Assets\Scripts\PlayerCharacterC_PS4.cs

```
/*////////////////////////script sem pulo, apenas movimento/////////////////////////////////////////////////////
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private Transform cam;
    private CharacterController Personagem;
   //private Animator anim;

    public float velocidade_Movimento;
    private Vector3 direcao;
    private Vector3 direcao_Movimento;

    //variaveis para suavização de rotação do personagem
    private float tempo_De_Giro_Suave;
    private float velocidade_De_Giro_Suave;

    void Start()
    {
        cam = Camera.main.transform;
        Personagem = GetComponent<CharacterController>();
        //anim = GetComponent<Animator>();
    }

    void Update()
    {
        pegar_Comandos();
        mover_Personagem();
    }

    void pegar_Comandos()
    {
        direcao = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
    }

    void mover_Personagem()
    {
        if(direcao.magnitude > 0.1f)
        {
            float targetAngle = Mathf.Atan2(direcao.x, direcao.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
            float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref velocidade_De_Giro_Suave, tempo_De_Giro_Suave);
            transform.rotation = Quaternion.Euler(0f, angle, 0f);

            direcao_Movimento = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;
        }
        Personagem.Move(direcao_Movimento.normalized * velocidade_Movimento * direcao.magnitude * Time.deltaTime);
    }
}*/
/*////////////////////////este segundo funciona com pulo/////////////////////////////////////////////////////
using UnityEngine;

public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private CharacterController _controller;

    [SerializeField]private float _playerSpeed = 5f;
    [SerializeField]private float _rotationSpeed = 10f;
    [SerializeField]private Camera _followCamera;

    private Vector3 _playerVelocity;
    private bool _groundedPlayer;

    [SerializeField]private float _jumpHeight = 1.0f;
    [SerializeField]private float _gravityValue = -9.81f;

    private void Start()
    {
        _controller = GetComponent<CharacterController>();
    }

    private void Update()
    {
        Movement();
    }

    void Movement()
    {
        _groundedPlayer = _controller.isGrounded;
        if (_groundedPlayer && _playerVelocity.y < 0)
        {
            _playerVelocity.y = 0f;
        }

        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");
```

```

        Vector3 movementInput = Quaternion.Euler(0, _followCamera.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
        Vector3 movementDirection = movementInput.normalized;

        _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);

        if (movementDirection != Vector3.zero)
        {
            Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);

            transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _rotationSpeed * Time.deltaTime);
        }
        if (Input.GetButtonDown("Jump") && _groundedPlayer)
        {
            _playerVelocity.y += Mathf.Sqrt(_jumpHeight * -3.0f * _gravityValue);
        }

        _playerVelocity.y += _gravityValue * Time.deltaTime;
        _controller.Move(_playerVelocity * Time.deltaTime);
    }
}*/
/*////////////////////união dos dois códigos acima funcionando movimento e pulo com atraso/////////////////////
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private CharacterController _controle;
    private Camera _seguirCamera;

    [SerializeField]
    private float _velocidadePersonagem = 5f;
    [SerializeField]
    private float _velocidadeRotacao = 10f;
    [SerializeField]
    private float _alturaPulo = 1.0f;
    [SerializeField]
    private float _valorGravidade = -30f;//9.81f

    private Vector3 _velocidadeY;
    private bool _jogadorNoChao;

    private void Start()
    {
        _controle = GetComponent<CharacterController>();
        _seguirCamera = Camera.main;
    }

    private void Update()
    {
        Mover();
    }

    void Mover()
    {
        _jogadorNoChao = _controle.isGrounded;
        if (_jogadorNoChao && _velocidadeY.y < 0)
        {
            _velocidadeY.y = 0f;
        }

        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        Vector3 movementInput = Quaternion.Euler(0, _seguirCamera.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
        Vector3 movementDirection = movementInput.normalized;

        _controle.Move(movementDirection * _velocidadePersonagem * Time.deltaTime);

        if (movementDirection != Vector3.zero)
        {
            Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _velocidadeRotacao * Time.deltaTime);
        }
       if (Input.GetButtonDown("Jump") && _jogadorNoChao)
        {
            _velocidadeY.y += Mathf.Sqrt(_alturaPulo * -2.0f * _valorGravidade);
        }

        _velocidadeY.y += _valorGravidade * Time.deltaTime;
        _controle.Move(_velocidadeY * Time.deltaTime);
    }
}*//*
////////////////////////////////Script de pulo duplo perfeito 17/06/2023 18:16/////////////////////////////////////////
//Script de pulo duplo perfeito 18/06/2023 18:22
//Michael Moraes Sabino
//trillobit3s@gmail.com

using UnityEngine;

```

```
[RequireComponent(typeof(CharacterController))]
public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private CharacterController _controller;
    private Camera _followCam;

    [SerializeField]private float _playerSpeed = 5f;
    [SerializeField]private float _speedRotation = 10f;

    [SerializeField]private float _gravity = 9.8f;
    [SerializeField]private float _jumpSpeed = 3f;
    [SerializeField]private float _doubleJump = 0.5f;

    private float _directionY;
    private bool _canDoubleJump = false;

    private void Start()
    {
        _controller = GetComponent<CharacterController>();
        _followCam = Camera.main;

     //  Cursor.lockState = CursorLockMode.Locked;
     //  Cursor.visible = false;
    }

    private void Update()
    {
        Mover();
    }

    void Mover()
    {
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        Vector3 movementInput = Quaternion.Euler(0, _followCam.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
        Vector3 movementDirection = movementInput.normalized;

        if (movementDirection != Vector3.zero)
        {
            Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _speedRotation * Time.deltaTime);
        }

        if(_controller.isGrounded){
            _canDoubleJump = true;
            if(Input.GetButtonDown("Jump")){
                _directionY = _jumpSpeed;
                anim.SetBool("Jump", true);
            }
        }else{
            if(Input.GetButtonDown("Jump") && _canDoubleJump){
                _directionY = _jumpSpeed * _doubleJump;
                _canDoubleJump = false;
            }
        }

        _directionY -= _gravity * Time.deltaTime;
        movementDirection.y = _directionY;
        _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);//remove "_playerSpeed"
    }
}*/
//////////////////////////script sem pulo, apenas movimento////////////////////////////////////////////////////
/*
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private Transform cam;
    private CharacterController Personagem;
    private Animator anim;

    public float velocidade_Movimento;
    public float gravidade = 9.81f; // Valor da gravidade
    private Vector3 direcao;
    private Vector3 direcao_Movimento;
    private float velocidadeVertical; // Velocidade vertical do personagem

    //variaveis para suavização de rotação do personagem
    private float tempo_De_Giro_Suave;
    private float velocidade_De_Giro_Suave;

    void Start()
    {
        cam = Camera.main.transform;
        Personagem = GetComponent<CharacterController>();
        anim = GetComponent<Animator>();
    }

    void Update()
    {
```

```
        pegar_Comandos();
        mover_Personagem();
        animar_Personagem();
    }

    void pegar_Comandos()
    {
        direcao = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
    }

    void mover_Personagem()
    {
        if (direcao.magnitude > 0.1f)
        {
            float targetAngle = Mathf.Atan2(direcao.x, direcao.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
            float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref velocidade_De_Giro_Suave, tempo_De_Giro_Suave);
            transform.rotation = Quaternion.Euler(0f, angle, 0f);

            direcao_Movimento = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;
        }

        // Aplicar a gravidade usando o método de integração de Euler
        if (Personagem.isGrounded) // Se o personagem estiver no chão
        {
            velocidadeVertical = 0f; // A velocidade vertical é zero
        }
        else // Se o personagem estiver no ar
        {
            velocidadeVertical -= gravidade * Time.deltaTime; // Aplica a gravidade à velocidade vertical
        }

        direcao_Movimento.y = velocidadeVertical; // Atualiza a componente vertical do movimento

        Personagem.Move(direcao_Movimento.normalized * velocidade_Movimento * direcao.magnitude * Time.deltaTime);
    }

    void animar_Personagem()
    {
        float velocidade = direcao.magnitude * velocidade_Movimento;
        anim.SetFloat("MoveSpeed", velocidade);

        if (velocidade > 0.1f)
        {
            anim.SetBool("Idle", false);
            anim.SetBool("Walk", true);
          //  anim.SetBool("Run", true);
        }
        else
        {
            anim.SetBool("Walk", false);
            anim.SetBool("Run", false);
        //    anim.SetBool("Idle", true);
        }
    }
}*/
//apenas traduzido
/*
using UnityEngine;

public class Player : MonoBehaviour
{
    private CharacterController personagem;

    public float velocidade = 5f;
    public float velocidadeRotacao = 10f;
    public Camera seguirCamera;

    private Vector3 velocidadeJogador;
    private bool jogadorNoChao;

    public float alturaPulo = 1.0f;
    public float gravidade = -9.81f;

    void Start()
    {
        personagem = GetComponent<CharacterController>();
    }

    void Update()
    {
        Mover();
    }

    void Mover()
    {
        jogadorNoChao = personagem.isGrounded;
        if (jogadorNoChao && velocidadeJogador.y < 0)
        {
            velocidadeJogador.y = 0f;
        }

        float horizontalInput = Input.GetAxis("Horizontal");
```

```
        float verticalInput = Input.GetAxis("Vertical");

        Vector3 movementInput = Quaternion.Euler(0, seguirCamera.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
        Vector3 movementDirection = movementInput.normalized;

        personagem.Move(movementDirection * velocidade * Time.deltaTime);

        if (movementDirection != Vector3.zero)
        {
            Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, velocidadeRotacao * Time.deltaTime);
        }
        if (Input.GetButtonDown("Jump") && jogadorNoChao)
        {
            velocidadeJogador.y += Mathf.Sqrt(alturaPulo * -3.0f * gravidade);
        }

        velocidadeJogador.y += gravidade * Time.deltaTime;
        personagem.Move(velocidadeJogador * Time.deltaTime);
    }
}
*/

//////////////////////////////////////Script de animação 26/07/2023  PM11:27/////////////////////////////////////////////
//Ainda possui erros de pulo, o personagem não salta mais que o basico da animação feita no blender
//porem se uso o _jumpSpeed ela salta duas vezes
//uma do _directionY e outra da animação
//até o momento o personagem "fica parado, anda, pula, anda e pula"
/*
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private CharacterController _controller;
    private Camera _followCam;
    private Animator anim;

    [SerializeField] private float _playerSpeed = 5f;
    [SerializeField] private float _speedRotation = 10f;

    [SerializeField] private float _gravity = 9.8f;//use a gravidade de 2
  //[SerializeField] private float _jumpSpeed = 3f;
  //[SerializeField] private float _doubleJump = 0.5f;

    private float _directionY;
    // private bool _canDoubleJump = false;

    private void Start()
    {
        _controller = GetComponent<CharacterController>();
        _followCam = Camera.main;
        anim = GetComponent<Animator>();
    }

    private void Update()
    {
        Mover();
    }

    void Mover()
    {
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        Vector3 movementInput = Quaternion.Euler(0, _followCam.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
        Vector3 movementDirection = movementInput.normalized;

        if (movementDirection != Vector3.zero)
        {
            Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _speedRotation * Time.deltaTime);

            anim.SetFloat("MoveSpeed", movementDirection.magnitude * _playerSpeed);
        }
        else
        {
            anim.SetFloat("MoveSpeed", 0f);
        }

        if (_controller.isGrounded)
        {
            //_canDoubleJump = true;

            if (Input.GetButtonDown("Jump"))
            {
                anim.SetTrigger("JumpTrigger");
               // _directionY = _jumpSpeed;
            }
        }*/
        /*else
        {
```

```
            if (Input.GetButtonDown("Jump") && _canDoubleJump)
            {
              //_directionY = _jumpSpeed * _doubleJump;
                _canDoubleJump = false;
            }
        }*//*

        _directionY -= _gravity * Time.deltaTime;//
        movementDirection.y = _directionY;
      //_controller.Move(movementDirection * Time.deltaTime);//esta linha deixa o player lento, para resolver usa-se a de baixo
        _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);//esta linha normaliza a velocidade do player
    }
}

*/
//teste de altura do pulo 27/07/2023 17:15
//esta ficando bom
using UnityEngine;

[RequireComponent(typeof(CharacterController))]
public class PlayerCharacterC_PS4 : MonoBehaviour
{
    private CharacterController _controller;
    private Camera _followCam;
    private Animator anim;

    [SerializeField] private float _playerSpeed = 5f;
    [SerializeField] private float _speedRotation = 10f;
    [SerializeField] private float _gravity = 9.8f;
    [SerializeField] private float _jumpForce = 0.9f;

    private float _directionY;
    public bool _isJumping;

    private void Start()
    {
        _controller = GetComponent<CharacterController>();
        _followCam = Camera.main;
        anim = GetComponent<Animator>();
    }

    private void Update()
    {
        Mover();
    }

    void Mover()
    {
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");

        Vector3 movementInput = Quaternion.Euler(0, _followCam.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
        Vector3 movementDirection = movementInput.normalized;

        if (movementDirection != Vector3.zero)
        {
            Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _speedRotation * Time.deltaTime);

            anim.SetFloat("MoveSpeed", movementDirection.magnitude * _playerSpeed);
        }
        else
        {
            anim.SetFloat("MoveSpeed", 0f);
        }

        if (_controller.isGrounded)
        {
            if (Input.GetButtonDown("Jump"))
            {
                // Aplica a força do pulo ao jogador
                _directionY = _jumpForce;
                _isJumping = true;
                anim.SetTrigger("JumpTrigger");
            }
            else
            {
                // Garante que o jogador permaneça colado ao chão quando não estiver pulando
                _directionY = -0.5f;
                _isJumping = false;
            }
        }
        else
        {
            // Aplica a gravidade enquanto estiver no ar
            _directionY -= _gravity * Time.deltaTime;
        }

        movementDirection.y = _directionY;
        _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);
    }
}
```