

Assets\Scripts\PlayerCharacterC_PS4.cs

```
1  /*////////////////////////////////////////script sem pulo, apenas movimento////////////////////////////////////////
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  [RequireComponent(typeof(CharacterController))]
7  public class PlayerCharacterC_PS4 : MonoBehaviour
8  {
9      private Transform cam;
10     private CharacterController Personagem;
11     //private Animator anim;
12
13     public float velocidade_Movimento;
14     private Vector3 direcao;
15     private Vector3 direcao_Movimento;
16
17     //variaveis para suavização de rotação do personagem
18     private float tempo_De_Giro_Suave;
19     private float velocidade_De_Giro_Suave;
20
21     void Start()
22     {
23         cam = Camera.main.transform;
24         Personagem = GetComponent<CharacterController>();
25         //anim = GetComponent<Animator>();
26     }
27
28     void Update()
29     {
30         pegar_Comandos();
31         mover_Personagem();
32     }
33
34     void pegar_Comandos()
35     {
36         direcao = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
37     }
38
39     void mover_Personagem()
40     {
41         if(direcao.magnitude > 0.1f)
42         {
43             float targetAngle = Mathf.Atan2(direcao.x, direcao.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
44             float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref velocidade_De_Giro_Suave, tempo_De_Giro_Suave);
45             transform.rotation = Quaternion.Euler(0f, angle, 0f);
46
47             direcao_Movimento = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;
48         }
49         Personagem.Move(direcao_Movimento.normalized * velocidade_Movimento * direcao.magnitude * Time.deltaTime);
50     }
51 }*/
52 /*////////////////////////////////////////este segundo funciona com pulo////////////////////////////////////////
53 using UnityEngine;
54
55 public class PlayerCharacterC_PS4 : MonoBehaviour
56 {
57     private CharacterController _controller;
58
59     [SerializeField]private float _playerSpeed = 5f;
60     [SerializeField]private float _rotationSpeed = 10f;
61     [SerializeField]private Camera _followCamera;
62
63     private Vector3 _playerVelocity;
64     private bool _groundedPlayer;
65
66     [SerializeField]private float _jumpHeight = 1.0f;
67     [SerializeField]private float _gravityValue = -9.81f;
68
69     private void Start()
70     {
71         _controller = GetComponent<CharacterController>();
72     }
73
74     private void Update()
75     {
76         Movement();
77     }
78
79     void Movement()
80     {
81         _groundedPlayer = _controller.isGrounded;
82         if (_groundedPlayer && _playerVelocity.y < 0)
83         {
84             _playerVelocity.y = 0f;
85         }
86
87         float horizontalInput = Input.GetAxis("Horizontal");
88         float verticalInput = Input.GetAxis("Vertical");
```

```

89
90     Vector3 movementInput = Quaternion.Euler(0, _followCamera.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
91     Vector3 movementDirection = movementInput.normalized;
92
93     _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);
94
95     if (movementDirection != Vector3.zero)
96     {
97         Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
98
99         transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _rotationSpeed * Time.deltaTime);
100     }
101     if (Input.GetButtonDown("Jump") && _groundedPlayer)
102     {
103         _playerVelocity.y += Mathf.Sqrt(_jumpHeight * -3.0f * _gravityValue);
104     }
105
106     _playerVelocity.y += _gravityValue * Time.deltaTime;
107     _controller.Move(_playerVelocity * Time.deltaTime);
108 }
109 */
110 /*////////////////////uniao dos dois códigos acima funcionando movimento e pulo com atraso////////////////////
111 using System.Collections;
112 using System.Collections.Generic;
113 using UnityEngine;
114
115 [RequireComponent(typeof(CharacterController))]
116 public class PlayerCharacterC_PS4 : MonoBehaviour
117 {
118     private CharacterController _controle;
119     private Camera _seguirCamera;
120
121     [SerializeField]
122     private float _velocidadePersonagem = 5f;
123     [SerializeField]
124     private float _velocidadeRotacao = 10f;
125     [SerializeField]
126     private float _alturaPulo = 1.0f;
127     [SerializeField]
128     private float _valorGravidade = -30f;//9.81f
129
130     private Vector3 _velocidadeY;
131     private bool _jogadorNoChao;
132
133     private void Start()
134     {
135         _controle = GetComponent<CharacterController>();
136         _seguirCamera = Camera.main;
137     }
138
139     private void Update()
140     {
141         Mover();
142     }
143
144     void Mover()
145     {
146         _jogadorNoChao = _controle.isGrounded;
147         if (_jogadorNoChao && _velocidadeY.y < 0)
148         {
149             _velocidadeY.y = 0f;
150         }
151
152         float horizontalInput = Input.GetAxis("Horizontal");
153         float verticalInput = Input.GetAxis("Vertical");
154
155         Vector3 movementInput = Quaternion.Euler(0, _seguirCamera.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
156         Vector3 movementDirection = movementInput.normalized;
157
158         _controle.Move(movementDirection * _velocidadePersonagem * Time.deltaTime);
159
160         if (movementDirection != Vector3.zero)
161         {
162             Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
163             transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _velocidadeRotacao * Time.deltaTime);
164         }
165         if (Input.GetButtonDown("Jump") && _jogadorNoChao)
166         {
167             _velocidadeY.y += Mathf.Sqrt(_alturaPulo * -2.0f * _valorGravidade);
168         }
169
170         _velocidadeY.y += _valorGravidade * Time.deltaTime;
171         _controle.Move(_velocidadeY * Time.deltaTime);
172     }
173 }*/
174 //////////////////////////////////Script de pulo duplo perfeito 17/06/2023 18:16////////////////////////////////////
175 //Script de pulo duplo perfeito 18/06/2023 18:22
176 //Michael Moraes Sabino
177 //trillobit3s@gmail.com
178
179 using UnityEngine;
180

```

```

181 [RequireComponent(typeof(CharacterController))]
182 public class PlayerCharacterC_PS4 : MonoBehaviour
183 {
184     private CharacterController _controller;
185     private Camera _followCam;
186
187     [SerializeField]private float _playerSpeed = 5f;
188     [SerializeField]private float _speedRotation = 10f;
189
190     [SerializeField]private float _gravity = 9.8f;
191     [SerializeField]private float _jumpSpeed = 3f;
192     [SerializeField]private float _doubleJump = 0.5f;
193
194     private float _directionY;
195     private bool _canDoubleJump = false;
196
197     private void Start()
198     {
199         _controller = GetComponent<CharacterController>();
200         _followCam = Camera.main;
201
202         // Cursor.lockState = CursorLockMode.Locked;
203         // Cursor.visible = false;
204     }
205
206     private void Update()
207     {
208         Mover();
209     }
210
211     void Mover()
212     {
213         float horizontalInput = Input.GetAxis("Horizontal");
214         float verticalInput = Input.GetAxis("Vertical");
215
216         Vector3 movementInput = Quaternion.Euler(0, _followCam.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
217         Vector3 movementDirection = movementInput.normalized;
218
219         if (movementDirection != Vector3.zero)
220         {
221             Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
222             transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _speedRotation * Time.deltaTime);
223         }
224
225         if(_controller.isGrounded){
226             _canDoubleJump = true;
227             if(Input.GetButtonDown("Jump")){
228                 _directionY = _jumpSpeed;
229                 anim.SetBool("Jump", true);
230             }
231         }else{
232             if(Input.GetButtonDown("Jump") && _canDoubleJump){
233                 _directionY = _jumpSpeed * _doubleJump;
234                 _canDoubleJump = false;
235             }
236         }
237
238         _directionY -= _gravity * Time.deltaTime;
239         movementDirection.y = _directionY;
240         _controller.Move(movementDirection * _playerSpeed * Time.deltaTime); //remove "_playerSpeed"
241     }
242 }*/
243 //////////////////////////////////script sem pulo, apenas movimento////////////////////////////////////
244 /*
245 using UnityEngine;
246
247 [RequireComponent(typeof(CharacterController))]
248 public class PlayerCharacterC_PS4 : MonoBehaviour
249 {
250     private Transform cam;
251     private CharacterController Personagem;
252     private Animator anim;
253
254     public float velocidade_Movimento;
255     public float gravidade = 9.81f; // Valor da gravidade
256     private Vector3 direcao;
257     private Vector3 direcao_Movimento;
258     private float velocidadeVertical; // Velocidade vertical do personagem
259
260     //variaveis para suavização de rotação do personagem
261     private float tempo_De_Giro_Suave;
262     private float velocidade_De_Giro_Suave;
263
264     void Start()
265     {
266         cam = Camera.main.transform;
267         Personagem = GetComponent<CharacterController>();
268         anim = GetComponent<Animator>();
269     }
270
271     void Update()
272     {

```

```

273     pegar_Comandos();
274     mover_Personagem();
275     animar_Personagem();
276 }
277
278 void pegar_Comandos()
279 {
280     direcao = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
281 }
282
283 void mover_Personagem()
284 {
285     if (direcao.magnitude > 0.1f)
286     {
287         float targetAngle = Mathf.Atan2(direcao.x, direcao.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
288         float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref velocidade_De_Giro_Suave, tempo_De_Giro_Suave);
289         transform.rotation = Quaternion.Euler(0f, angle, 0f);
290
291         direcao_Movimento = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;
292     }
293
294     // Aplicar a gravidade usando o método de integração de Euler
295     if (Personagem.isGrounded) // Se o personagem estiver no chão
296     {
297         velocidadeVertical = 0f; // A velocidade vertical é zero
298     }
299     else // Se o personagem estiver no ar
300     {
301         velocidadeVertical -= gravidade * Time.deltaTime; // Aplica a gravidade à velocidade vertical
302     }
303
304     direcao_Movimento.y = velocidadeVertical; // Atualiza a componente vertical do movimento
305
306     Personagem.Move(direcao_Movimento.normalized * velocidade_Movimento * direcao.magnitude * Time.deltaTime);
307 }
308
309 void animar_Personagem()
310 {
311     float velocidade = direcao.magnitude * velocidade_Movimento;
312     anim.SetFloat("MoveSpeed", velocidade);
313
314     if (velocidade > 0.1f)
315     {
316         anim.SetBool("Idle", false);
317         anim.SetBool("Walk", true);
318         // anim.SetBool("Run", true);
319     }
320     else
321     {
322         anim.SetBool("Walk", false);
323         anim.SetBool("Run", false);
324         // anim.SetBool("Idle", true);
325     }
326 }
327 */
328 //apenas traduzido
329 /*
330 using UnityEngine;
331
332 public class Player : MonoBehaviour
333 {
334     private CharacterController personagem;
335
336     public float velocidade = 5f;
337     public float velocidadeRotacao = 10f;
338     public Camera seguirCamera;
339
340     private Vector3 velocidadeJogador;
341     private bool jogadorNoChao;
342
343     public float alturaPulo = 1.0f;
344     public float gravidade = -9.81f;
345
346     void Start()
347     {
348         personagem = GetComponent<CharacterController>();
349     }
350
351     void Update()
352     {
353         Mover();
354     }
355
356     void Mover()
357     {
358         jogadorNoChao = personagem.isGrounded;
359         if (jogadorNoChao && velocidadeJogador.y < 0)
360         {
361             velocidadeJogador.y = 0f;
362         }
363
364         float horizontalInput = Input.GetAxis("Horizontal");

```



```

365         float verticalInput = Input.GetAxis("Vertical");
366
367         Vector3 movementInput = Quaternion.Euler(0, seguirCamera.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
368         Vector3 movementDirection = movementInput.normalized;
369
370         personagem.Move(movementDirection * velocidade * Time.deltaTime);
371
372         if (movementDirection != Vector3.zero)
373         {
374             Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
375             transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, velocidadeRotacao * Time.deltaTime);
376         }
377         if (Input.GetButtonDown("Jump") && jogadorNoChao)
378         {
379             velocidadeJogador.y += Mathf.Sqrt(alturaPulo * -3.0f * gravidade);
380         }
381
382         velocidadeJogador.y += gravidade * Time.deltaTime;
383         personagem.Move(velocidadeJogador * Time.deltaTime);
384     }
385 }
386 */
387
388 //////////////////////////////////////////////////////////////////Script de animação 26/07/2023 PM11:27////////////////////////////////////////////////////////////////
389 //Ainda possui erros de pulo, o personagem não salta mais que o basico da animação feita no blender
390 //porem se uso o _jumpSpeed ela salta duas vezes
391 //uma do _directionY e outra da animação
392 //até o momento o personagem "fica parado, anda, pula, anda e pula"
393 /*
394 using UnityEngine;
395
396 [RequireComponent(typeof(CharacterController))]
397 public class PlayerCharacterC_PS4 : MonoBehaviour
398 {
399     private CharacterController _controller;
400     private Camera _followCam;
401     private Animator anim;
402
403     [SerializeField] private float _playerSpeed = 5f;
404     [SerializeField] private float _speedRotation = 10f;
405
406     [SerializeField] private float _gravity = 9.8f;//use a gravidade de 2
407     //[SerializeField] private float _jumpSpeed = 3f;
408     //[SerializeField] private float _doubleJump = 0.5f;
409
410     private float _directionY;
411     // private bool _canDoubleJump = false;
412
413     private void Start()
414     {
415         _controller = GetComponent<CharacterController>();
416         _followCam = Camera.main;
417         anim = GetComponent<Animator>();
418     }
419
420     private void Update()
421     {
422         Mover();
423     }
424
425     void Mover()
426     {
427         float horizontalInput = Input.GetAxis("Horizontal");
428         float verticalInput = Input.GetAxis("Vertical");
429
430         Vector3 movementInput = Quaternion.Euler(0, _followCam.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
431         Vector3 movementDirection = movementInput.normalized;
432
433         if (movementDirection != Vector3.zero)
434         {
435             Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
436             transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _speedRotation * Time.deltaTime);
437
438             anim.SetFloat("MoveSpeed", movementDirection.magnitude * _playerSpeed);
439         }
440         else
441         {
442             anim.SetFloat("MoveSpeed", 0f);
443         }
444
445         if (_controller.isGrounded)
446         {
447             //_canDoubleJump = true;
448
449             if (Input.GetButtonDown("Jump"))
450             {
451                 anim.SetTrigger("JumpTrigger");
452                 //_directionY = _jumpSpeed;
453             }
454         }*/
455     /*else
456     {

```

```
457         if (Input.GetButtonDown("Jump") && _canDoubleJump)
458         {
459             //_directionY = _jumpSpeed * _doubleJump;
460             _canDoubleJump = false;
461         }
462     }*//*
463
464     _directionY -= _gravity * Time.deltaTime;//
465     movementDirection.y = _directionY;
466     //_controller.Move(movementDirection * Time.deltaTime);//esta linha deixa o player lento, para resolver usa-se a de baixo
467     _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);//esta linha normaliza a velocidade do player
468 }
469 }
470
471 */
472 //teste de altura do pulo 27/07/2023 17:15
473 //esta ficando bom
474 using UnityEngine;
475
476 [RequireComponent(typeof(CharacterController))]
477 public class PlayerCharacterC_PS4 : MonoBehaviour
478 {
479     private CharacterController _controller;
480     private Camera _followCam;
481     private Animator anim;
482
483     [SerializeField] private float _playerSpeed = 5f;
484     [SerializeField] private float _speedRotation = 10f;
485     [SerializeField] private float _gravity = 9.8f;
486     [SerializeField] private float _jumpForce = 0.9f;
487
488     private float _directionY;
489     public bool _isJumping;
490
491     private void Start()
492     {
493         _controller = GetComponent<CharacterController>();
494         _followCam = Camera.main;
495         anim = GetComponent<Animator>();
496     }
497
498     private void Update()
499     {
500         Mover();
501     }
502
503     void Mover()
504     {
505         float horizontalInput = Input.GetAxis("Horizontal");
506         float verticalInput = Input.GetAxis("Vertical");
507
508         Vector3 movementInput = Quaternion.Euler(0, _followCam.transform.eulerAngles.y, 0) * new Vector3(horizontalInput, 0, verticalInput);
509         Vector3 movementDirection = movementInput.normalized;
510
511         if (movementDirection != Vector3.zero)
512         {
513             Quaternion desiredRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
514             transform.rotation = Quaternion.Slerp(transform.rotation, desiredRotation, _speedRotation * Time.deltaTime);
515
516             anim.SetFloat("MoveSpeed", movementDirection.magnitude * _playerSpeed);
517         }
518         else
519         {
520             anim.SetFloat("MoveSpeed", 0f);
521         }
522
523         if (_controller.isGrounded)
524         {
525             if (Input.GetButtonDown("Jump"))
526             {
527                 // Aplica a força do pulo ao jogador
528                 _directionY = _jumpForce;
529                 _isJumping = true;
530                 anim.SetTrigger("JumpTrigger");
531             }
532             else
533             {
534                 // Garante que o jogador permaneça colado ao chão quando não estiver pulando
535                 _directionY = -0.5f;
536                 _isJumping = false;
537             }
538         }
539         else
540         {
541             // Aplica a gravidade enquanto estiver no ar
542             _directionY -= _gravity * Time.deltaTime;
543         }
544
545         movementDirection.y = _directionY;
546         _controller.Move(movementDirection * _playerSpeed * Time.deltaTime);
547     }
548 }
```