

○○○

```
/// <summary>Handle player input for movement (left/right)</summary>
private void HandleInput()
{
    directionX = 0;
    if (!isDead && !inConversation && !isBlocking)
    {
        if (Input.GetKey(KeyCode.D)) { directionX = 1; spriteRenderer.flipX = false; }
        else if (Input.GetKey(KeyCode.A)) { directionX = -1; spriteRenderer.flipX = true; }
    }
}

/// <summary>Apply player movement and gravity using CharacterController</summary>
private void HandleMovement()
{
    Vector3 move = transform.right * directionX * (isRunning ? runSpeed : walkSpeed);
    move.y = velocityY;
    cc.Move(move * Time.deltaTime);

    if (!isGrounded) velocityY -= gravity * Time.deltaTime;
}

/// <summary>Handle jumping logic if on ground and enough stamina</summary>
private void HandleJump()
{
    if (Input.GetKeyDown(KeyCode.F) && (isGrounded || statPlayer.stamina_player > jumpForce * 0.3f))
    {
        velocityY = jumpForce;
        statPlayer.stamina_player -= jumpForce * 0.3f;
        PlaySound("jump");
    }
}

/// <summary>Handle sprinting logic and stamina consumption/regeneration</summary>
private void HandleSprint()
{
    isRunning = Input.GetKey(KeyCode.Space) && directionX != 0 && canUseStamina;
    if (isRunning) statPlayer.stamina_player -= Time.deltaTime * 2.5f;
    else if (statPlayer.stamina_player < statPlayer.stats_Player[1].maxValue)
        statPlayer.stamina_player += Time.deltaTime * statPlayer.regen_stamina;
}

/// <summary>Handle attacking logic, cooldown, hitboxes, and particle effects</summary>
private void HandleAttack()
{
    if (currentAttackCooldown > 0) { currentAttackCooldown -= Time.deltaTime; return; }

    if (Input.GetMouseButtonDown(0) && statPlayer.stamina_player > 2 && !isDead && !inConversation)
    {
        animator.SetTrigger("attack");
        currentAttackCooldown = attackCooldown;
        statPlayer.stamina_player -= 10;

        // Enable proper attack hitbox depending on facing direction
        if (spriteRenderer.flipX) { leftAttack.SetActive(true); rightAttack.SetActive(false); }
        else { leftAttack.SetActive(false); rightAttack.SetActive(true); }

        PlaySound("mainAttack");
        SpawnAttackParticles();
    }
}

/// <summary>Handle blocking (guard) using right mouse button</summary>
private void HandleBlock()
{
    if (Input.GetMouseButtonDown(1)) isBlocking = true;
    if (Input.GetMouseButtonUp(1)) isBlocking = false;

    animator.SetBool("guard", isBlocking);
}

/// <summary>Update animation states based on movement, sprint, jump, and death</summary>
private void UpdateAnimations()
{
    animator.SetBool("walk", directionX != 0 && !isRunning);
    animator.SetBool("run", isRunning);
    animator.SetBool("jump", !isGrounded);
    animator.SetBool("death", isDead);
}

/// <summary>Play walking or running sound based on movement and ground status</summary>
private void HandleSounds()
{
    if (directionX != 0 && isGrounded)
    {
        if (!walkRunAudio.isPlaying) walkRunAudio.Play();
        walkRunAudio.pitch = isRunning ? 1.5f : 1.2f;
    }
    else walkRunAudio.Stop();
}

/// <summary>Reduce attack chromatic aberration effect over time</summary>
private void UpdateEffects()
{
    if (attackEffect.intensity.value > 0) attackEffect.intensity.value -= Time.deltaTime;
}

/// <summary>Spawn particle effect for attack</summary>
private void SpawnAttackParticles()
{
    GameObject ps = Instantiate(attackParticle, transform);
    ps.transform.localPosition = Vector3.up * 0.6f;
    ps.transform.localScale *= 3.5f;
    Destroy(ps, 3f);
}

/// <summary>Play a specific sound clip by name</summary>
/// <param name="name">Name of the sound: "jump", "mainAttack", "sprintAttack", "takeDmg"</param>
public void PlaySound(string name)
{
    switch (name)
    {
        case "jump": jumpClip.Play(); break;
        case "mainAttack": mainAttackClip.Play(); break;
        case "sprintAttack": sprintAttackClip.Play(); break;
        case "takeDmg": takeDmgClip.Play(); break;
    }
}

/// <summary>Detect if player has touched the ground</summary>
/// <param name="other">Collider of the object entered</param>
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("ground")) isGrounded = true;
}

/// <summary>Detect if player left the ground</summary>
/// <param name="other">Collider of the object exited</param>
private void OnTriggerExit(Collider other)
{
    if (other.CompareTag("ground")) isGrounded = false;
}
```