```
/// <summary>
    /// Regenerates player health based on distance from "home"
    /// </summary>
    private void HandleHealing()
    {
        if (mainHome == null) return;

        float distanceFromHome = Vector2.Distance(mainHome.transform.position,
transform.position);
        float healAmount = Mathf.Lerp(maxHeal, minHeal, distanceFromHome / maxDistance);
        healAmount = Mathf.Clamp(healAmount, minHeal, maxHeal);

        if (currentHP < maxHP)
        {
            currentHP += healAmount * Time.deltaTime;
            currentHP = Mathf.Min(currentHP, maxHP);
        }
    }

    /// <summary>
    /// Controls the freeze effect (changes color and timer)
    /// </summary>
    private void HandleFreezeEffect()
    {
        if (freezeTimer <= 0)
        {
            Color newColor = animatorPlayerScript.sp.color;
            newColor.r = 1f;  // normal color
            animatorPlayerScript.sp.color = newColor;
        }
        else
        {
            freezeTimer -= Time.deltaTime;
        }
    }

    /// <summary>
    /// Sets attack type in animator (for animations)
    /// </summary>
    private void HandleAttackType()
    {
        animatorPlayerScript.animator.SetInteger("typeAttack", typeAttack);
    }

    /// <summary>
    /// Reads input for movement direction and sets movement vector
    /// </summary>
    private void HandleMovementInput()
    {
        if (cooldownTimer > 0)
        {
            cooldownTimer -= Time.deltaTime;
            movement = Vector2.zero;
            return;
        }

        float moveX = 0f;
        float moveY = 0f;

        if (Input.GetKey(KeyCode.W)) moveY += 1f;
        if (Input.GetKey(KeyCode.S)) moveY -= 1f;
        if (Input.GetKey(KeyCode.D)) moveX += 1f;
        if (Input.GetKey(KeyCode.A)) moveX -= 1f;

        movement = new Vector2(moveX, moveY).normalized;
    }

    /// <summary>
    /// Moves the player physically using Rigidbody2D
    /// </summary>
    private void MovePlayer()
    {
        Vector2 newPosition = rb.position + movement * speed * Time.fixedDeltaTime;
        rb.MovePosition(newPosition);
    }

    /// <summary>
    /// Applies damage to the player and handles freeze effect if applicable
    /// </summary>
    public void TakeDamage(int dmg, bool freeze)
    {
        if (!playerAttackScript.attack)
        {
            currentHP -= dmg * Time.deltaTime;
            cameraDMG.SetTrigger("takeDMG");
            cooldownTimer = cooldown;
        }
        else
        {
            Debug.Log("Shield active, no damage taken");
        }

        if (freeze)
        {
            ApplyFreezeEffect();
        }
    }

    /// <summary>
    /// Applies freeze effect (changes color and plays sound)
    /// </summary>
    private void ApplyFreezeEffect()
    {
        Color newColor = animatorPlayerScript.sp.color;
        newColor.r = 0f;
        animatorPlayerScript.sp.color = newColor;

        freezeTimer = Random.Range(freezeCooldown * 0.6f, freezeCooldown);

        GameObject effect = Instantiate(soundEffectFreezPrefab);
        Destroy(effect, 2f);
    }

    /// <summary>
    /// Detects when player enters triggers and applies relevant effects
    /// </summary>
    private void OnTriggerEnter2D(Collider2D other)
    {
        switch (other.gameObject.tag)
        {
            case "green_cristals":
                mainUI.IncreaseCrystalGreen(1);
                PlayCrystalEffect(other.gameObject);
                break;
            case "fire_cristals":
                mainUI.IncreaseCrystalFire(1);
                PlayCrystalEffect(other.gameObject);
                break;
            case "ice_cristals":
                mainUI.IncreaseCrystalIce(1);
                PlayCrystalEffect(other.gameObject);
                break;
            case "gold_cristals":
                mainUI.IncreaseCrystalGold(1);
                PlayCrystalEffect(other.gameObject);
                break;
            case "tarc":
                nearSlimePen = true;
                break;
        }
    }

    /// <summary>
    /// Plays crystal collection effect and destroys the crystal object
    /// </summary>
    private void PlayCrystalEffect(GameObject crystal)
    {
        Destroy(crystal);
        GameObject effect = Instantiate(soundEffectCristalsPrefab);
        Destroy(effect, 2f);
    }

    /// <summary>
    /// Detects when player exits the "tarc" trigger
    /// </summary>
    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("tarc"))
        {
            nearSlimePen = false;
        }
    }

    /// <summary>
    /// Updates reference to the closest GameObject with tag "slime_pen"
    /// </summary>
    private void UpdateClosestSlimePen()
    {
        closestSlimePen = GetClosestObjectWithTag("slime_pen");
    }

    /// <summary>
    /// Finds the closest GameObject with the specified tag
    /// </summary>
    public GameObject GetClosestObjectWithTag(string tag)
    {
        GameObject[] objects = GameObject.FindGameObjectsWithTag(tag);

        GameObject closest = null;
        float minDistance = Mathf.Infinity;
        Vector3 playerPos = transform.position;

        foreach (var obj in objects)
        {
            float dist = Vector3.Distance(playerPos, obj.transform.position);
            if (dist < minDistance)
            {
                minDistance = dist;
                closest = obj;
            }
        }
        return closest;
    }
```