```
/// <summary>
    /// Updates health bar and reed signal
    /// </summary>
    private void UpdateUI()
    {
        hpSlider.value = hp;
        reedSignal.SetActive(seesPlayer);
    }

    /// <summary>
    /// Enemy moves towards the player
    /// </summary>
    private void FollowPlayer(float distanceToPlayer)
    {
        seesPlayer = true;
        ResetFollowCooldown();
        MoveTowards(player.transform.position);
        PlayWalkAnimation();
    }

    /// <summary>
    /// Enemy attacks player if in range
    /// </summary>
    private void AttackPlayer()
    {
        seesPlayer = true;
        StopWalkAnimation();

        if (currentAttackCooldown <= 0)
        {
            animator.SetTrigger("attack");
            currentAttackCooldown = attackCooldown;
            FlipTowards(player.transform.position);
            StartCoroutine(AttackTriggerCoroutine());
        }
    }

    /// <summary>
    /// Enemy patrols predefined positions
    /// </summary>
    private void Patrol()
    {
        seesPlayer = false;
        Vector3 target = patrolPositions[currentPatrolIndex].position;
        FlipTowards(target);
        MoveTowards(target);
        PlayWalkAnimation();

        if (Vector3.Distance(transform.position, target) < 0.5f)
            NextPatrolPoint();
    }

    /// <summary>
    /// Moves enemy towards target position
    /// </summary>
    private void MoveTowards(Vector3 target)
    {
        Vector3 direction = (target - transform.position).normalized;
        direction.y = 0;
        direction.z = 0;
        transform.position += direction * moveSpeed * Time.deltaTime;
    }

    /// <summary>
    /// Switch to next patrol position
    /// </summary>
    private void NextPatrolPoint()
    {
        currentPatrolIndex = (currentPatrolIndex + 1) % patrolPositions.Length;
    }

    /// <summary>
    /// Flips enemy sprite to face target
    /// </summary>
    private void FlipTowards(Vector3 target)
    {
        if (transform.position.x > target.x)
            animator.gameObject.transform.rotation = Quaternion.Euler(0, 180, 0);
        else
            animator.gameObject.transform.rotation = Quaternion.Euler(0, 0, 0);
    }

    /// <summary>
    /// Handles attack trigger and sound
    /// </summary>
    private IEnumerator AttackTriggerCoroutine()
    {
        yield return new WaitForSeconds(triggerAttackOnDelay);
        triggerAttack.SetActive(true);
        mainAttackClip.Play();
        yield return new WaitForSeconds(triggerAttackOffDelay);
        triggerAttack.SetActive(false);
    }

    /// <summary>
    /// Handles enemy receiving damage
    /// </summary>
    public void TakeDamage(float damage)
    {
        bloodTakeDmgClip.Play();
        hp -= damage;
    }

    /// <summary>
    /// Handles death and giving XP
    /// </summary>
    private void HandleDeath()
    {
        animator.SetBool("walk", false);
        animator.SetBool("death", true);
        walkSound.Stop();

        if (!gaveXp)
        {
            playerStats.KillAnEnemy(attackPower);
            gaveXp = true;
        }

        Destroy(gameObject, 5f);
    }

    /// <summary>
    /// Updates cooldown timers
    /// </summary>
    private void UpdateCooldowns()
    {
        if (currentAttackCooldown > 0) currentAttackCooldown -= Time.deltaTime;
        if (seesPlayer && currentFollowCooldown > 0) currentFollowCooldown -= Time.deltaTime;
        if (currentFollowCooldown <= 0) seesPlayer = false;
    }

    /// <summary>
    /// Reset follow cooldown
    /// </summary>
    private void ResetFollowCooldown()
    {
        currentFollowCooldown = followResetCooldown;
    }

    /// <summary>
    /// Plays walk animation and sound
    /// </summary>
    private void PlayWalkAnimation()
    {
        animator.SetBool("walk", true);
        if (!walkSound.isPlaying) walkSound.Play();
    }

    /// <summary>
    /// Stops walk animation and sound
    /// </summary>
    private void StopWalkAnimation()
    {
        animator.SetBool("walk", false);
        walkSound.Stop();
    }
```