```
///<Summary>
/// Handles reading player input for movement and rotation.
/// Updates move and rotate vectors for the current frame.
///</Summary>
private void HandlePlayerInput()
{
    _movePlayer = _moveInput.action.ReadValue<Vector2>();
    _rotatePlayer = _mouseInput.action.ReadValue<Vector2>();

    if (_runSpeed == 0)
        _cameraPeeking = _cameraPeekingInput.action.ReadValue<float>();
    else
        _cameraPeeking = 0;
}

///<Summary>
/// Handles sprint logic and stamina consumption.
/// Activates or deactivates running speed.
///</Summary>
private void HandleSprint()
{
    if (_sprintInput.action.WasPressedThisFrame() && _sliderStamina.value > 0 && _canSprint
        && (_movePlayer.x != 0 || _movePlayer.y != 0))
        _runSpeed = _saveRunSpeed;
    else if (_sprintInput.action.WasReleasedThisFrame() || !_canSprint)
        _runSpeed = 0;

    if (_runSpeed > 0 && (_movePlayer.x != 0 || _movePlayer.y != 0))
        _sliderStamina.value -= _subtractStamina * Time.deltaTime;
    else if (_canRegenStamina)
        _sliderStamina.value += (_subtractStamina * 1.5f) * Time.deltaTime;
}

///<Summary>
/// Updates camera peeking animations based on player input.
/// Sets animator booleans for left or right peeking.
///</Summary>
private void UpdateCameraPeeking()
{
    CameraPeekingState state = CameraPeekingState.Stop;
    if (_cameraPeeking == -1) state = CameraPeekingState.Right;
    else if (_cameraPeeking == 1) state = CameraPeekingState.Left;

    _cameraPeekingAnimator.SetBool("left", state == CameraPeekingState.Left);
    _cameraPeekingAnimator.SetBool("right", state == CameraPeekingState.Right);
}

///<Summary>
/// Handles walking and running animations, movement, and doll detection.
/// Adjusts player velocity based on input and speed modifiers.
///</Summary>
private void HandleMovement()
{
    if (_movePlayer.x != 0 || _movePlayer.y != 0)
    {
        Vector3 move = (transform.forward * ((_movePlayer.y > 0) ? _movePlayer.y : _movePlayer.y
* 0.6f))
                    + (transform.right * _movePlayer.x * 0.6f);

        SetMovePlayer(move, (_walkSpeed + _runSpeed) * GetSpeedOfCameraPeeking());

        UpdateAnimationsAndDollDetection();
    }
    else
    {
        MainCameraAnimator.SetBool("walk", false);
        MainCameraAnimator.SetBool("run", false);
        GameEngineScript.CheckDollDistance(transform, _detectionRangeIdle, _detectionRangeIdle,
true);
    }
}

///<Summary>
/// Updates camera and player rotation based on mouse input.
/// Clamps pitch to prevent excessive vertical rotation.
///</Summary>
private void HandleRotation()
{
    float rotatePitch = _rotatePlayer.y * (_mouseSensitivity +
_mouseSensitivityManager._mouseSensitiveValue) * Time.deltaTime;
    float rotateYaw = _rotatePlayer.x * (_mouseSensitivity +
_mouseSensitivityManager._mouseSensitiveValue) * Time.deltaTime;

    _cameraPitch += rotatePitch;
    _cameraYaw += rotateYaw;
    _cameraPitch = Mathf.Clamp(_cameraPitch, -_pitchLimit, _pitchLimit);

    transform.localRotation = Quaternion.Euler(0, _cameraYaw, 0);
    MainCameraAnimator.transform.localRotation = Quaternion.Euler(-_cameraPitch, 0, 0);
}

///<Summary>
/// Moves the player character using CharacterController.
/// Takes a move vector and speed multiplier as input.
///</Summary>
public void SetMovePlayer(Vector3 move, float speed)
{
    move.y = 0;
    if (_cc.enabled)
        _cc.Move(move * (speed * _speedTest) * Time.deltaTime);
}

///<Summary>
/// Handles stamina cooldown after sprinting.
/// Prevents sprinting until cooldown completes.
///</Summary>
private IEnumerator CooldownRegenStamina()
{
    _canSprint = false;
    _canRegenStamina = false;
    yield return new WaitForSeconds(2);
    _canRegenStamina = true;
    yield return new WaitForSeconds(1);
    _canSprint = true;
}

///<Summary>
/// Toggles debug speed for testing purposes.
/// Can increase player movement multiplier temporarily.
///</Summary>
private void SetSpeedTest(InputAction.CallbackContext ctx)
{
    switch (_speedTest)
    {
        case 1: _speedTest = 4; break;
        default: _speedTest = 1; break;
    }
}

///<Summary>
/// Adjusts player animation speeds and doll detection ranges
/// based on whether the player is walking or running.
///</Summary>
private void UpdateAnimationsAndDollDetection()
{
    if (_runSpeed == _saveRunSpeed)
    {
        MainCameraAnimator.SetBool("run", true);
        MainCameraAnimator.SetBool("walk", false);
        GameEngineScript.CheckDollDistance(transform, _detectionRangeRun, _detectionRangeRun,
true);
    }
    else
    {
        MainCameraAnimator.SetBool("run", false);
        MainCameraAnimator.SetBool("walk", true);
        GameEngineScript.CheckDollDistance(transform, _detectionRangeWalk, _detectionRangeWalk,
true);
    }
}
```