```
/// <summary>
    /// Initializes references and default values.
    /// </summary>
    private void InitializeReferences()
    {
        slider_boss_hp.maxValue = boss_hp;
        attack_sound_effect.Stop();

        thePlayer = GameObject.FindGameObjectWithTag("Player");
        statsPlayer_script = thePlayer.GetComponent<statsPlayer>();
        mainCharacterController_script = thePlayer.GetComponent<mainCharacterController>();
        boss_animator = the_boss.GetComponent<Animator>();

        save_cooldown_for_attack = cooldown_for_attack;
        save_cooldown_for_move = cooldown_for_move;
    }

    /// <summary>
    /// Updates the boss HP slider.
    /// </summary>
    private void UpdateUI()
    {
        slider_boss_hp.value = boss_hp;
    }

    /// <summary>
    /// Handles boss movement between points and attack logic.
    /// </summary>
    private void HandleMovementAndAttack()
    {
        if (!bool_for_move)
            StartCoroutine(MoveCooldownRoutine());

        MoveToTarget();
        FlipTowardsPlayer();
        TryAttack();
    }

    /// <summary>
    /// Moves boss towards current target position.
    /// </summary>
    private void MoveToTarget()
    {
        Transform target = all_positions[currentIndex];
        float distance = Vector3.Distance(the_boss.transform.position, target.position);

        if (distance > 5)
        {
            float speed = attack_on_move ? walks_speed * 10 : walks_speed;
            the_boss.transform.position += (target.position -
the_boss.transform.position).normalized * speed * Time.deltaTime;
        }
        else
        {
            attack_on_move = false;
        }
    }

    /// <summary>
    /// Flips boss sprite to face player.
    /// </summary>
    private void FlipTowardsPlayer()
    {
        if (!attack_on_move)
        {
            the_sprite.transform.localRotation = the_boss.transform.position.x <
thePlayer.transform.position.x
                ? Quaternion.Euler(0, 0, 0)
                : Quaternion.Euler(0, 180, 0);
        }
    }

    /// <summary>
    /// Checks and triggers boss attack.
    /// </summary>
    private void TryAttack()
    {
        if (!coollider_for_attack.activeSelf && !fix_attack)
        {
            fix_attack = true;
            boss_animator.SetTrigger("skill_1");
            attack_sound_effect.Play();
            StartCoroutine(AttackCooldownRoutine());
        }
    }

    /// <summary>
    /// Coroutine for enabling/disabling attack collider.
    /// </summary>
    private IEnumerator AttackCooldownRoutine()
    {
        yield return new WaitForSeconds(0.6f);
        coollider_for_attack.SetActive(true);
        yield return new WaitForSeconds(0.25f);
        coollider_for_attack.SetActive(false);
        fix_attack = false;
    }

    /// <summary>
    /// Handles boss death behavior.
    /// </summary>
    private void HandleDeath()
    {
        if (!boss_death)
        {
            boss_death = true;
            boss_animator.SetTrigger("death");
            all_sound_effects.SetActive(false);
        }

        if (bg_music != null)
            bg_music.volume -= Time.deltaTime * 0.3f;
    }

    /// <summary>
    /// Coroutine for boss movement cooldown and random target selection.
    /// </summary>
    private IEnumerator MoveCooldownRoutine()
    {
        bool_for_move = true;
        SetNextMoveIndex();
        yield return new WaitForSeconds(Random.Range(cooldown_for_move * 0.6f,
cooldown_for_move));
        bool_for_move = false;
        attack_on_move = false;
    }

    /// <summary>
    /// Chooses next move index and determines if attack_on_move happens.
    /// </summary>
    private void SetNextMoveIndex()
    {
        currentIndex++;
        if (currentIndex >= all_positions.Count)
        {
            currentIndex = change ? Random.Range(0, 4) : Random.Range(4, all_positions.Count -
1);
            change = !change;
        }

        attack_on_move = Random.Range(0, 3) > 1;
    }

    /// <summary>
    /// Coroutine to spawn projectiles at random intervals.
    /// </summary>
    private IEnumerator GenerateProjectilesRoutine()
    {
        int selector = Random.Range(1, 4);

        for (int i = 0; i < selector; i++)
        {
            GameObject proj = Instantiate(all_projectiles[Random.Range(0,
all_projectiles.Count)]);
            proj.transform.SetParent(position_for_projectiles[projectileIndex]);
            proj.transform.localPosition = Vector3.zero;
            yield return new WaitForSeconds(Random.Range(1.5f, 3));
            projectileIndex++;
        }

        yield return new WaitForSeconds(Random.Range(2, 3));
        projectileIndex = 0;

        if (boss_hp > 0)
            StartCoroutine(GenerateProjectilesRoutine());
    }

    /// <summary>
    /// Detects collision with player attacks.
    /// </summary>
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("player_attack"))
            TakeDamage();
    }

    /// <summary>
    /// Reduces boss HP when hit by player.
    /// </summary>
    private void TakeDamage()
    {
        boss_hp -= mainCharacterController_script.power_of_player;
    }

    /// <summary>
    /// Adjust camera field of view and slight shake during attacks.
    /// </summary>
    private void UpdateCameraEffect()
    {
        float distance_to_player = Vector3.Distance(the_boss.transform.position,
thePlayer.transform.position);

        // Zoom camera based on distance
        cam.fieldOfView = Mathf.Clamp(cam.fieldOfView + (distance_to_player >= 22.5f ? 5 : -5) *
Time.deltaTime, 100f, 120f);

        // Small camera shake during attack
        if (attack_on_move)
        {
            Vector3 shake = camera_move ? new Vector3(10, 0, 0) : new Vector3(-10, 0, 0);
            cam.transform.localPosition += shake * Time.deltaTime;
            StartCoroutine(SetCameraMoveBool());
        }
    }

    /// <summary>
    /// Coroutine to toggle camera movement boolean.
    /// </summary>
    private IEnumerator SetCameraMoveBool()
    {
        yield return new WaitForSeconds(0.05f);
        camera_move = !camera_move;
    }
```