```
/// <summary>
/// Handles player directional movement including forward, backward,
/// sideways and applies gravity and bonus speed multipliers
/// </summary>
private void HandleMovement()
{
    if (isGround && velocityY < 0) velocityY = -2f;

    move = (transform.forward * directionY + transform.right * directionX) * (speedPlayer +
testing_speed);
    move.y = velocityY;

    if (!inConversation && !you_are_death)
    {
        if (can_do_dodge)
        {
            cc.Move((move * bonusSpeed) * Time.deltaTime);
        }
        else
        {
            dodge_move = (transform.right * direction_for_dodge) * speedPlayer;
            dodge_move.y = velocityY;
            cc.Move((dodge_move * dodge_speed) * Time.deltaTime);
        }
    }

    velocityY -= gravity * Time.deltaTime;
}

/// <summary>
/// Handles player rotation based on mouse input,
/// applying sensitivity and clamping vertical rotation
/// </summary>
private void HandleRotation()
{
    if (!playerAttack_.get_mainInventory_is_Open() && !you_are_death)
    {
        rotationX += Input.GetAxis("Mouse X") * (mouseSensitive * Time.deltaTime);
        transform.rotation = Quaternion.Euler(0, rotationX, 0);

        rotationY += Input.GetAxis("Mouse Y") * (mouseSensitive * Time.deltaTime);
        rotationY = Mathf.Clamp(rotationY, -70, 70);
        mainCamera.rotation = Quaternion.Euler(rotationY * -1, rotationX, 0);
    }
}

/// <summary>
/// Handles sprinting input, stamina consumption,
/// and adjusts player speed accordingly
/// </summary>
private void HandleSprint()
{
    if (Input.GetKey(KeyCode.LeftShift))
    {
        statsPlayer_.cooldown_to_regen_the_stamina = 2;
        speedPlayer = sprint;
        if (statsPlayer_.numbers_of_enemy_is_in_battle_with_player > 0)
        {
            statsPlayer_.all_curentStats[1] -= Time.deltaTime * 7f;
        }
    }
    else
    {
        speedPlayer = saveSpeed;
    }
}

/// <summary>
/// Handles jumping input and applies jump force
/// if player is on the ground and not dodging
/// </summary>
private void HandleJump()
{
    if (Input.GetKeyDown(KeyCode.Space) && isGround && can_do_dodge && directionX != 0 &&
directionY == 0 && !you_are_death)
    {
        StartCoroutine(dodgeCooldown(directionX));
    }
    else if (Input.GetKeyDown(KeyCode.Space) && isGround && !you_are_death)
    {
        velocityY = jumpForce;
    }
}

/// <summary>
/// Detects interactable objects in front of player using raycast
/// and enables interaction icon or triggers chest interaction
/// </summary>
private void HandleInteraction()
{
    RaycastHit hit;
    if (Physics.Raycast(mainCamera.position, mainCamera.rotation * Vector3.forward, out hit,
range_Interact, layerMask_treasure))
    {
        if (Input.GetKeyDown(KeyCode.E) && !you_are_death)
        {
            GameObject gameObject_interact = hit.collider.gameObject;
            if (gameObject_interact.CompareTag("chest"))
            {
                chestInteract(gameObject_interact);
            }
        }
        if (!set_ico_for_interact)
        {
            E_Interact.SetActive(true);
            set_ico_for_interact = true;
        }
    }
    else
    {
        if (set_ico_for_interact)
        {
            E_Interact.SetActive(false);
            set_ico_for_interact = false;
        }
    }
}

/// <summary>
/// Updates movement animation and walking sound effect
/// depending on current speed, sprint and ground state
/// </summary>
private void HandleMovementAnimationAndSound()
{
    if ((directionX != 0 || directionY != 0))
    {
        if (speedPlayer == sprint)
        {
            if (inCombat && !test)
            {
                statsPlayer_.all_curentStats[1] -= sprint * Time.deltaTime;
                statsPlayer_.staminaRegen = false;
                statsPlayer_.cooldwonStamina = statsPlayer_.save_cooldwonStamina;
            }
            bounsAnimator.SetBool("walk", true);
            bounsAnimator.speed = 3f;
            if (isGround)
            {
                if (!walk_sound_effect.isPlaying) walk_sound_effect.Play();
                walk_sound_effect.pitch = 1.5f;
            }
            else walk_sound_effect.Stop();
        }
        else
        {
            bounsAnimator.SetBool("walk", true);
            bounsAnimator.speed = 1.5f;
            if (isGround)
            {
                if (!walk_sound_effect.isPlaying) walk_sound_effect.Play();
                walk_sound_effect.pitch = 1.05f;
            }
            else walk_sound_effect.Stop();
        }
    }
    else
    {
        bounsAnimator.SetBool("walk", false);
        bounsAnimator.speed = 1;
        walk_sound_effect.Stop();
    }
}

/// <summary>
/// Reads WASD input and updates directional values
/// for forward/backward and left/right movement
/// </summary>
private void HandleDirectionalInput()
{
    if (Input.GetKey(KeyCode.W)) directionY = 1;
    else if (Input.GetKey(KeyCode.S)) directionY = -1;
    else directionY = 0;

    if (Input.GetKey(KeyCode.D)) directionX = 1;
    else if (Input.GetKey(KeyCode.A)) directionX = -1;
    else directionX = 0;
}
```