```
// PlayerAnime.cs

using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System;

public class PlayerAnime : MonoBehaviour
{
    public Transform m_Target;                                  // ゴールオブジェクト
    private CharacterController m_CharacterController;          // キャラクターコントローラー
    public Animator m_Animator;                                 // プレイヤーアニメーター
    private Player m_Player;                                    // Playerスクリプト
    [SerializeField]
    private GameObject m_PlayerModel;                           // プレイヤーモデル
    [SerializeField]
    private LimitTimer m_LimitTimer;                            // LimitTimerスクリプト
    [SerializeField]
    private ArClockNeedle m_ArClockNeedle;                      // ArClockNeedleスクリプト

    public ControllerCamera m_ControllerCamera;                 // ControllerCameraスクリプト
    public float m_ZoomScheduledTime;                           // ズームする時間
    public float m_GoalZoomDistance;                            // ゴールオブジェクトとの距離
    private float m_ZoomVelocity = 1.0f;                        // ズーム速度
    public bool m_GoalAnime;                                    // ゴールアニメーションフラグ

    public bool m_Wacth = false;                                // ウォッチアニメーションフラグ

    public GoalSmoothMove m_GoalSmoothMove;                     // goalSmoothMoveスクリプト
    public bool m_SmoothFlag = false;                           // 遷移が可能かどうかを知るためのフラグ

    [HideInInspector]
    public Color m_Color;                                       // カラー

    // Use this for initialization
    void Start () {
        m_CharacterController = GetComponent<CharacterController>();
        m_Animator = GetComponent<Animator>();
        m_Player = GetComponent<Player>();

        m_ControllerCamera = GameObject.Find("Main Camera").GetComponent<ControllerCamera>();
        m_GoalSmoothMove = GameObject.Find("Main Camera").GetComponent<GoalSmoothMove>();
    }
```

```
        // Update is called once per frame
    void Update()
    {
        // 死亡、ゴールのアニメーション再生中はキー操作を無効にする

        if(m_Player.HP == 0 || m_LimitTimer.m_timer <= 0)
        {
            m_Animator.SetBool("Dead", true);     // 死亡アニメーション
            m_Player.walkSpeed = 0;                // 死亡アニメーション再生中は動けないようにする
            m_Player.rotateSpeed = 0;              // 死亡アニメーション再生中は動けないようにする
            m_LimitTimer.enabled = false;
            m_Player.enabled = false;
            // 太陽（Directional Light）の動作をさせないようにするためスクリプトを止める
            GameObject.FindGameObjectWithTag("Sun").GetComponent<New_SunShine_Move>().enabled = false;
            // 死亡が確認できたタイミングで時間を止める為にスクリプト自体を止める
            m_ArClockNeedle.enabled = false;

            // エネミーの動作を停止させる
            foreach (GameObject enemy in GameObject.FindGameObjectsWithTag("Enemy"))
            {
                // EnemyAIスクリプトを取得し停止させる
                EnemyAI enemyAI = enemy.GetComponent<EnemyAI>();
                enemyAI.enabled = false;
                // エネミーのNavMeshAgentコンポーネントを取得し停止させる
                UnityEngine.AI.NavMeshAgent navMeshAgent =
enemy.GetComponent<UnityEngine.AI.NavMeshAgent>();
                navMeshAgent.enabled = false;
            }

            // アニメーターステートの取得
            AnimatorStateInfo animatorState = m_Animator.GetCurrentAnimatorStateInfo(0);

            // アニメーターの状態が"Dead_Anim"だったら
            if (animatorState.IsName("Dead_Anim"))
            {
                // "Dead_Anim"のアニメーションが終了しているか？          ※ 0 ～ 0.9 なら再生中 1以上ならアニメーションが終了している
                if (animatorState.normalizedTime > 1)
                {
                    // 死亡アニメーション直後にプレイヤーを消す（プレイヤーを透明度にして見えなくする）
                    m_Color = m_PlayerModel.GetComponent<Renderer>().material.color;
                    m_Color.a = Mathf.Clamp(m_Color.a, 0, 1);
```

```
                m_Color.a -= 0.01f;
                m_PlayerModel.GetComponent<Renderer>().material.color = m_Color;
            }
        }
    }

    // 指定されたキーが入力中、太陽を回転させてる時のアニメーションを再生
    if (Input.GetKey(KeyCode.Z) || (Input.GetKey(KeyCode.X)) || (Input.GetKey(KeyCode.JoystickButton6)) ||
(Input.GetKey(KeyCode.JoystickButton7)))
    {
        m_Wacth = true;
    }
    else
    {
        m_Wacth = false;
    }

    m_Animator.SetBool("wacth", m_Wacth);        // 時間を動かすアニメーション

    //GoalCameraZoom();
    GoalTargetSmoothMove();

}

private void OnTriggerEnter(Collider other)
{
    if ((other.gameObject.tag == "Goal"))
    {
        m_GoalAnime = true;
        // ゴールアニメーションの飛び込み時にターゲットへ飛び込ませるため
        transform.LookAt(m_Target.transform.position);
        m_Player.walkSpeed = 0;                          // アニメーション再生中は動かないようにする
        m_Player.rotateSpeed = 0;                        // アニメーション再生中は動かないようにする
        m_ControllerCamera.cameraAngleSpeed = 0;         // カメラの動作（回転・ズーム）をしないようにす
る
        m_LimitTimer.enabled = false;                    // タイマーの動作を止める

        // エネミーの動きを止める
        // ※ ゴール演出中でもエネミーが加入してくるので止める必要があったため
        foreach (GameObject enemy in GameObject.FindGameObjectsWithTag("Enemy"))
        {
            EnemyAI enemyAI = enemy.GetComponent<EnemyAI>();
            enemyAI.enabled = false;
```

```
                UnityEngine.AI.NavMeshAgent navMeshAgent =
enemy.GetComponent<UnityEngine.AI.NavMeshAgent>();
                navMeshAgent.enabled = false;
            }
        }
    }

    /// <summary>
    /// カメラを一定の距離まで近づけてからゴールアニメーションをさせる
    /// </summary>
    private void GoalCameraZoom()
    {
        // ゴールアニメーションを再生条件
        if (m_GoalAnime == true)
        {
            // カメラを指定（goalZoomDistance）した距離まで近づける
            m_ControllerCamera.cameraDistance = Mathf.SmoothDamp(m_ControllerCamera.cameraDistance,
m_GoalZoomDistance, ref m_ZoomVelocity, m_ZoomScheduledTime, 0.5f, 0.5f);

            // カメラの距離が3.5以下か？
            if (m_ControllerCamera.cameraDistance <= 3.5f)
            {
                m_GoalAnime = false;
                // カメラが目標距離に達していたらゴールアニメーションを再生させる
                if (m_GoalAnime == false && m_ControllerCamera.cameraDistance <= 3.5f)
                {
                    m_Animator.SetBool("Goal", true);          // ゴールアニメーション

                    m_ControllerCamera.cameraZoomSpeed = 0;
                }
            }
        }
    }

    /// <summary>
    /// カメラを目標位置と角度（ゴール用に置いたターゲットの場所）まで遷移させてからアニメーションさせる
    /// </summary>
    private void GoalTargetSmoothMove()
    {
        // ゴールアニメーションを再生条件
        if (m_GoalAnime == true)
        {
            // ControllerCameraスクリプトを止めてGoalSmoothMoveスクリプトを動かす
```

```
                // ※ ControllerCameraとGoalSmoothMoveはお互いにターゲットが違う上に干渉してし合うため片方が動いている時は片方を止める必要がある
                m_ControllerCamera.enabled = false;
                m_GoalSmoothMove.enabled = true;

                if (m_GoalSmoothMove.enabled == true)
                {
                    m_SmoothFlag = true;
                }
            }

            // カメラが目標位置に達していたらゴールアニメーションを再生させる
            if (m_GoalAnime == false && m_GoalSmoothMove.m_FlagFlag == true)
            {
                m_Animator.SetBool("Goal", true);          // ゴールアニメーション

                m_ControllerCamera.cameraZoomSpeed = 0;
            }
        }
    }
```