[적 AI 구현에 관련된 클래스에 대한 코드 설명]

1. **NavMeshAgent**

   - NavMeshAgent는 Unity에서 제공하는 강력한 AI 길찾기 시스템의 핵심 클래스.
   - 이 클래스는 게임 오브젝트가 NavMesh를 기반으로 환경 내에서 자동으로 이동할 수 있도록 도와준다.
     - NavMesh(내비게이션 메시 => 오브젝트 메시를 기반으로 생성된 움직일 수 있는 영역)

2. **MonsterAI**

   - 적을 FSM 패턴으로 구현하기 위해 직접 제작한 클래스이다.
   - 적이 게임의 환경이나 다른 오브젝트에 영향을 받아 상태가 변할 수 있도록 제작해 평소에는 랜덤 위치를 순찰하다 자신의 시야 범위에 플레이어가 나타나면 추격하게 된다. 벽 사이에 숨은 플레이어는 발견이 불가능하게 구현했다.

[코드 요약]

```
Unity 메시지 | 참조 0개
void Awake()
{
    hp = maxHp;
    nav = GetComponent<NavMeshAgent>();
    player = GameObject.FindWithTag("Player").transform;
}

Unity 메시지 | 참조 0개
void Update()
{
    if(Physics.Raycast(transform.position+new Vector3(0,0.5f,0), Vector3.down, 1)) // 몬스터가 아래가 땅일 때만 이동 가능하게 함.
    {
        nav.enabled = true;
    }
    if (state != MonsterSate.Die)
    {
        RandomMoveTime(); // 탐색 시 이동 시간 제어
        MonsterApproach(); // 몬스터가 플레이어와 마주쳤을때의 처리 결정
        MonsterActingState(); // 몬스터가 자신의 상태에 맞게 동작하도록 함.
    }
    else if(dieTime > 0) dieTime -= Time.deltaTime; // dieTime : 죽고나서 일정 시간 동안 대기하는 경우 사용

    if (dieTime <= 0 && dieTime != 52) // 52 : no repeat
    {
        dieTime = 52;
        if (!useGun) // 총을 든 적이 아닌 경우
        {
            GameObject eff = Instantiate(dieEffect, transform.position, Quaternion.identity); // 폭발 이펙트 생성
            Destroy(eff, eff.GetComponent<ParticleSystem>().main.duration);
        }
        Destroy(gameObject);
    }
}
```

   - Awake 함수에선 변수들의 초기값을 설정한다.
   - Update 함수에선 적의 이동 가능 여부와 죽음 이후의 처리하며 적이 스스로 행동하도록 유도하는 함수 MonsterApproach, MonsterActingState 등을 호출한다.

```
void MonsterApproach()
{
    Vector3 start;
    if (useGun)
    {
        start = shootPoint.position;
    }
    else
        start = transform.position;

    Debug.DrawRay(start, Vector3.Normalize(player.position - start) * sightRange);
    RaycastHit hit;

    if (player.GetComponent<Player>().state != Player.PlayerCurrentState.Die &&
        Physics.Raycast(start + new Vector3(0,sightHeight,0), player.position - start, out hit, sightRange,1)
        && hit.collider.gameObject.CompareTag("Player"))
    {
        if (Vector3.Magnitude(hit.point - start) <= attackDistance)
        {
            if (useGun && nav.enabled)
            {
                nav.isStopped = true;
            }
            if (nav.enabled && animator && state != MonsterSate.Attack) AnimationPlay("Attack");
            state = MonsterSate.Attack;
        }
        else
        {
            if (useGun && nav.enabled) nav.isStopped = false;
            if (animator && state != MonsterSate.Chase) AnimationPlay("Walk");
            state = MonsterSate.Chase;
        }
    }
    else
    {
        if (useGun && nav.enabled) nav.isStopped = false;
        if (animator && state == MonsterSate.Attack) AnimationPlay("Walk");
        state = MonsterSate.Patrol;
    }
}
```

- MonsterApproach 함수에서는 Raycast를 이용해 sightRange만큼 플레이어방향으로 레이저를 쏴 플레이어를 탐지한다. 다른 오브젝트가 아니라 플레이어가 탐지되면 적의 상태를 변경해서 다른 행동을 하도록 만든다.

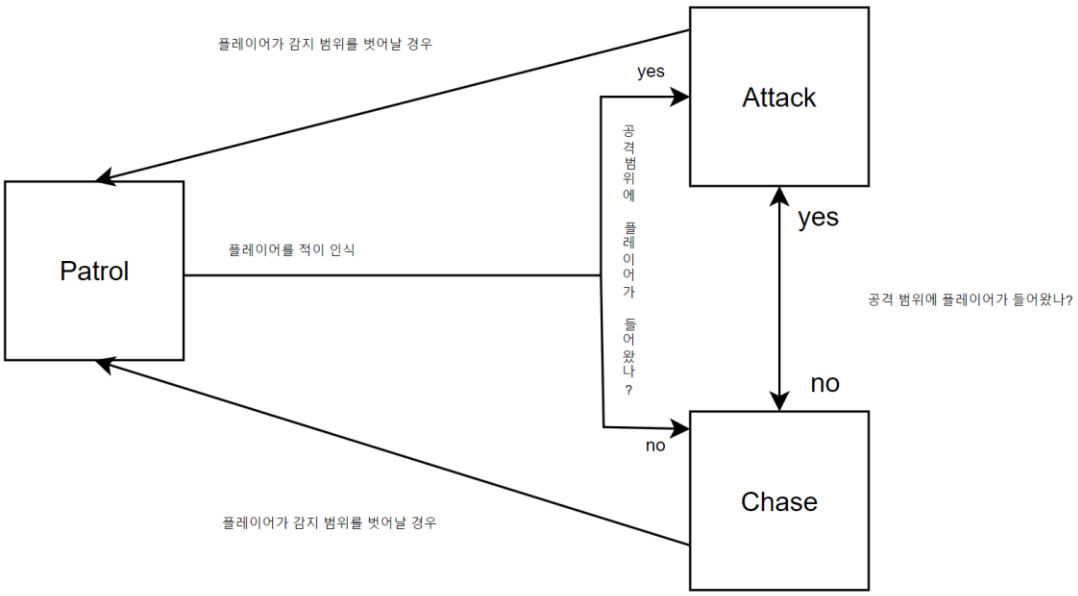
  ■ 이때 상태는 다음과 같이 변화한다.

    ◆ 플레이어가 공격 거리(attackDistance) 안에 들어오면 공격을 진행하는 Attack 상태로 변경하고, 그 외 상황은 플레이어를 추격하는 상태 Chase로 변경한다.

```
void MonsterActingState()
{
    if (hp != 0)
        switch (state)
        {
            case MonsterSate.Patrol:
                RandomMove();
                break;
            case MonsterSate.Chase:
                if(nav.enabled)
                nav.destination = player.position;
                break;
            case MonsterSate.Attack:
                transform.LookAt(player);
                break;
        }
    else
    {
        if(beforeDieEffect)
            beforeDieEffect.SetActive(true);
        state = MonsterSate.Die;
        animator.SetTrigger("Die");
        if (nav.enabled)
        {
            nav.isStopped = true;
            nav.enabled = false;
        }
        GetComponent<Collider>().enabled = false;
    }
}
```

- MonsterActingState 함수에서는 적이 살아있으면 자신의 상태 State에 맞게 행동하도록 함수를 호출한다. 기본적으로 적은 정찰 Patrol 상태이며, 해당 상태에선 랜덤 위치를 목적지로 설정해 해당 위치로 이동하도록 하였다.

[적 AI 의 FSM 구조]


플레이어가 감지 범위를 벗어날 경우
yes
Attack
공격범위에 플레이어가 들어왔나?
yes
공격 범위에 플레이어가 들어왔나?
Patrol
플레이어를 적이 인식
no
no
Chase
플레이어가 감지 범위를 벗어날 경우