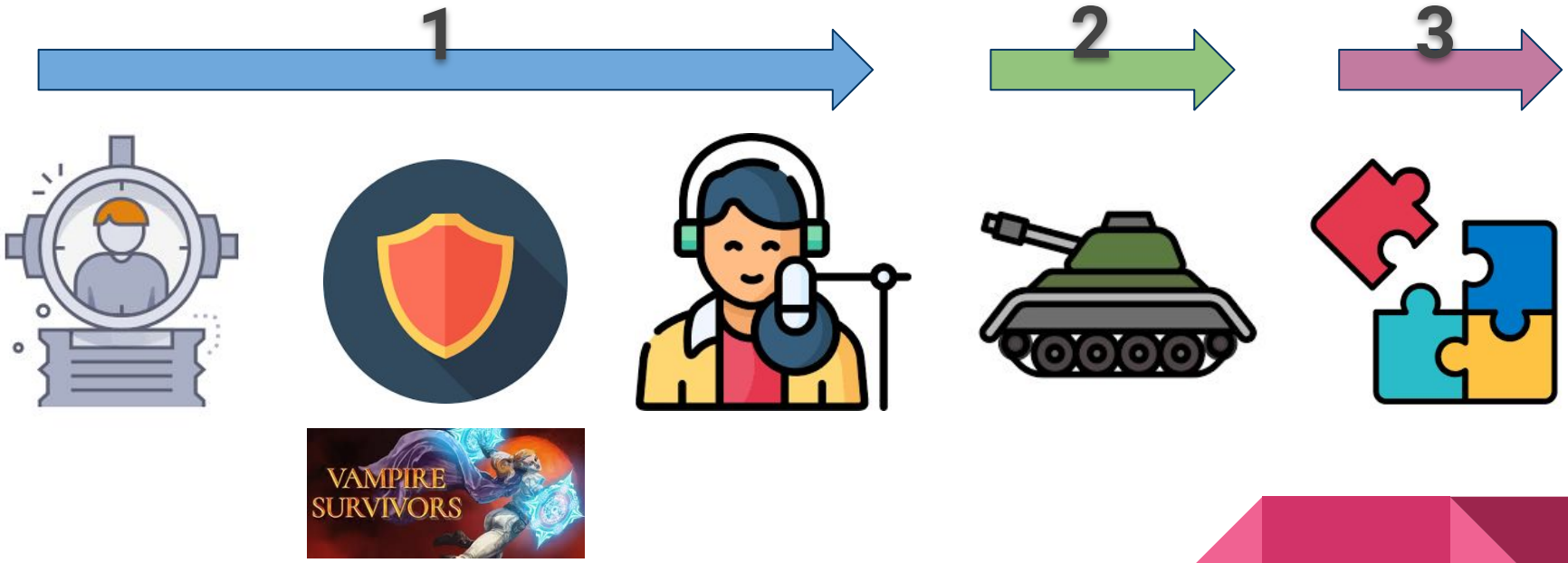
현실 직시



선택과 집중 - 우선순위



선택과 집중 - 우선순위

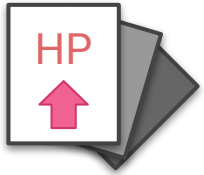


개발 - 배틀 필드

배틀 필드

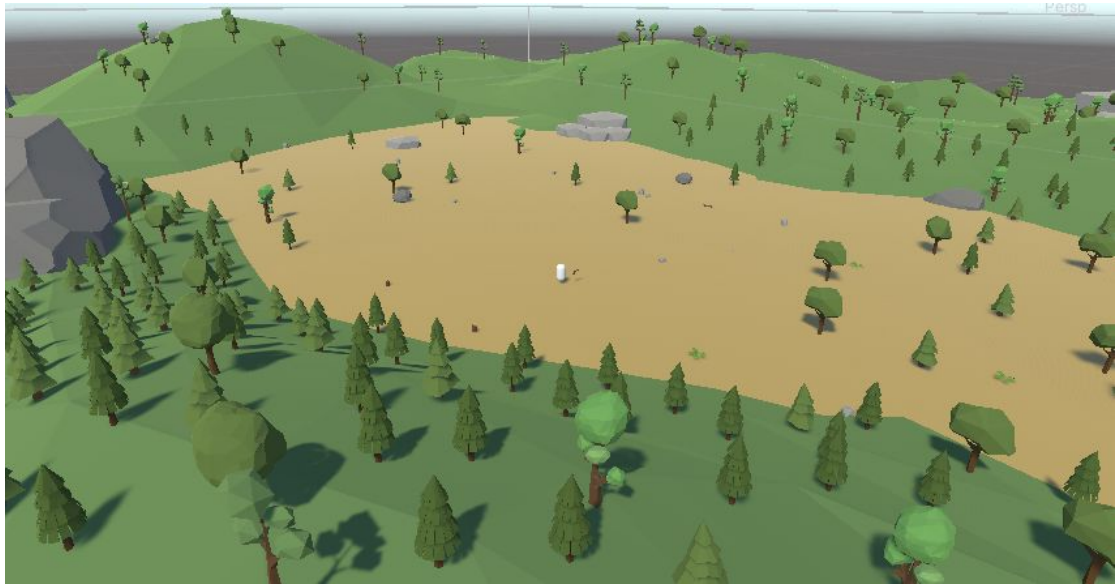
사운드매니저

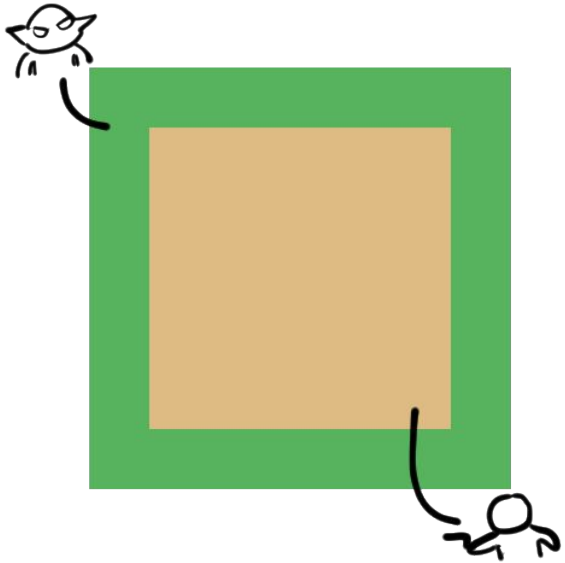
게임매니저



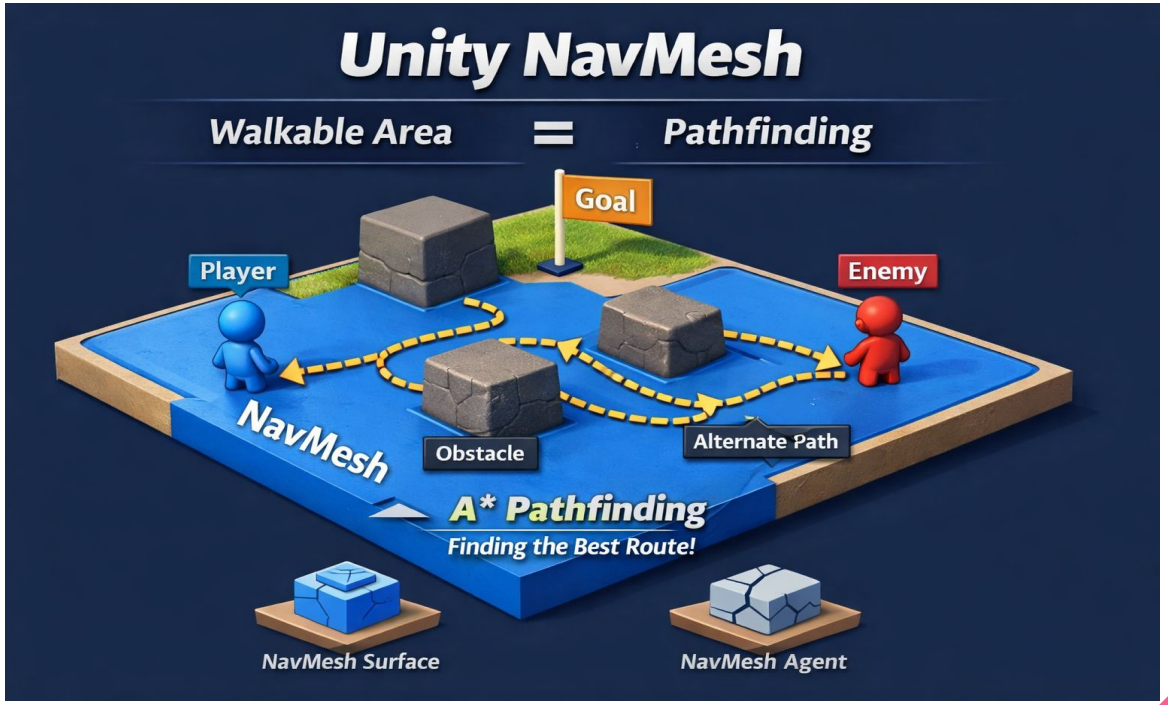
어빌리티매니저

맵 설명

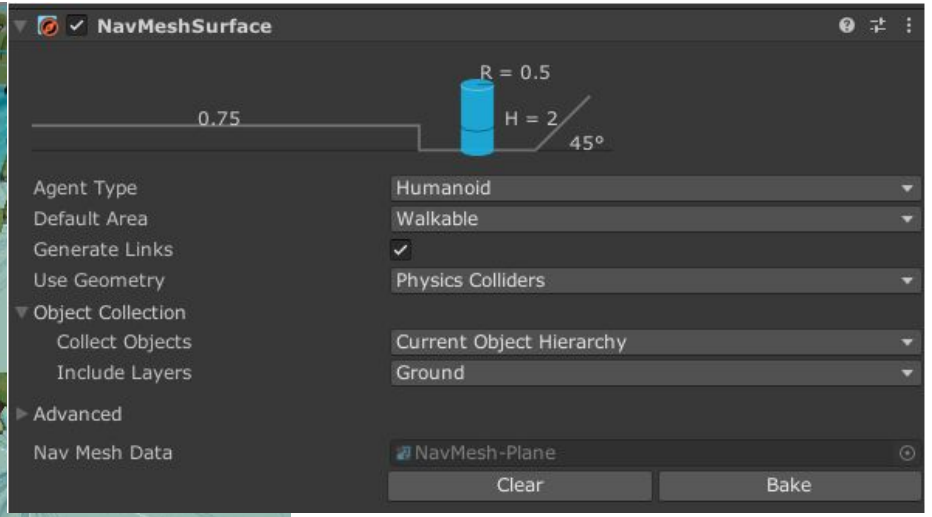




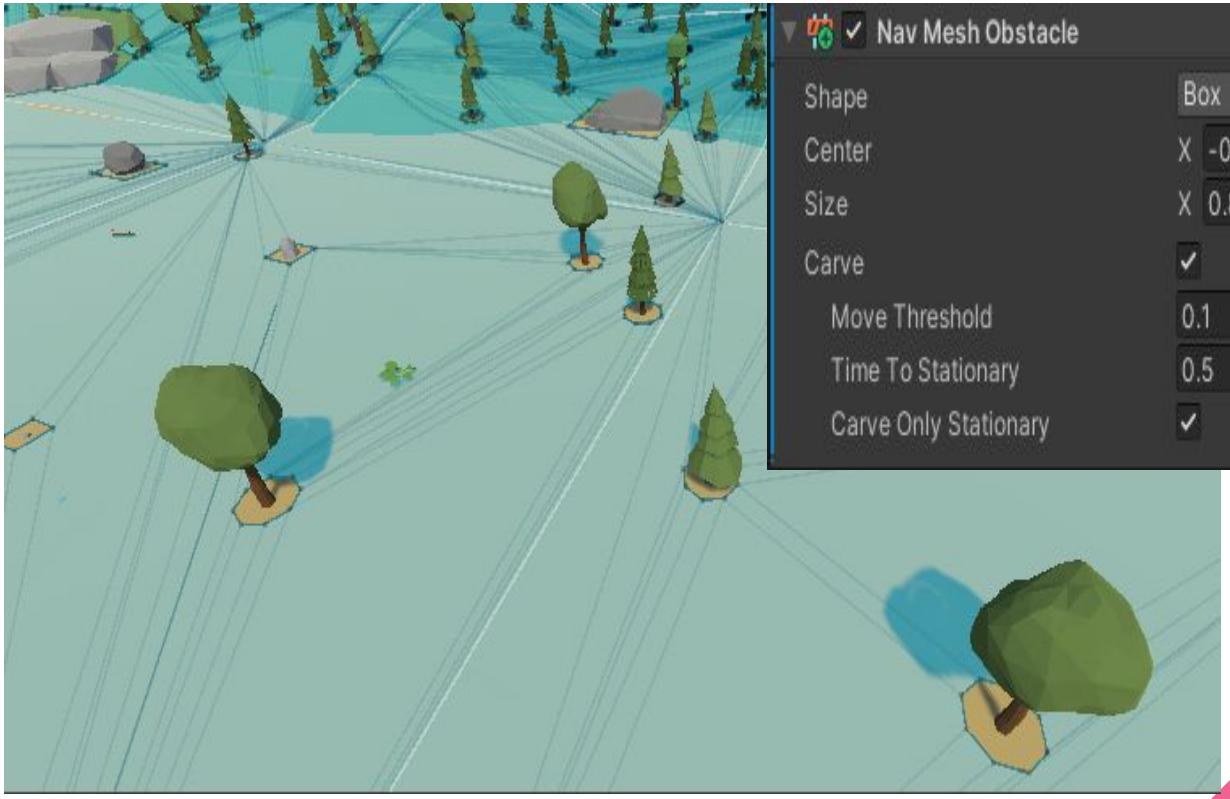
NavMesh

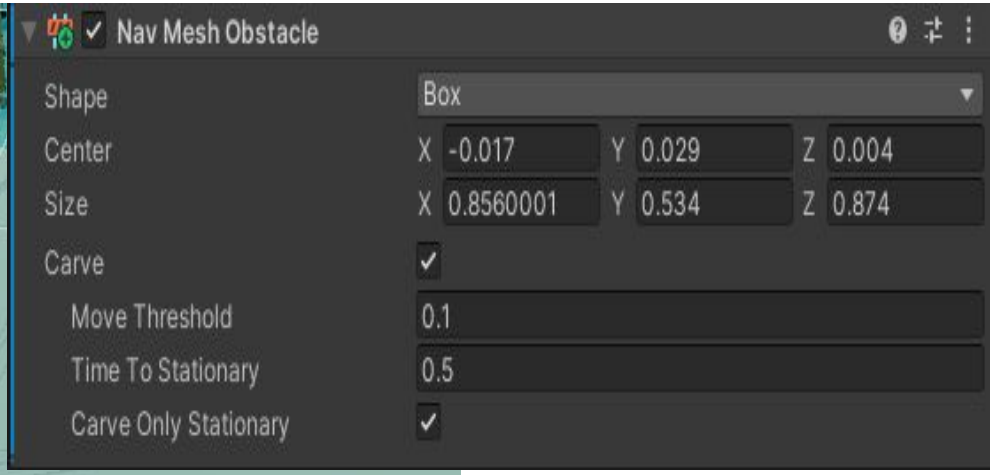

Unity NavMesh
Walkable Area
=
Pathfinding
Goal
Player
Enemy
NavMesh
Obstacle
Alternate Path
A* Pathfinding
Finding the Best Route!
NavMesh Surface
NavMesh Agent

NavMesh Surface

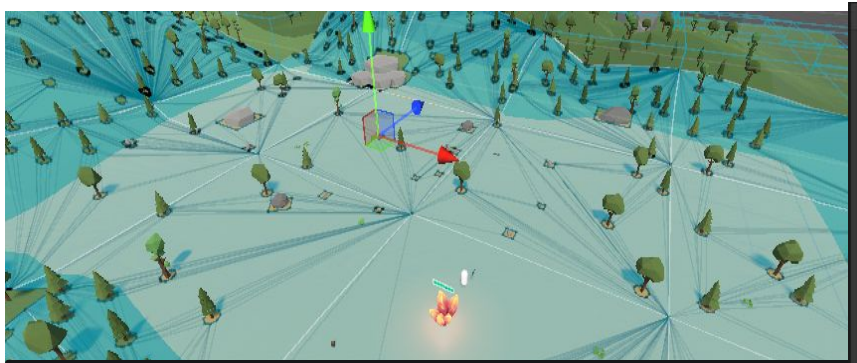


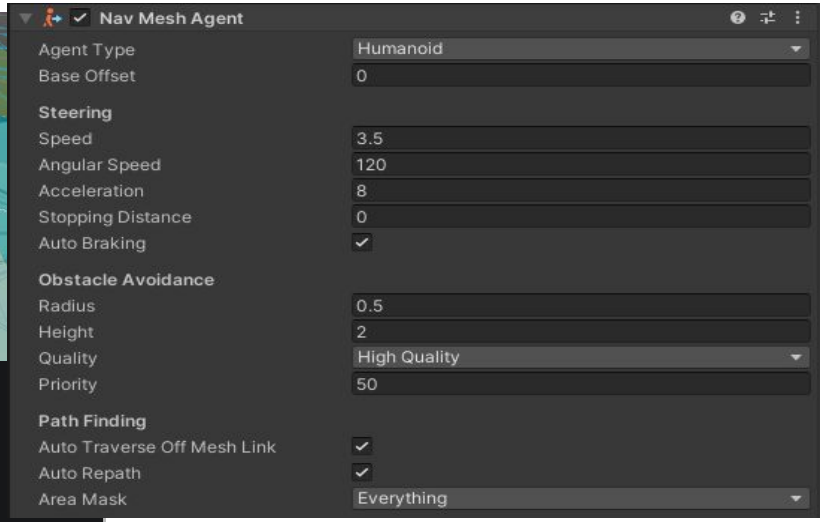
NavMesh Obstacle





NavMesh Agent

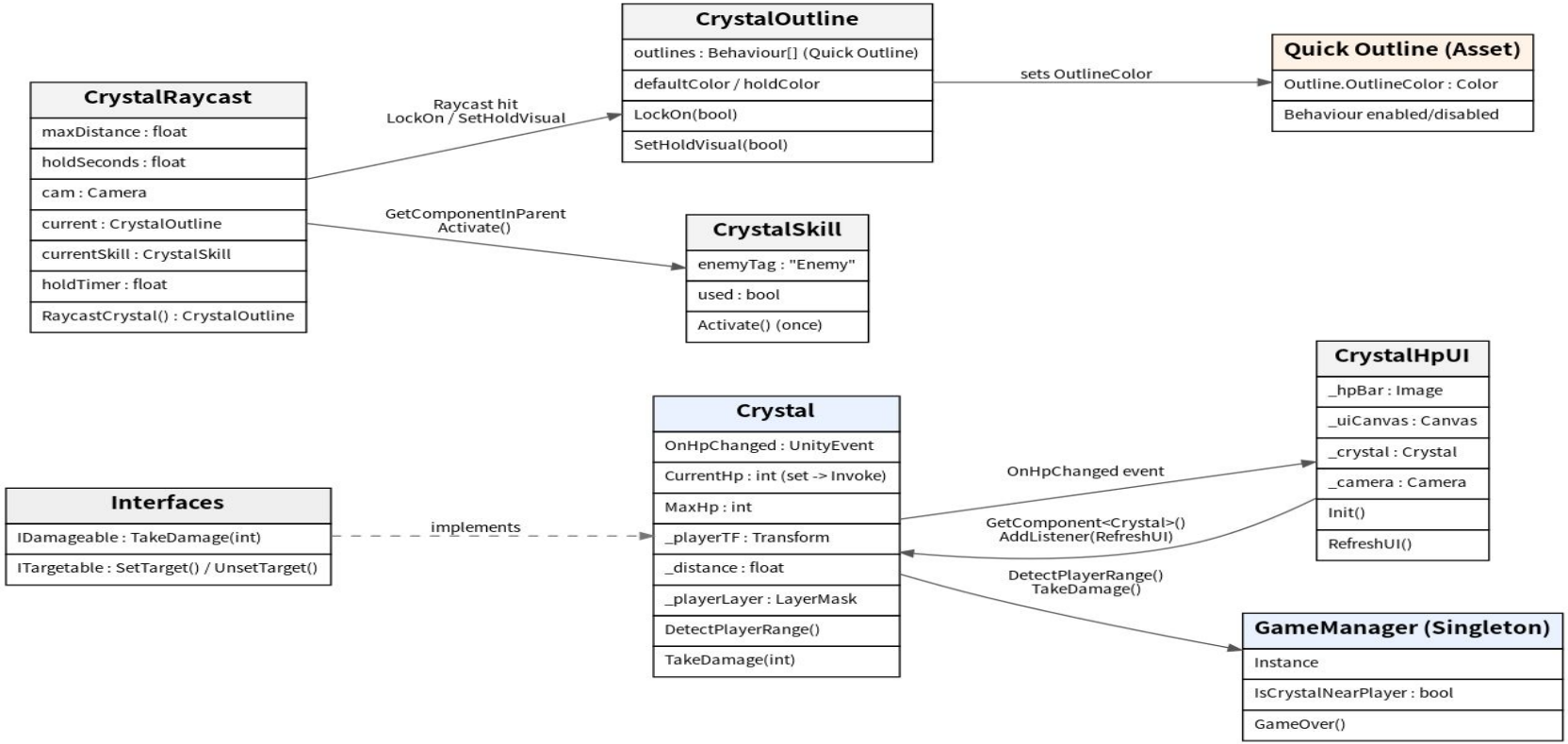





```
public class MonsterMovement : MonoBehaviour
{
    [SerializeField] private Transform _crystalTF;
    [SerializeField] private Transform _playerTF;
    [SerializeField] private Monster _monster;
    private NavMeshAgent _navMeshAgent;

    private void Start()
    {
        _navMeshAgent = GetComponent<NavMeshAgent>();
        _navMeshAgent.speed = _monster.MoveSpeed;
        _navMeshAgent.angularSpeed = 200f;
        _navMeshAgent.acceleration = _monster.MoveSpeed * 1.5f;
        _navMeshAgent.stoppingDistance = _monster.AttackRange;
    }
```

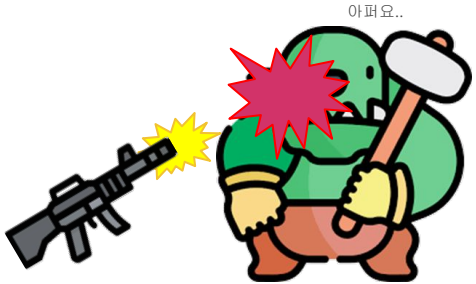
크리스탈

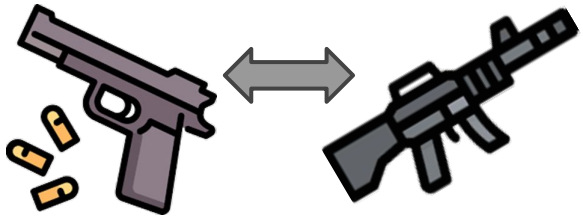
크리스탈


CrystalRaycast
maxDistance : float
holdSeconds : float
cam : Camera
current : CrystalOutline
currentSkill : CrystalSkill
holdTimer : float
RaycastCrystal() : CrystalOutline
Raycast hit
LockOn / SetHoldVisual
GetComponentInParent
Activate()
CrystalOutline
outlines : Behaviour[] (Quick Outline)
defaultColor / holdColor
LockOn(bool)
SetHoldVisual(bool)
sets OutlineColor
Quick Outline (Asset)
Outline.OutlineColor : Color
Behaviour enabled/disabled
CrystalSkill
enemyTag : "Enemy"
used : bool
Activate() (once)
Interfaces
IDamageable : TakeDamage(int)
ITargetable : SetTarget() / UnsetTarget()
implements
Crystal
OnHpChanged : UnityEvent
CurrentHp : int (set -> Invoke)
MaxHp : int
_playerTF : Transform
_distance : float
_playerLayer : LayerMask
DetectPlayerRange()
TakeDamage(int)
OnHpChanged event
GetComponent<Crystal>()
AddListener(RefreshUI)
DetectPlayerRange()
TakeDamage()
CrystalHpUI
_hpBar : Image
_uiCanvas : Canvas
_crystal : Crystal
_camera : Camera
Init()
RefreshUI()
GameManager (Singleton)
Instance
IsCrystalNearPlayer : bool
GameOver()

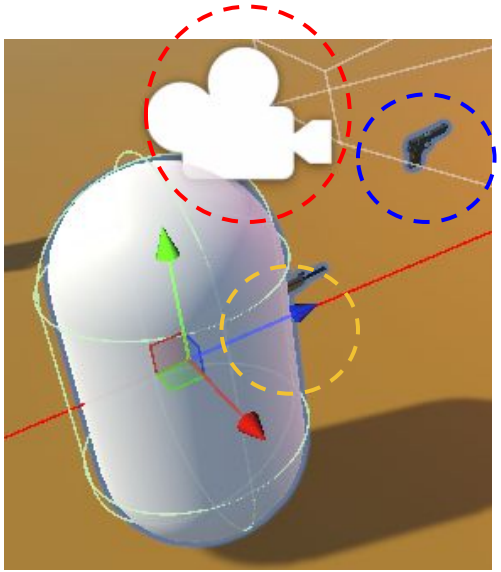
개발 - 플레이어

플레이어의 역할





플레이어 구현 - 프리팹 구조



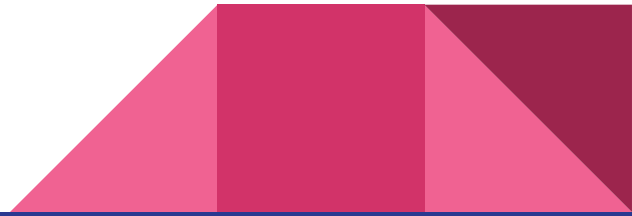
Main Camera

Weapon

GrenadePoint

플레이어 구현 - 클래스 구조

Status	체력, 이동속도 등 + 피격 함수
Controller	시선, 이동, 점프
Level	경험치, 레벨
Weapon	무기 조준, 교체, 사용, 재장전
AbilityBase	어빌리티 적용



플레이어 구현 - Controller


```
private void MoveRigidbody()
{
    _dir = Vector3.zero;
    _dir.x = Input.GetAxisRaw("Horizontal");
    _dir.z = Input.GetAxisRaw("Vertical");
    if (_dir == Vector3.zero)
    {
        _isWalking = false;
        return;
    }
    Vector3 moveDir = transform.TransformDirection(_dir);
    _rb.MovePosition(transform.position + moveDir.normalized * (_moveSpeed * Time.deltaTime));
    _isWalking = true;
}
```

플레이어 구현 - Controller (cont`)

```
private void Jump()
{
    if (Input.GetKey(KeyCode.Space) && !_isJumping)
    {
        _isJumping = true;
        AudioManager.Instance.PlaySound(_jumpSound);
        _rb.AddForce(transform.up * 7, ForceMode.Impulse);
    }
}

이벤트 함수  신규 *
private void OnCollisionEnter(Collision other)
{
    if (other.gameObject.CompareTag("Ground"))
    {
        _isJumping = false;
    }
}
```

플레이어 구현 - Weapon ; 교체

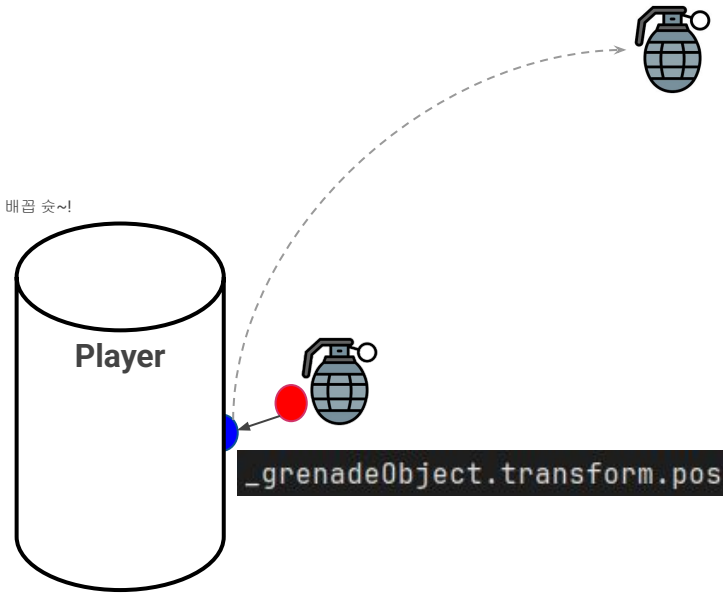

```
_weaponObjects[index].transform.position =
    Vector3.Lerp(_disWpPosArr[index].position,
        _enWpPosArr[index].position, interval);
_weaponObjects[_curWpIndex].transform.position =
    Vector3.Lerp(_enWpPosArr[_curWpIndex].position,
        _disWpPosArr[_curWpIndex].position, interval);
```

플레이어 구현 - Weapon ; 재장전


```
_weaponObjects[_curWpIndex].transform.localRotation =
    Quaternion.Slerp(origin, target, interval);
```

```
_weaponObjects[_curWpIndex].transform.localRotation =
    Quaternion.Slerp(target, origin, interval);
```

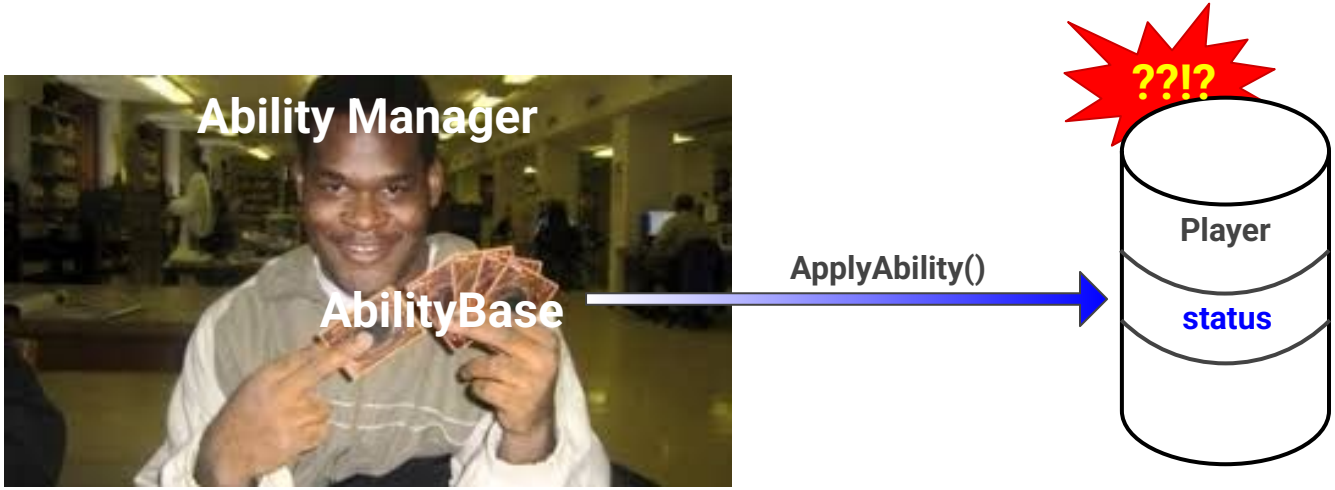
플레이어 구현 - Weapon ; 수류탄 투척




```
GameObject grenadeObj = Instantiate(_grenadeObject,
    _grenadePoint.transform.position, _grenadePoint.transform.rotation);

Rigidbody rb = grenadeObj.GetComponent<Rigidbody>();
rb.isKinematic = false;
rb.constraints = RigidbodyConstraints.None;
rb.useGravity = true;

SphereCollider col = grenadeObj.AddComponent<SphereCollider>();
col.radius = 0.15f;
col.isTrigger = false;

Vector3 throwDir = _grenadePoint.transform.forward * 14f + _grenadePoint.transform.up * 8f;

rb.AddForce(throwDir, ForceMode.Impulse);
```

```
_grenadeObject.transform.position = _grenadePoint.transform.position;
```

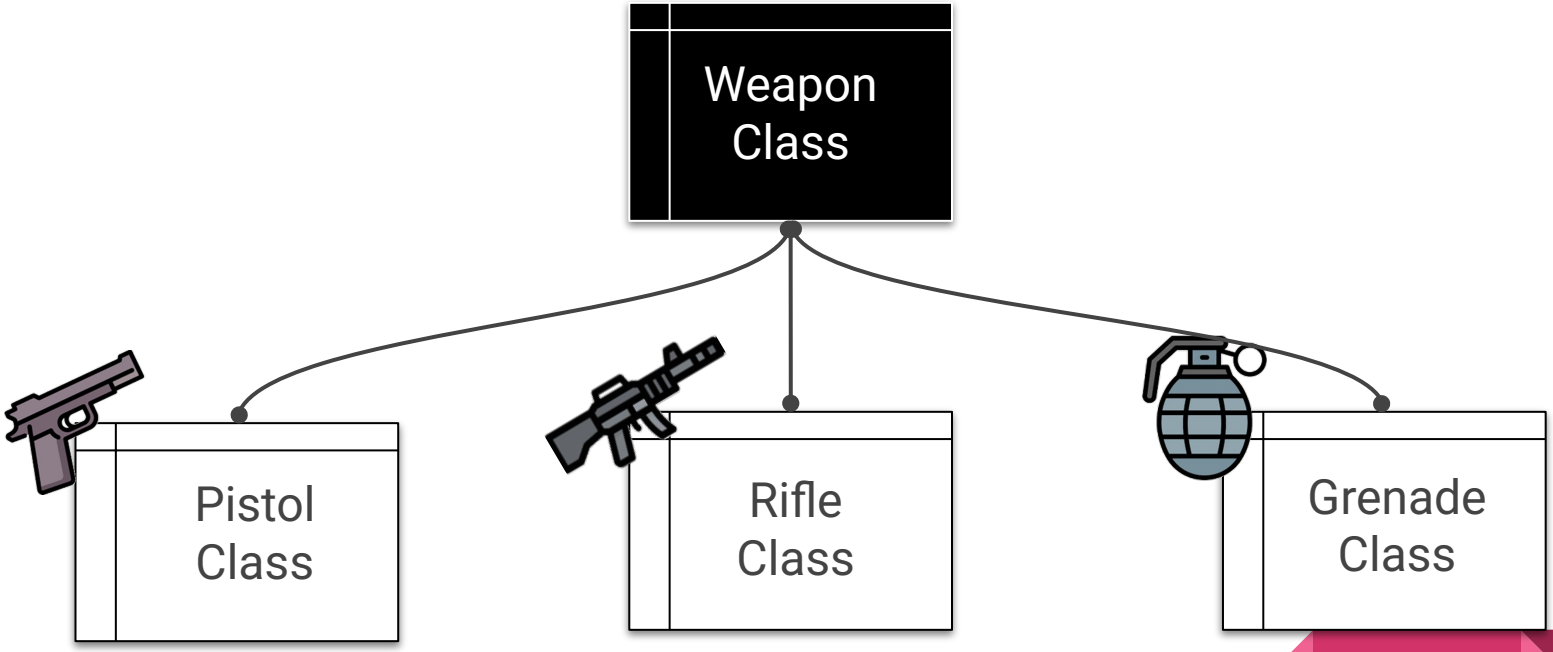
플레이어 구현 - Ability Base



개발 - 무기

준비된 무기

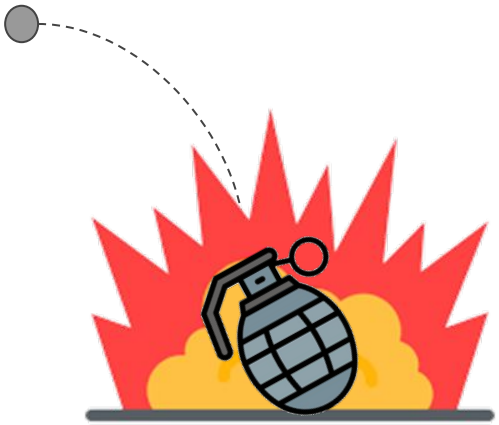
무기 데이터 구조



각 무기 Status

Object 종류	Status	Type	initial	min	MAX	min by Ability	Max by Ability
권총	Damage	int	5	2	10	-1	2
	CriticalChance	float	0	0	0.7	-0.02	0.05
	CriticalDamage	int	15	4	30	-2	4
	Magazine	int	7	7	15	-1	2
	AttackRate	float	1	-	-	-	-
소총	Damage	int	10	5	30	-2	3
	CriticalChance	float	0	0	0.2	-0.01	0.03
	CriticalDamage	int	15	10	40	-2	3
	Magazine	int	30	30	45	-1	2
	AttackRate	float	0.3	-	-	-	-
수류탄	Damage	int	50	25	100	-5	10
	EffectRange	float	10	10	10	-	-
	Magazine	int	3	0	6	-1	1
	ChargeTime	float	3	-	-	-	-

수류탄 탄착



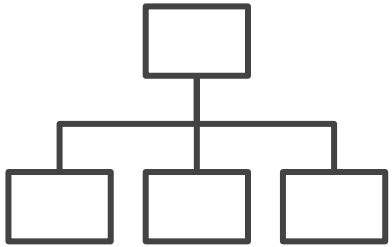
```
Collider[] colliders = Physics.OverlapSphere(transform.position, EffectRange);
```

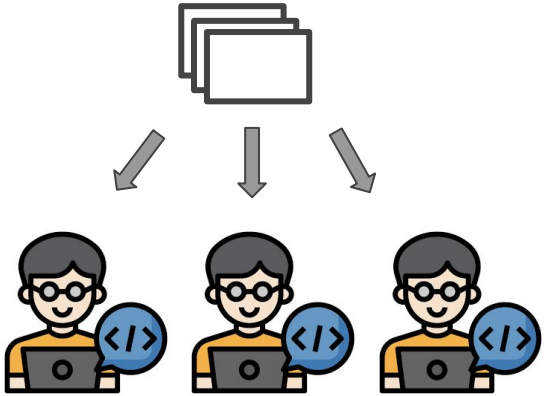
```
rb.AddExplosionForce(ExplosionForce, transform.position,
    explosionRadius: EffectRange, upwardsModifier: 2.0f);
```

감사하지말입니다

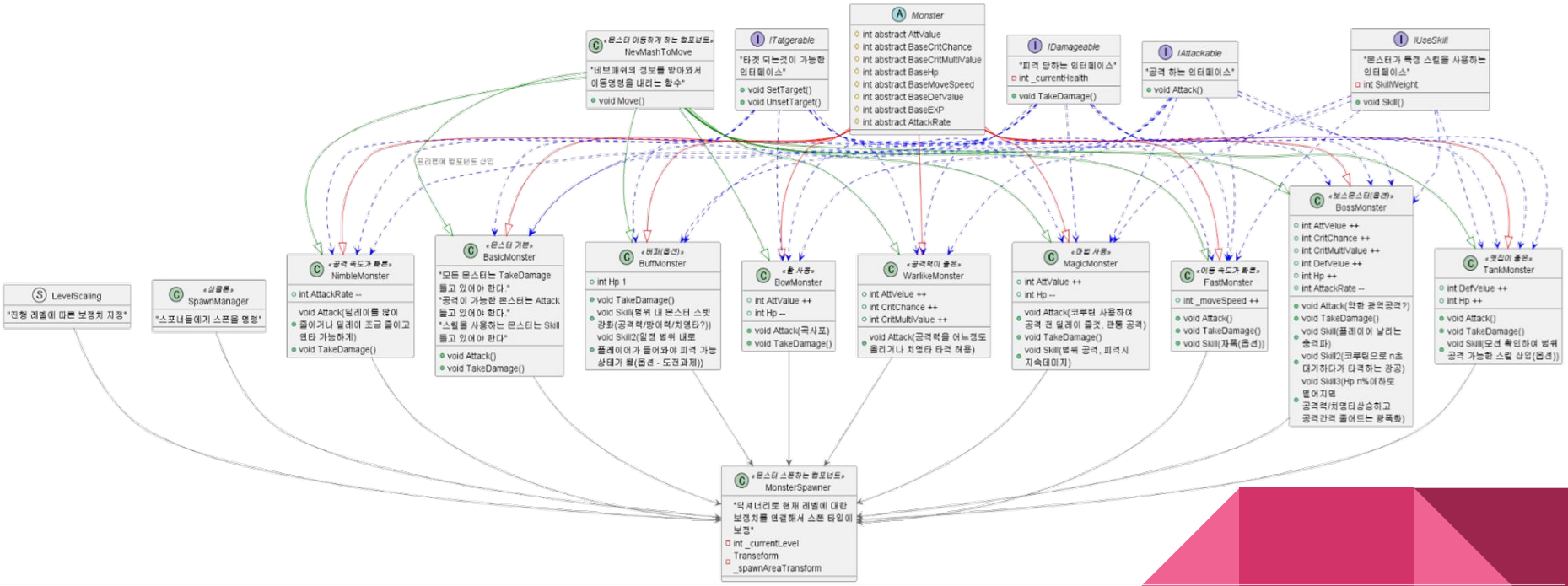
몬스터

몬스터 구현 방식 요약

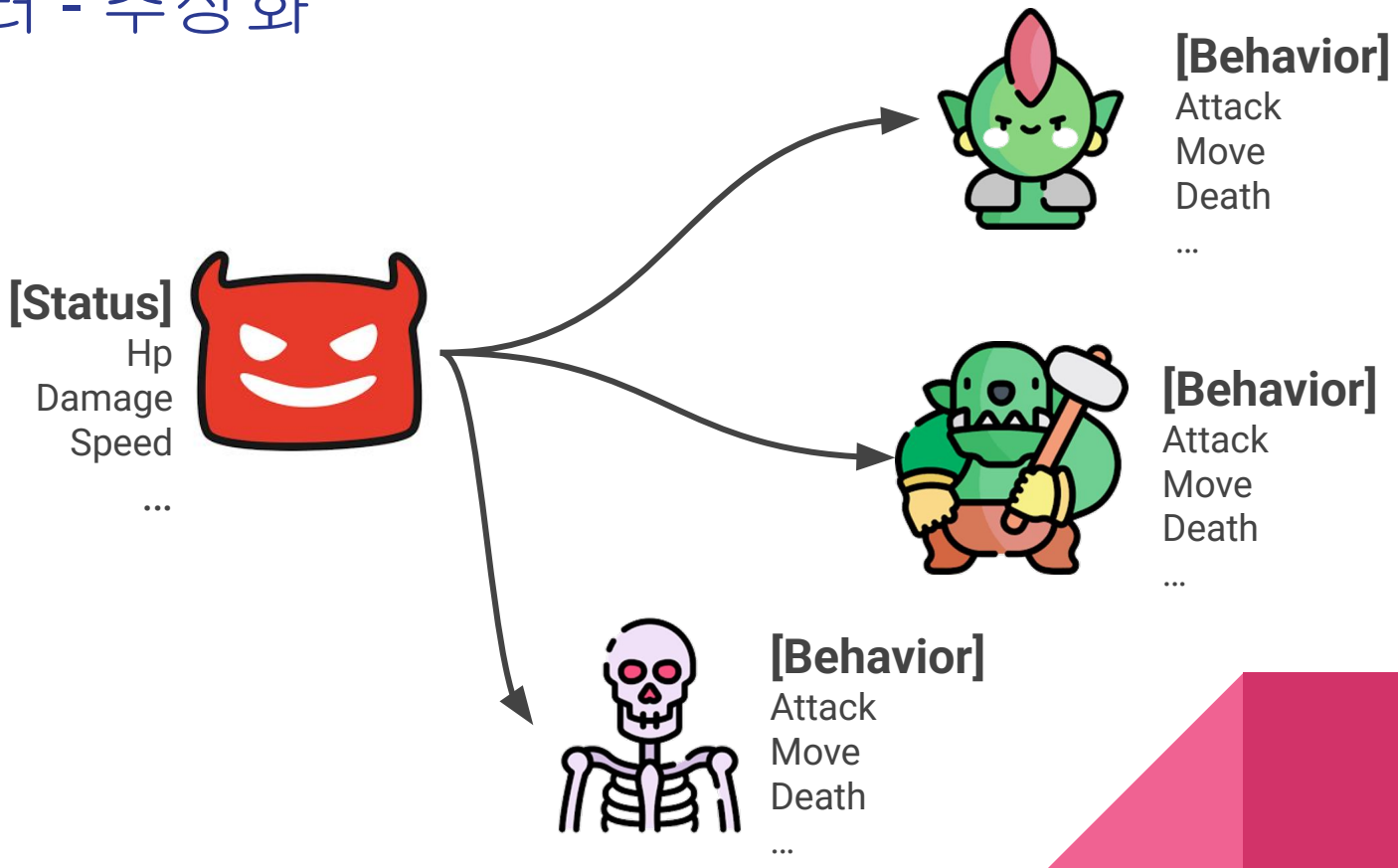




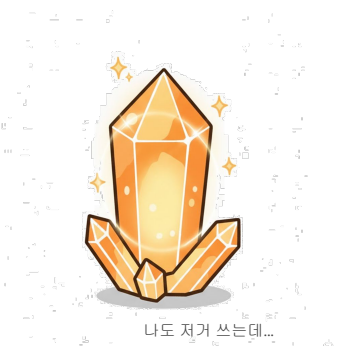
몬스터 관련 Object 상호작용 도식화

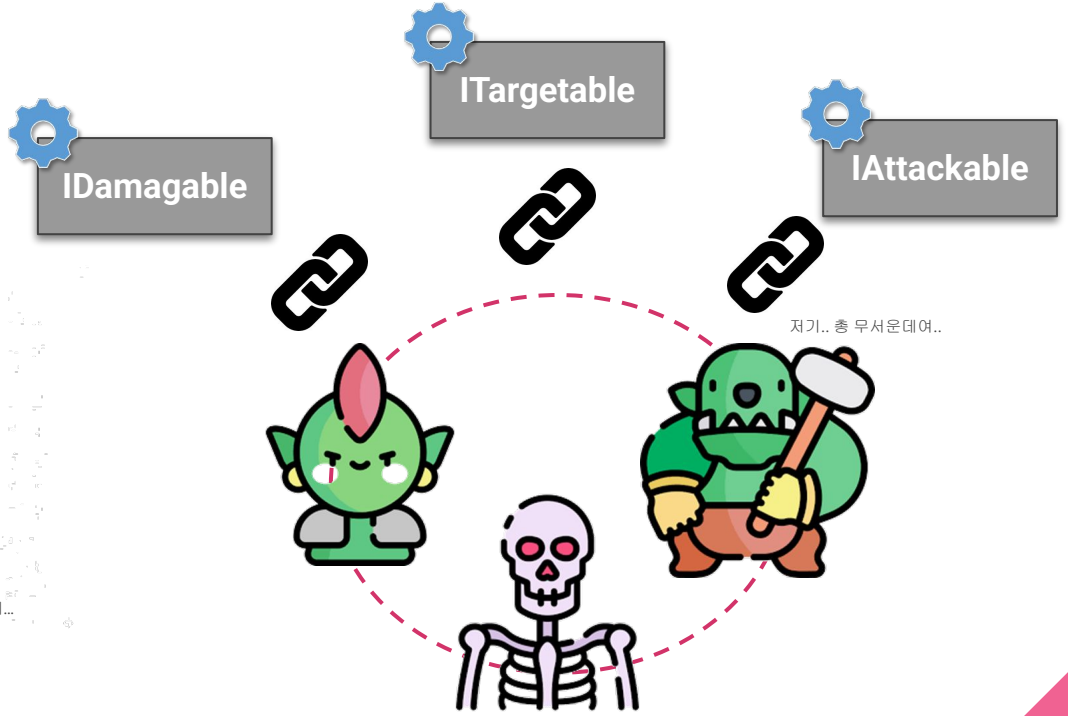


몬스터 - 추상화


[Status]
Hp
Damage
Speed
...
[Behavior]
Attack
Move
Death
...
[Behavior]
Attack
Move
Death
...
[Behavior]
Attack
Move
Death
...

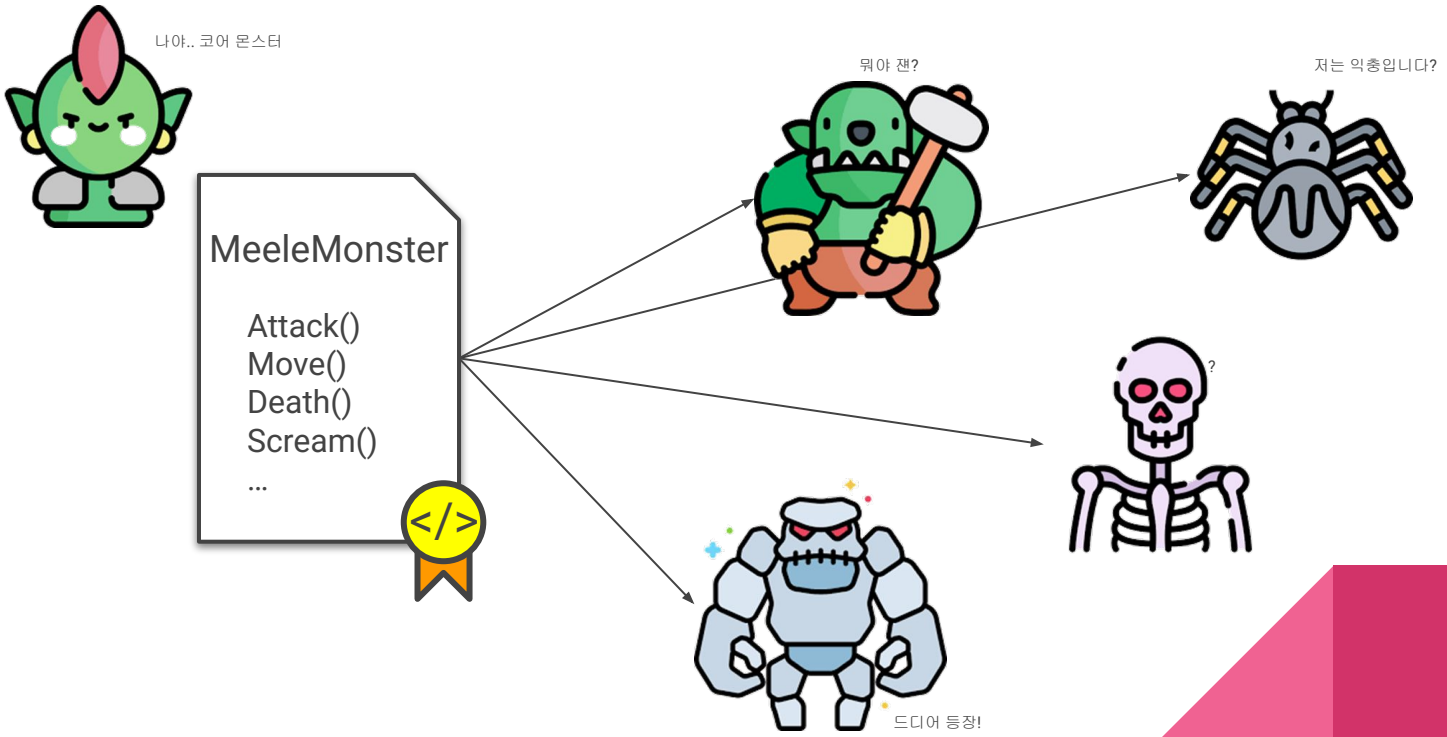
몬스터 - 인터페이스


나도 저거 쓰는데...


IDamagable
ITargetable
IAttackable
저기.. 총 무서운데여..


아너두?
Player

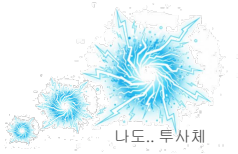
몬스터 - 접근 공격형 몬스터



몬스터 - 원거리 공격형 몬스터

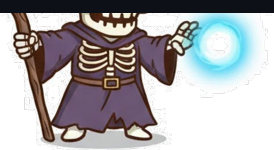
RangeMonster

Attack()
Move()
Death()
Scream()
...

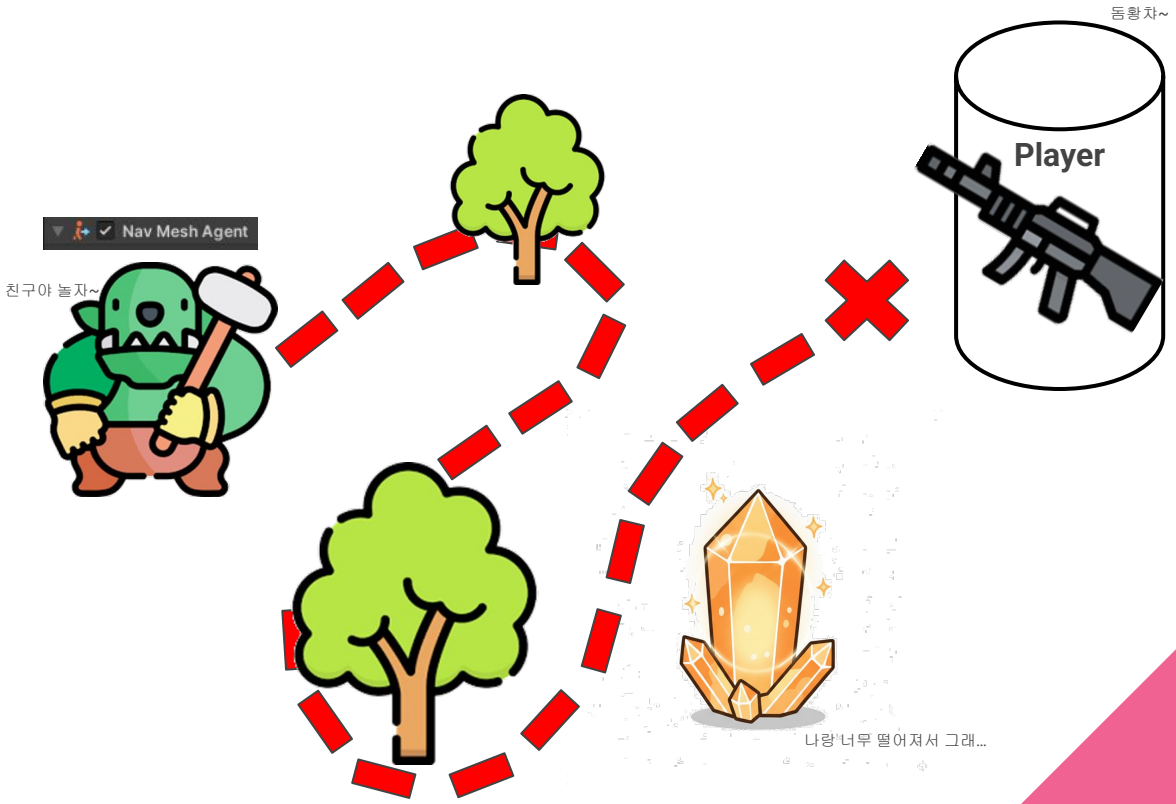


몬스터 - 원거리 공격형 몬스터

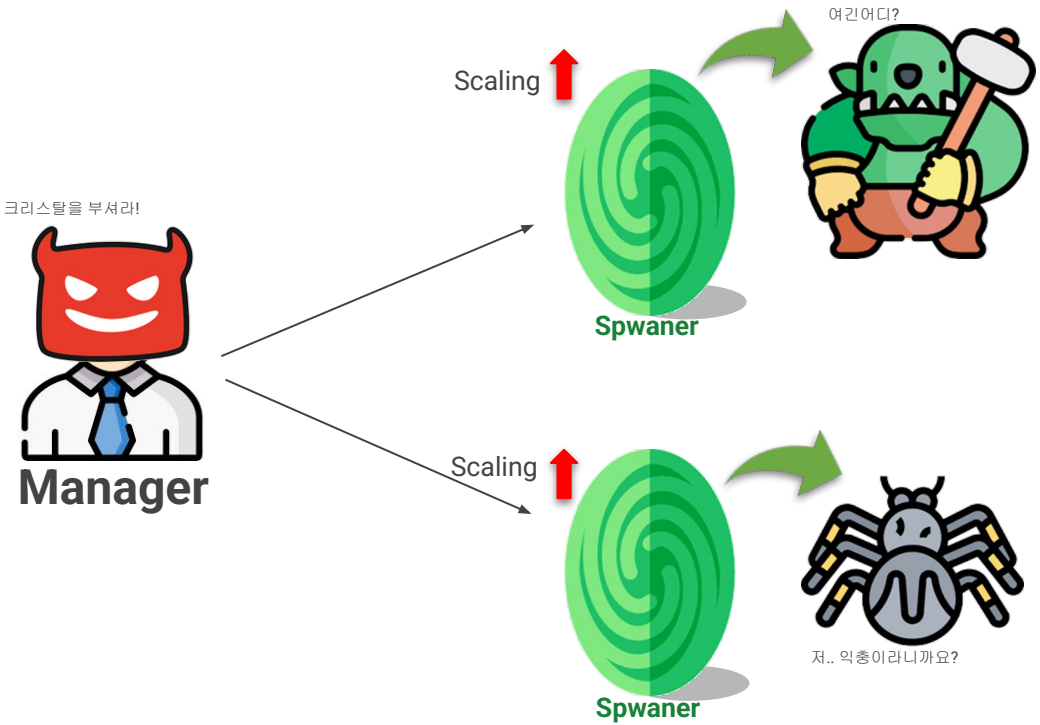
```
private void OnCollisionEnter(Collision other)
{
    if (!(other is IDamageable)) return;
    Debug.Log($"충돌 : {other.gameObject.name}");

    (other as IDamageable).TakeDamage(_shootingMonster.DamageCalc(_shootingMonster.Damage));
    other.gameObject.GetComponent<IDamageable>()?.TakeDamage(_shootingMonster.DamageCalc(_shootingMonster.Damage));
    Debug.Log($"{_shootingMonster.name} : 명중! {_shootingMonster.DamageCalc(_shootingMonster.Damage)} 피해");
    gameObject.SetActive(false);
}
```



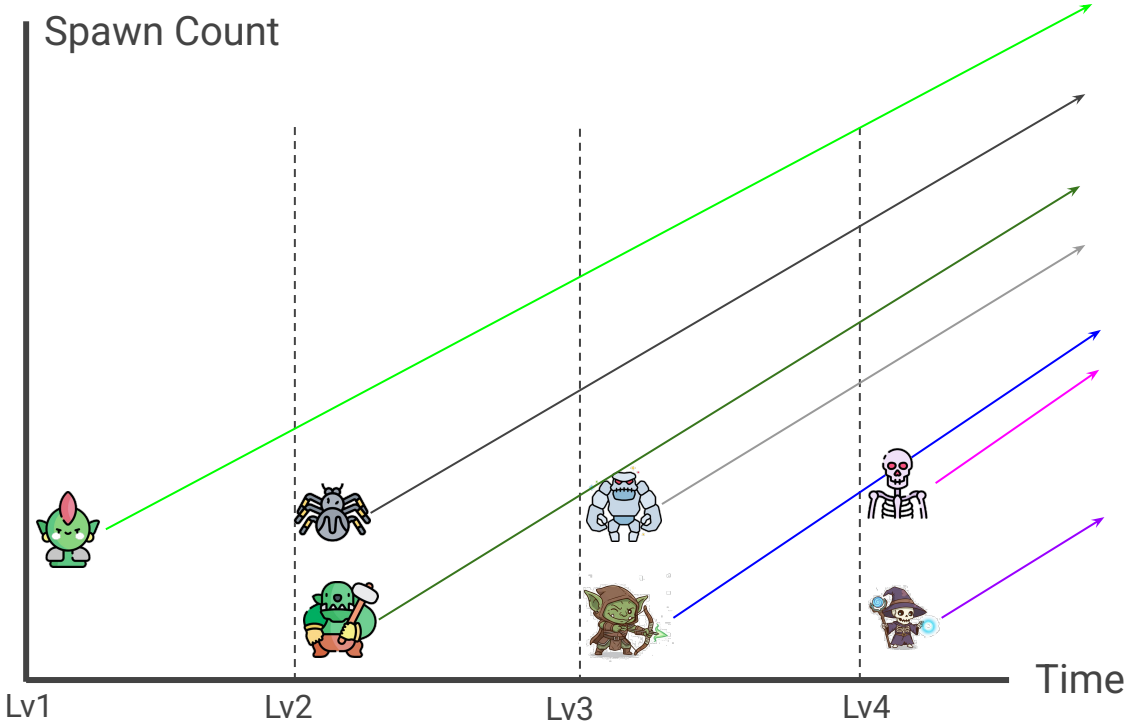
몬스터 - 몬스터 이동



몬스터 파트 작업 - 몬스터 소환

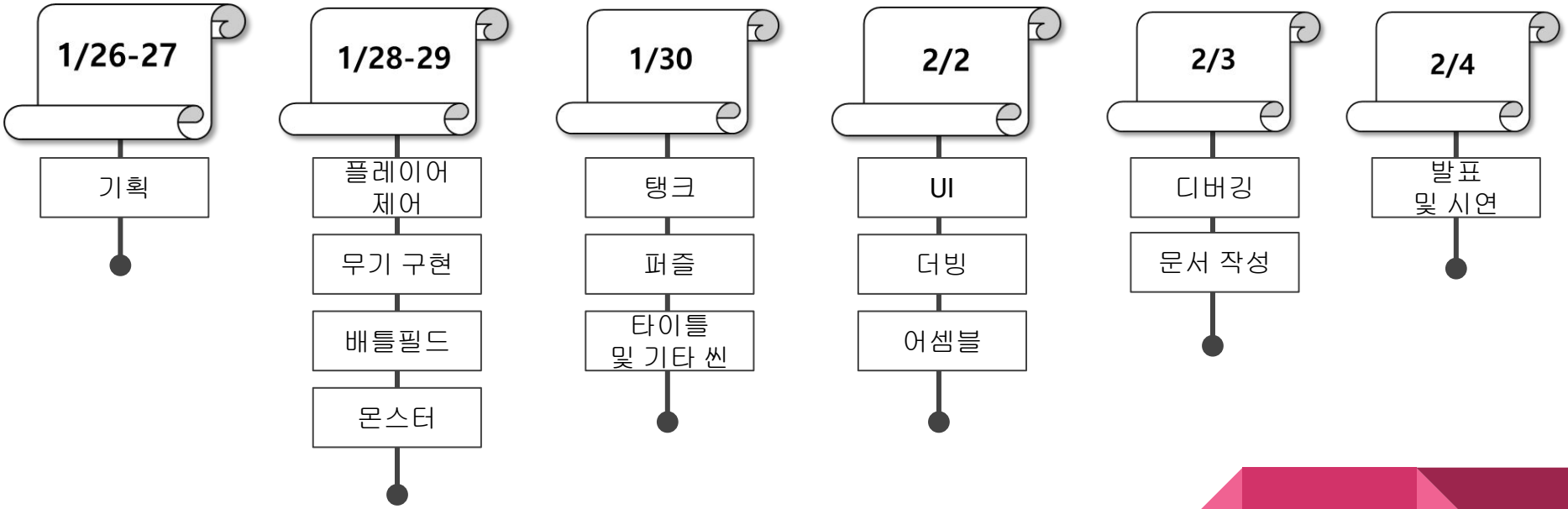


몬스터 - 소환 빈도

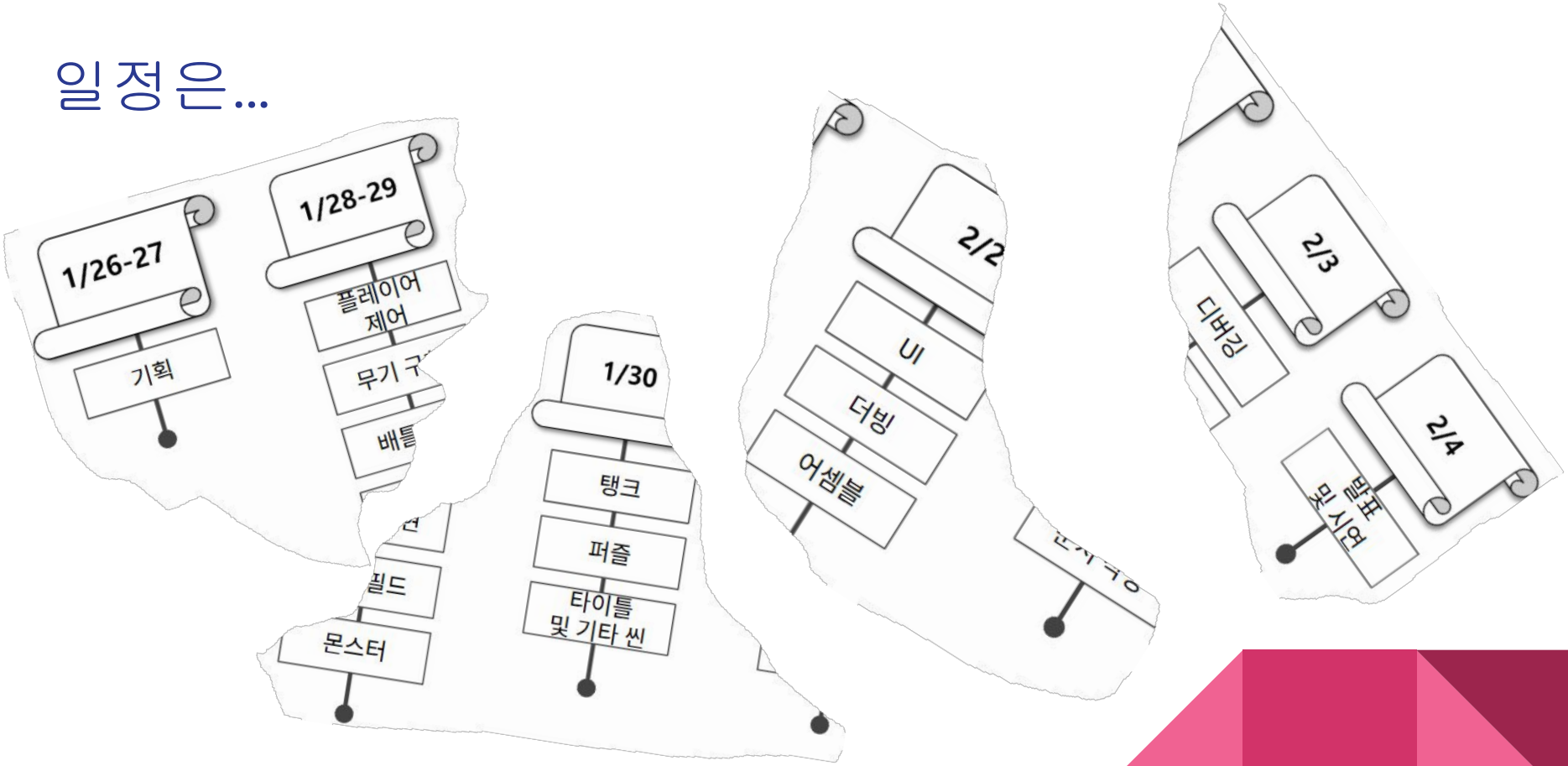


회 고

일정 계획


1/26-27
기획
1/28-29
플레이어 제어
무기 구현
배틀필드
몬스터
1/30
탱크
퍼즐
타이틀 및 기타 씬
2/2
UI
더빙
어셈블
2/3
디버깅
문서 작성
2/4
발표 및 시연

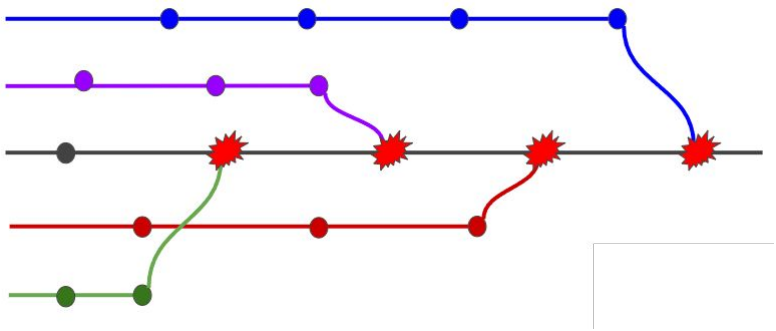
일정은...

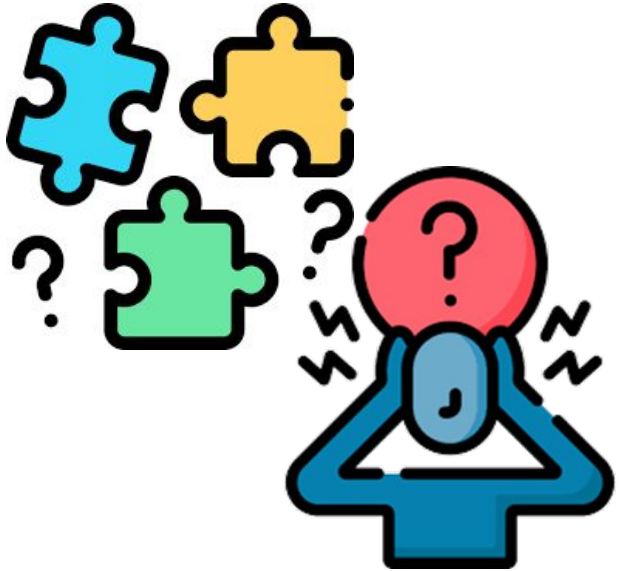

1/26-27
기획
1/28-29
플레이어 제어
무기 구
배틀
길드
몬스터
1/30
탱크
퍼즐
타이틀 및 기타 씬
2/2
UI
더빙
어셈블
2/3
디버깅
2/4
발표 및 시연

어쩔수 없는 선택..

협업

Git Flow



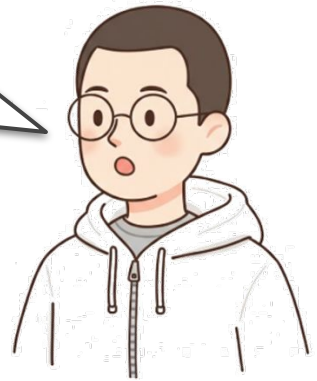


협업 (cont`)

우리들의 말

데이터 중심 소통 👍
코루틴 학습⬆
물리 엔진 이해도⬆
이벤트 함수 미숙 🫣

Git 미숙 🫣
아직 코딩이 어려움 🥲
실제 게임의 제작의 재미 😊
데이터 조립의 재미 😀



우리들의 말

코딩 외적인 기여
모르면 묻는 용기 ✊
친절한 강사님 ♥

이벤트 사용 숙지
문제 해결 능력
소통이야말로 힘 💪
어려운 Git 🤦
지식 부족에 의한 아쉬움
🥲

우리들의 말

Q & A

Thanks for your attention.