```
// Moves the slime toward the player or home if not sleeping
void MoveSlime()
{
    if (sleep) return;

    float distToPlayer = Vector2.Distance(transform.position, thePlayer.transform.position);
    float distToHome = Vector2.Distance(transform.position, home.transform.position);

    if (distToPlayer < distanceAttack && distToHome < 60)
        ai.SetDestination(thePlayer.transform.position);
    else
        ai.SetDestination(home.transform.position);
}

// Performs sprint attack when slime is near the player
void AttackSprint()
{
    ai.speed = 0;

    if (cooldownAttack <= 0)
    {
        Vector2 direction = (thePlayer.transform.position - transform.position).normalized;
        transform.position += (Vector3)(direction * sprintAttack * Time.deltaTime);

        if (Vector2.Distance(transform.position, thePlayer.transform.position) < 3)
        {
            thePlayer_script.takeDMG(dmg, false);
            PlayAttackSound();
        }

        cooldownAttack = saveCooldownAttack;
    }
    else
    {
        cooldownAttack -= Time.deltaTime;
    }
}

// Plays the attack sound with cooldown
void PlayAttackSound()
{
    if (cooldownSound <= 0)
    {
        var sound = Instantiate(soundEffect_attack, transform.position, Quaternion.identity,
transform);
        Destroy(sound, 1);
        cooldownSound = saveCooldownSound;
    }
}

// Creates explosion effect for fire slime
void FireExplosionEffect()
{
    if (cooldownExplosion <= 0 && !sleep)
    {
        Instantiate(soundEffect_Getattack, transform.position, Quaternion.identity, transform);
        Instantiate(soundEffect_explosion, transform.position, Quaternion.identity, transform);
        Instantiate(fireExplosion, transform.position, Quaternion.identity, transform);

        sleep = true;
        cooldownExplosion = UnityEngine.Random.Range(saveCooldownExplosion * 0.6f,
saveCooldownExplosion);
    }
    else
    {
        cooldownExplosion -= Time.deltaTime;
    }
}

// Plays jump sound effect with cooldown
void JumpSoundEffect()
{
    if (cooldownJumpSound <= 0 && !sleep)
    {
        var sound = Instantiate(soundEffect_slimeJump, transform.position, Quaternion.identity,
transform);
        Destroy(sound, 2);
        cooldownJumpSound = saveCooldownJumpSound;
    }
    else
    {
        cooldownJumpSound -= Time.deltaTime;
    }
}
```