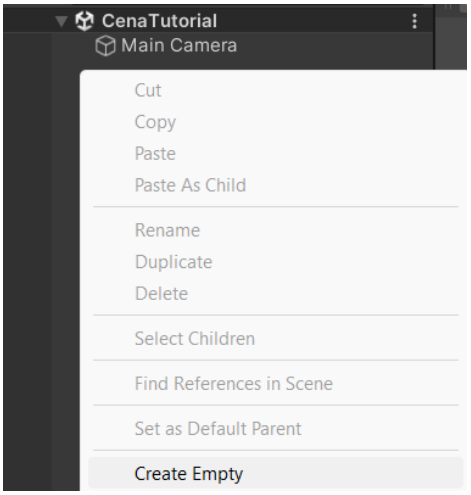
Recriando Celeste na Unity

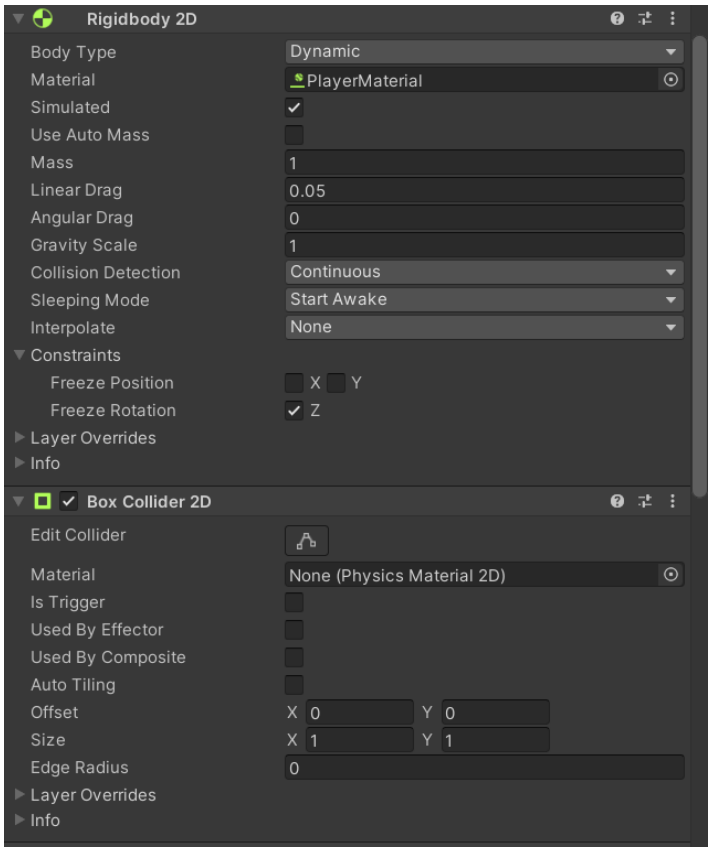
1 - Começando a preparação na Unity

Comece criando o objeto do jogador, clicando com o botão direito na Cena e selecionando Create Empty

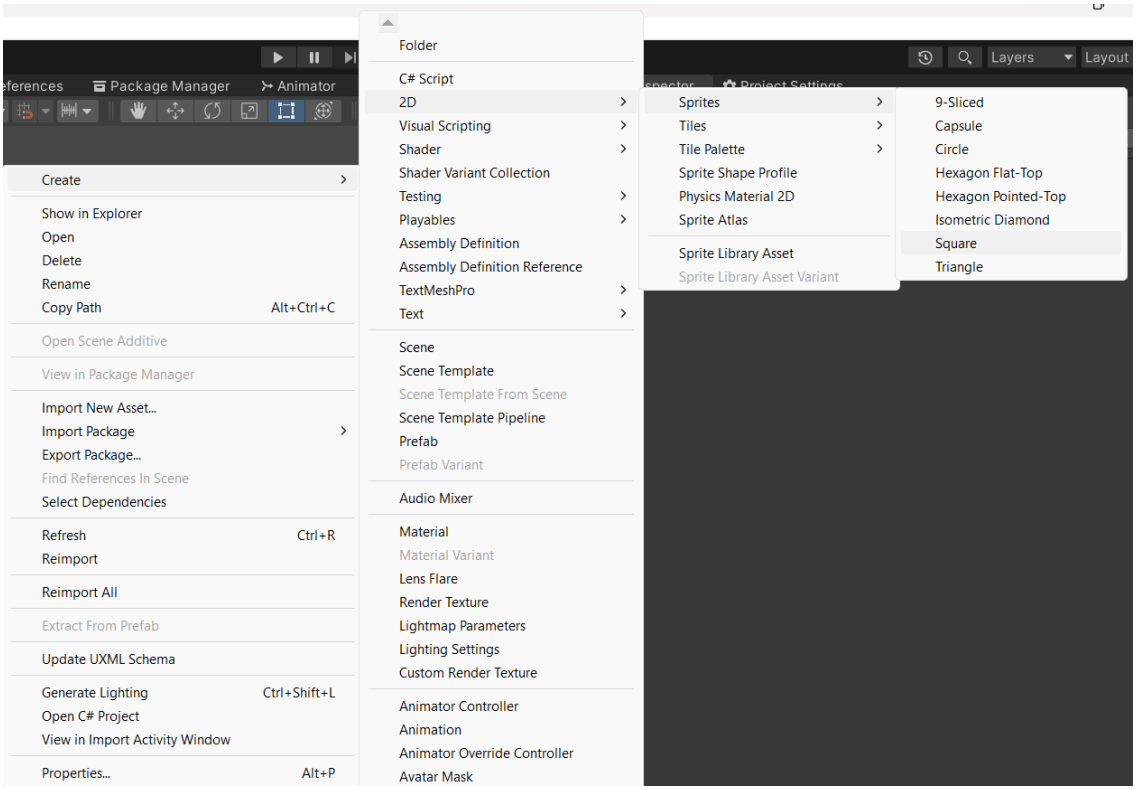


Agora, adicione o Rigidbody2D e o BoxCollider2D, marcando Freeze Rotation.

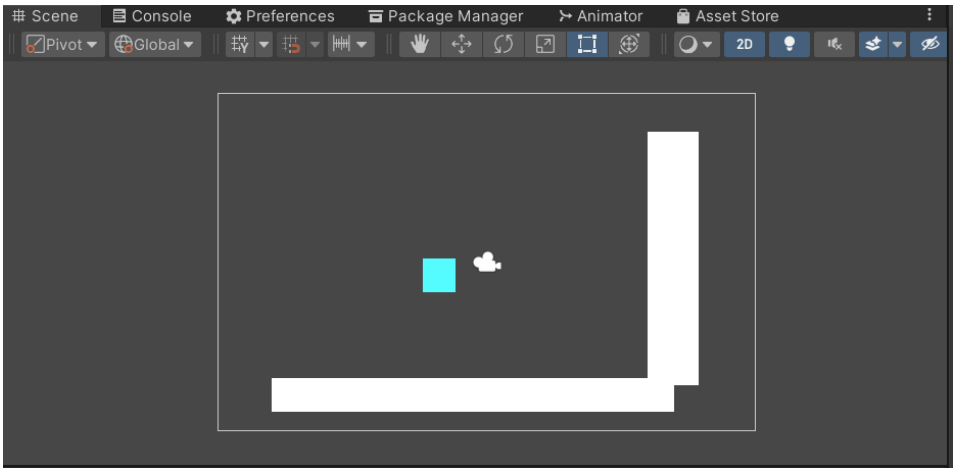
(ATENÇÃO: Existem a versão 3D deles, então lembre de adicionar aqueles que contêm o 2D no final)

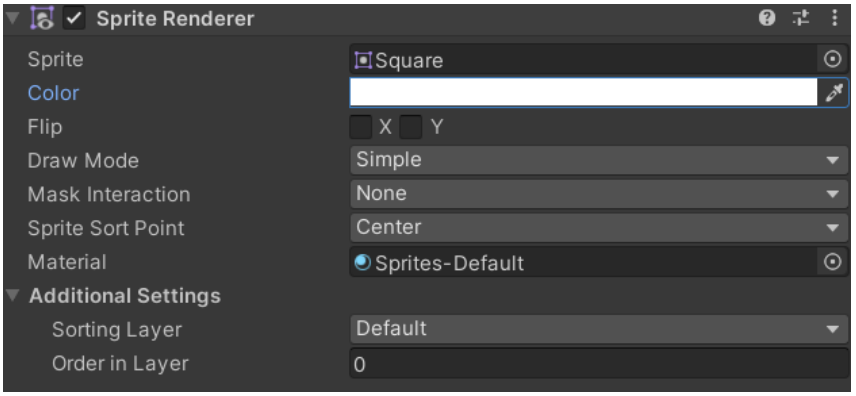


Crie o Sprite de um quadrado para começarmos o protótipo:

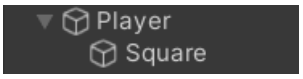


Coloque 3 quadrados na Scene. Apertando a tecla T, você pode alterar o tamanho dos quadrados selecionados. Altere também a cor de um quadrado, que será o protótipo do personagem.



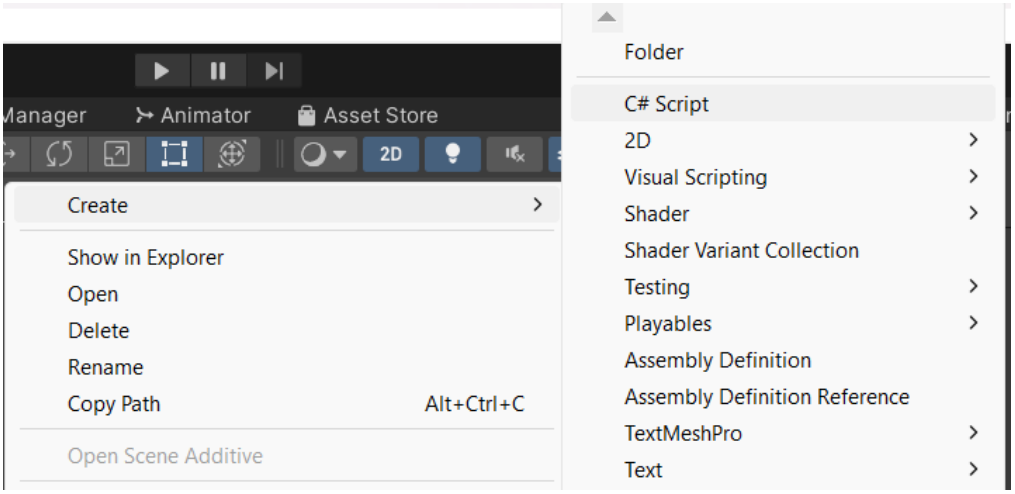


Coloque o quadrado colorido dentro do objeto vazio criado anteriormente

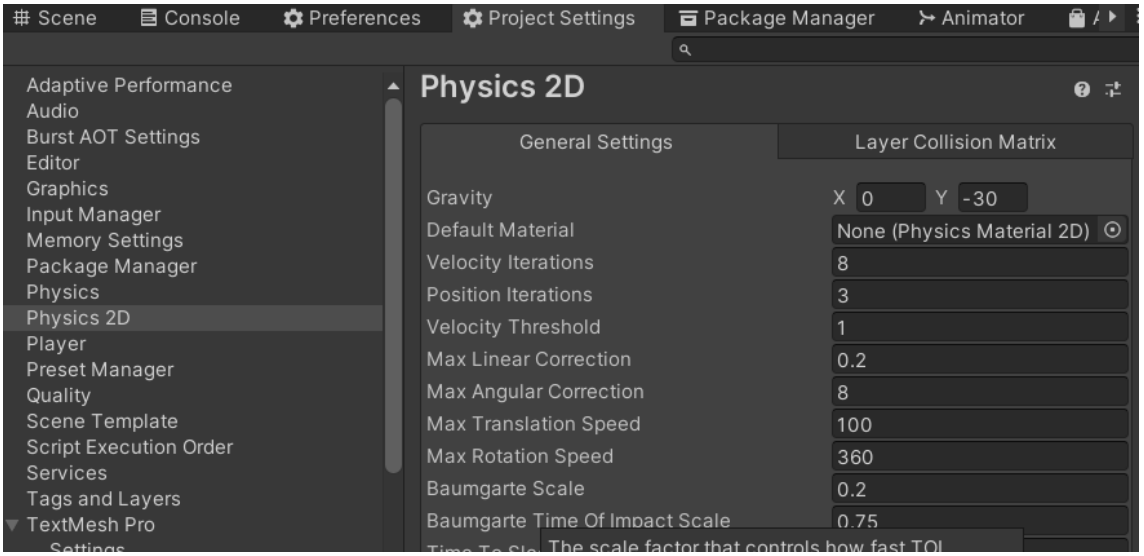


Crie um script e altere o nome para PlayerScript. Não dê enter antes de alterar o nome, ou seja, não tente renomear o script. Abra ele.

Caso você tente renomear o Script, troque dentro dele o public class ****** para o novo nome.



Mude a gravidade para -30 aqui:



2 - Mecânicas básicas de movimentação

2.1 – Movimentação Horizontal

Vamos criar uma variável pública (apenas para alterarmos dentro da Unity) float para a velocidade, uma float para a direção, uma Rigidbody2D para receber o componente colocado no jogador.

A direção irá receber um valor do input horizontal, para isso vamos utilizar o sistema da Unity. No método FixedUpdate, mudamos a velocidade do jogador de acordo com a direção.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    float direcao;

    Rigidbody2D rb;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        direcao = Input.GetAxisRaw("Horizontal");
    }

    void FixedUpdate(){
        rb.velocity = new Vector2(direcao * velocidade, rb.velocity.y);
    }
```

```
}
```

Agora, podemos testar e nosso quadrado já se movimenta de um lado para o outro.

2.2 – Pulo

Agora, vamos criar mais uma variável pública 'float' para a força do pulo, uma LayerMask para verificar o chão, uma BoxCollider2D para obter informações da colisão e uma bool para verificar se o personagem está no chão. Vamos criar também uma função do tipo 'void' para o pulo, uma função 'bool' para verificar se está no chão e verificar se a tecla é pressionada dentro do Update.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;

    bool noChao;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        col = GetComponent<BoxCollider2D>();
    }

    void Update()
    {
        noChao = IsGrounded();
        direcao = Input.GetAxisRaw("Horizontal");

        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }
    }

    void FixedUpdate(){
        rb.velocity = new Vector2(direcao * velocidade, rb.velocity.y);
```

```
    }

    void Pulo(){
        if(!noChao)
            return;
        rb.velocity = Vector2.zero;
        rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
    }
    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }
}
```

2.3 – Dash

Agora, criamos uma 'bool' para verificar se está dando dash, uma 'float' para a velocidade do dash e outra para o tempo dele, um Vector2 para pegar a direção do dash. Criamos também uma função para executar o dash e outra função para terminar o dash.

IEnumerator são funções que conseguem esperar um tempo específico.

Fazemos ela esperar com: yield return new WaitForSeconds(tempo);

Chamamos ela por: StartCoroutine("NomeDela")

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
```

```
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        col = GetComponent<BoxCollider2D>();
    }

    void Update()
    {
        noChao = IsGrounded();
        direcao = Input.GetAxisRaw("Horizontal");

        if(noChao){
            if(!podeDarDash){
                podeDarDash = true;
            }
        }

        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }
    }
```

```
    void FixedUpdate(){
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        }

    }

    void Pulo(){
        if(!noChao)
            return;
        rb.velocity = new Vector2(rb.velocity.x, 0);;
        rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
    }
    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }

    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
```

```
        podeDarDash = false;
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));

        yield return new WaitForSeconds(tempoDash);

        rb.gravityScale = 1;
        movimentacaoPadrao = true;
        rb.velocity = alvo;
        dashing = false;
    }
}
```

2.4 - Escalada

Escalada padrão

Aqui, criamos uma 'float' direcaoY e velocidadeParede, 'bool' tocandoParede e escalando. Testamos se a tecla Z está pressionada ou não e, dependendo do resultado, habilitamos ou desabilitamos a escalada.

Na função que começa a escalada, devemos usar StopAllCoroutines() para pararmos a IEnumerator do dash, para evitarmos bugs.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
```

```
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    float direcaoY;
    bool tocandoParede = false;
    bool escalando = false;
    public float velocidadeParede;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        col = GetComponent<BoxCollider2D>();
    }

    void Update()
    {
        noChao = IsGrounded();
        direcao = Input.GetAxisRaw("Horizontal");
        direcaoY = Input.GetAxisRaw("Vertical");
        tocandoParede = ChecaTocandoParede();

        if(noChao){
            if(!podeDarDash){
                podeDarDash = true;
            }
        }
```

```
        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }

        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
        }
    }

    void FixedUpdate(){
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }

    }

    void Pulo(){
        if(!noChao)
            return;
        rb.velocity = new Vector2(rb.velocity.x, 0);;
        rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
    }
```

```
    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }

    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));

        yield return new WaitForSeconds(tempoDash);

        rb.gravityScale = 1;
        movimentacaoPadrao = true;
        rb.velocity = alvo;
        dashing = false;
    }

    bool ChecaTocandoParede(){
```

```
        float extraHeight = 0.15f;
        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);
        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);
        return retorno || retorno2;
    }
    void EscalarParede(){
        escalando = true;
        movimentacaoPadrao = false;
        dashing = false;
        rb.gravityScale = 0;
        StopAllCoroutines();
    }
    void PararEscalarParede(){
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

Sistema de stamina

Aqui criamos duas 'floats': o tempo de stamina máximo e o tempo atual. Sempre que a personagem estiver escalando, queremos decrescer do valor do tempo atual. Ao chegar ao chão, o tempo é reiniciado e quando chega a 0, a escalada é terminada.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
```

```
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    float direcaoY;
    bool tocandoParede = false;
    bool escalando = false;
    public float velocidadeParede;

    public float tempoEscalada;
    float tempoEscaladaPassado = 0;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        col = GetComponent<BoxCollider2D>();
    }

    void Update()
    {
        noChao = IsGrounded();
        direcao = Input.GetAxisRaw("Horizontal");
        direcaoY = Input.GetAxisRaw("Vertical");
        tocandoParede = ChecaTocandoParede();

        if(noChao){
```

```
            if(!podeDarDash){
                podeDarDash = true;
            }
            if(!escalando){
                tempoEscaladaPassado = tempoEscalada;
            }
        }

        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }

        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
        }
        if(escalando){
            if(tempoEscaladaPassado < 0){
                PararEscalarParede();
            } else {
                tempoEscaladaPassado -= Time.deltaTime;
            }
        }

    }

    void FixedUpdate(){
        if(movimentacaoPadrao){
```

```
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }

    }

    void Pulo(){
        if(!noChao)
            return;
        rb.velocity = new Vector2(rb.velocity.x, 0);
        rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
    }
    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }

    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
```

```
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));


        yield return new WaitForSeconds(tempoDash);


        rb.gravityScale = 1;
        movimentacaoPadrao = true;
        rb.velocity = alvo;
        dashing = false;
    }


    bool ChecaTocandoParede(){
        float extraHeight = 0.15f;
        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);
        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);
        return retorno || retorno2;
    }
    void EscalarParede(){
        escalando = true;
        movimentacaoPadrao = false;
        dashing = false;
        rb.gravityScale = 0;
        StopAllCoroutines();
    }
    void PararEscalarParede(){
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

2.5 - Pulo na parede

Aqui, trocamos a ChecaTocandoParede() para retornar um valor que será a direção do pulo, assim, fazendo com que tocandoParede receba se esse valor for diferente de 0. Na função do pulo, adicionamos um else if para verificar se o jogador está na parede e executa o pulo na parede.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    float direcaoY;
    bool tocandoParede = false;
    bool escalando = false;
    public float velocidadeParede;

    int direcaoParede;
    public float tempoPuloParede;
```

```
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        col = GetComponent<BoxCollider2D>();
    }

    void Update()
    {
        noChao = IsGrounded();
        direcao = Input.GetAxisRaw("Horizontal");
        direcaoY = Input.GetAxisRaw("Vertical");
        direcaoParede = ChecaTocandoParede();
        tocandoParede = direcaoParede != 0;

        if(noChao){
            if(!podeDarDash){
                podeDarDash = true;
            }
            rb.gravityScale = 1;
        }

        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }

        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
```

```
        }


    }


    void FixedUpdate(){
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }


    }


    void Pulo(){
        if(dashing)
            return;
        if(noChao){
            pulando = true;
            rb.velocity = new Vector2(rb.velocity.x, 0);
            rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
        } else if(tocandoParede){
            rb.gravityScale = 1;
            movimentacaoPadrao = false;
            pulando = true;
            Vector2 direcaoPulo = new Vector2(direcaoParede,
1).normalized;
            if (Input.GetKey(KeyCode.Z) && escalando && direcao !=
direcaoParede)
            {
                direcaoPulo = new Vector2(rb.velocity.x, 0);
            }
            escalando = false;
            rb.velocity = new Vector2(0, 0);
```

```
            rb.AddForce(direcaoPulo * forcaPulo, ForceMode2D.Impulse);
            StartCoroutine(PosWallJump());
        }
    }
    IEnumerator PosWallJump(){
        yield return new WaitForSeconds(tempoPuloParede);
        if (Input.GetKey(KeyCode.Z) && tocandoParede)
        {
            EscalarParede();
        }
        else
        {
            movimentacaoPadrao = true;
        }
    }


    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }


    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
```

```
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));


        yield return new WaitForSeconds(tempoDash);


        rb.gravityScale = 1;
        movimentacaoPadrao = true;
        rb.velocity = alvo;
        dashing = false;
    }


    int ChecaTocandoParede(){
        float extraHeight = 0.15f;
        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);
        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);
        if (retorno)
        {
            return 1;
        }
        else if (retorno2)
        {
            return -1;
        }
        else
            return 0;
    }
    void EscalarParede(){
        escalando = true;
        movimentacaoPadrao = false;
        dashing = false;
```

```
        rb.gravityScale = 0;
        StopAllCoroutines();
    }
    void PararEscalarParede(){
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

2.6 - Melhorando o pulo

Pulo do tamanho certo

Criamos uma 'bool' que salva se o pulo está subindo e uma float que será o multiplicador da gravidade ao cair. Precisamos lembrar também de retornar o multiplicador da gravidade para 1 ao tocar no chão.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
```

```
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    float direcaoY;
    bool tocandoParede = false;
    bool escalando = false;
    public float velocidadeParede;

    public float tempoEscalada;
    float tempoEscaladaPassado = 0;

    int direcaoParede;
    public float tempoPuloParede;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        col = GetComponent<BoxCollider2D>();
    }

    void Update()
    {
        noChao = IsGrounded();
        direcao = Input.GetAxisRaw("Horizontal");
        direcaoY = Input.GetAxisRaw("Vertical");
        direcaoParede = ChecaTocandoParede();
        tocandoParede = direcaoParede != 0;

        if(noChao){
            if(!podeDarDash){
                podeDarDash = true;
            }
```

```
            if(!escalando){
                tempoEscaladaPassado = tempoEscalada;
            }
            rb.gravityScale = 1;
        }

        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }

        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
        }
        if(escalando){
            if(tempoEscaladaPassado < 0){
                PararEscalarParede();
            } else {
                tempoEscaladaPassado -= Time.deltaTime;
            }
        }
        if(pulando && !Input.GetKey(KeyCode.C) && movimentacaoPadrao){
            rb.gravityScale = gravidadeMultiplicador;
            pulando = false;
        }

    }

    void FixedUpdate(){
```

```
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }


    }


    void Pulo(){
        if(dashing)
            return;
        if(noChao){
            pulando = true;
            rb.velocity = new Vector2(rb.velocity.x, 0);
            rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
        } else if(tocandoParede){
            rb.gravityScale = 1;
            movimentacaoPadrao = false;
            pulando = true;
            Vector2 direcaoPulo = new Vector2(direcaoParede,
1).normalized;
            if (Input.GetKey(KeyCode.Z) && escalando && direcao !=
direcaoParede)
            {
                direcaoPulo = Vector2.up;
            }
            escalando = false;
            rb.velocity = new Vector2(0, 0);
            rb.AddForce(direcaoPulo * forcaPulo, ForceMode2D.Impulse);
            StartCoroutine(PosWallJump());
        }
    }
    IEnumerator PosWallJump(){
```

```
        yield return new WaitForSeconds(tempoPuloParede);
        if (Input.GetKey(KeyCode.Z) && tocandoParede)
        {
            EscalarParede();
        }
        else
        {
            movimentacaoPadrao = true;
        }
    }

    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }

    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
```

```
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));


        yield return new WaitForSeconds(tempoDash);


        rb.gravityScale = 1;

        movimentacaoPadrao = true;

        rb.velocity = alvo;

        dashing = false;

    }


    int ChecaTocandoParede(){

        float extraHeight = 0.15f;

        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);

        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);

        if (retorno)

        {

            return 1;

        }

        else if (retorno2)

        {

            return -1;

        }

        else

            return 0;

    }

    void EscalarParede(){

        escalando = true;

        movimentacaoPadrao = false;

        dashing = false;

        rb.gravityScale = 0;

        StopAllCoroutines();

    }

    void PararEscalarParede(){
```

```
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

Tempo de descida reduzido

Aqui, só precisamos verificar se o jogador está se movimentando normalmente e se a velocidade é menor que -0.5, assim aplicamos o multiplicador da gravidade.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    float direcaoY;
    bool tocandoParede = false;
    bool escalando = false;
    public float velocidadeParede;
```

```
public float tempoEscalada;
float tempoEscaladaPassado = 0;

public float gravidadeMultiplicador;
bool pulando = false;

int direcaoParede;
public float tempoPuloParede;

void Start()
{
    rb = GetComponent<Rigidbody2D>();
    col = GetComponent<BoxCollider2D>();
}

void Update()
{
    noChao = IsGrounded();
    direcao = Input.GetAxisRaw("Horizontal");
    direcaoY = Input.GetAxisRaw("Vertical");
    direcaoParede = ChecaTocandoParede();
    tocandoParede = direcaoParede != 0;

    if(noChao){
        if(!podeDarDash){
            podeDarDash = true;
        }
        if(!escalando){
            tempoEscaladaPassado = tempoEscalada;
        }
        rb.gravityScale = 1;
    }
```

```
        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }

        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
        }
        if(escalando){
            if(tempoEscaladaPassado < 0){
                PararEscalarParede();
            } else {
                tempoEscaladaPassado -= Time.deltaTime;
            }
        }

        if(pulando && !Input.GetKey(KeyCode.C) && movimentacaoPadrao){
            rb.gravityScale = gravidadeMultiplicador;
            pulando = false;
        }
        if (rb.velocity.y < -0.5f && movimentacaoPadrao)
        {
            rb.gravityScale = gravidadeMultiplicador;
            pulando = false;
        }

    }

    void FixedUpdate(){
```

```
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }


    }


    void Pulo(){
        if(dashing)
            return;
        if(noChao){
            pulando = true;
            rb.velocity = new Vector2(rb.velocity.x, 0);
            rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
        } else if(tocandoParede){
            rb.gravityScale = 1;
            movimentacaoPadrao = false;
            pulando = true;
            Vector2 direcaoPulo = new Vector2(direcaoParede,
1).normalized;
            if (Input.GetKey(KeyCode.Z) && escalando && direcao !=
direcaoParede)
            {
                direcaoPulo = Vector2.up;
            }
            escalando = false;
            rb.velocity = new Vector2(0, 0);
            rb.AddForce(direcaoPulo * forcaPulo, ForceMode2D.Impulse);
            StartCoroutine(PosWallJump());
        }
    }
    IEnumerator PosWallJump(){
```

```
        yield return new WaitForSeconds(tempoPuloParede);
        if (Input.GetKey(KeyCode.Z) && tocandoParede)
        {
            EscalarParede();
        }
        else
        {
            movimentacaoPadrao = true;
        }
    }

    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }

    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
```

```
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));


        yield return new WaitForSeconds(tempoDash);


        rb.gravityScale = 1;

        movimentacaoPadrao = true;

        rb.velocity = alvo;

        dashing = false;

    }


    int ChecaTocandoParede(){

        float extraHeight = 0.15f;

        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);

        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);

        if (retorno)

        {

            return 1;

        }

        else if (retorno2)

        {

            return -1;

        }

        else

            return 0;

    }

    void EscalarParede(){

        escalando = true;

        movimentacaoPadrao = false;

        dashing = false;

        rb.gravityScale = 0;

        StopAllCoroutines();

    }

    void PararEscalarParede(){
```

```
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

Tempo do coyote

Aqui criamos duas 'float': tempo máximo e tempo atual. Verificamos se o jogador está no chão e reiniciamos o tempo atual para o tempo máximo. No pulo não verificamos se o jogador está no chão e sim se o tempo atual é maior que 0.

No tempo de descida reduzido, precisamos adicionar uma verificação sobre o tempo do coyote, para não funcionar enquanto o tempo do coyote estiver ativo. Assim, quando o jogador cair de uma plataforma sem pular, ainda terá um tempo para pular sem sofrer uma gravidade maior.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;
```

```
float direcaoY;
bool tocandoParede = false;
bool escalando = false;
public float velocidadeParede;

public float tempoEscalada;
float tempoEscaladaPassado = 0;

public float gravidadeMultiplicador;
bool pulando = false;

int direcaoParede;
public float tempoPuloParede;

public float tempoCoyote;
float tempoCoyotePassado = 0;

void Start()
{
    rb = GetComponent<Rigidbody2D>();
    col = GetComponent<BoxCollider2D>();
}

void Update()
{
    noChao = IsGrounded();
    direcao = Input.GetAxisRaw("Horizontal");
    direcaoY = Input.GetAxisRaw("Vertical");
    direcaoParede = ChecaTocandoParede();
    tocandoParede = direcaoParede != 0;

    if(noChao){
        if(!podeDarDash){
```

```
                podeDarDash = true;
            }
            if(!escalando){
                tempoEscaladaPassado = tempoEscalada;
            }
            rb.gravityScale = 1;
            if(tempoCoyote != tempoCoyotePassado){
                tempoCoyotePassado = tempoCoyote;
            }
        }

        if(Input.GetKeyDown(KeyCode.C)){
            Pulo();
        }

        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }

        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
        }
        if(escalando){
            if(tempoEscaladaPassado < 0){
                PararEscalarParede();
            } else {
                tempoEscaladaPassado -= Time.deltaTime;
            }
        }

        if(pulando && !Input.GetKey(KeyCode.C) && movimentacaoPadrao){
            rb.gravityScale = gravidadeMultiplicador;
```

```
            pulando = false;
        }
        if (rb.velocity.y < -0.5f && movimentacaoPadrao &&
tempoCoyotePassado <= 0)
        {
            rb.gravityScale = gravidadeMultiplicador;
            pulando = false;
        }

        if(tempoCoyotePassado > 0){
            tempoCoyotePassado -= Time.deltaTime;
        }

    }

    void FixedUpdate(){
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }

    }

    void Pulo(){
        if(dashing)
            return;
        if(tempoCoyotePassado > 0){
            pulando = true;
            rb.velocity = new Vector2(rb.velocity.x, 0);
            rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
        } else if(tocandoParede){
```

```
            rb.gravityScale = 1;
            movimentacaoPadrao = false;
            pulando = true;
            Vector2 direcaoPulo = new Vector2(direcaoParede,
1).normalized;
            if (Input.GetKey(KeyCode.Z) && escalando && direcao !=
direcaoParede)
            {
                direcaoPulo = Vector2.up;
            }
            escalando = false;
            rb.velocity = new Vector2(0, 0);
            rb.AddForce(direcaoPulo * forcaPulo, ForceMode2D.Impulse);
            StartCoroutine(PosWallJump());
        }
        tempoCoyotePassado = 0;
    }
    IEnumerator PosWallJump(){
        yield return new WaitForSeconds(tempoPuloParede);
        if (Input.GetKey(KeyCode.Z) && tocandoParede)
        {
            EscalarParede();
        }
        else
        {
            movimentacaoPadrao = true;
        }
    }

    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }
```

```
    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
        rb.gravityScale = 0;
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));


        yield return new WaitForSeconds(tempoDash);


        rb.gravityScale = 1;
        movimentacaoPadrao = true;
        rb.velocity = alvo;
        dashing = false;
    }


    int ChecaTocandoParede(){
        float extraHeight = 0.15f;
        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);
        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);
        if (retorno)
        {
```

```
            return 1;
        }
        else if (retorno2)
        {
            return -1;
        }
        else
            return 0;
    }
    void EscalarParede(){
        escalando = true;
        movimentacaoPadrao = false;
        dashing = false;
        rb.gravityScale = 0;
        StopAllCoroutines();
    }
    void PararEscalarParede(){
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

“Buffering” do pulo

Criamos duas ‘float’: tempo máximo de buffering e tempo atual. Ao apertarmos a tecla C, reiniciamos o tempo. Agora, não conferimos se a tecla é pressionada para chamar o Pulo(), conferimos se o tempo atual é maior que 0.

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

public class PlayerScript : MonoBehaviour
{
    public float velocidade;
```

```
    public float forcaPulo;

    float direcao;
    Rigidbody2D rb;
    BoxCollider2D col;
    public LayerMask layerChao;
    bool noChao;

    bool movimentacaoPadrao = true;
    bool podeDarDash = true;
    bool dashing = false;
    public float forcaDash;
    public float tempoDash;
    Vector2 direcaoDash;

    float direcaoY;
    bool tocandoParede = false;
    bool escalando = false;
    public float velocidadeParede;

    public float tempoEscalada;
    float tempoEscaladaPassado = 0;

    public float gravidadeMultiplicador;
    bool pulando = false;

    int direcaoParede;
    public float tempoPuloParede;

    public float tempoCoyote;
    float tempoCoyotePassado = 0;

    public float tempoBufferingPulo;
    float tempoBufferingPuloPassado = 0;
```

```
void Start()
{
    rb = GetComponent<Rigidbody2D>();
    col = GetComponent<BoxCollider2D>();
}

void Update()
{
    noChao = IsGrounded();
    direcao = Input.GetAxisRaw("Horizontal");
    direcaoY = Input.GetAxisRaw("Vertical");
    direcaoParede = ChecaTocandoParede();
    tocandoParede = direcaoParede != 0;

    if(noChao){
        if(!podeDarDash){
            podeDarDash = true;
        }
        if(!escalando){
            tempoEscaladaPassado = tempoEscalada;
        }
        rb.gravityScale = 1;
        if(tempoCoyote != tempoCoyotePassado){
            tempoCoyotePassado = tempoCoyote;
        }
    }

    if(Input.GetKeyDown(KeyCode.C)){
        tempoBufferingPuloPassado = tempoBufferingPulo;
    }
    if(tempoBufferingPuloPassado > 0){
        tempoBufferingPuloPassado -= Time.deltaTime;
        Pulo();
```

```
        }



        if(Input.GetKeyDown(KeyCode.X)){
            Dash();
        }


        if(Input.GetKeyDown(KeyCode.Z) && tocandoParede){
            EscalarParede();
        } else if((Input.GetKeyUp(KeyCode.Z) || !tocandoParede) &&
escalando){
            PararEscalarParede();
        }
        if(escalando){
            if(tempoEscaladaPassado < 0){
                PararEscalarParede();
            } else {
                tempoEscaladaPassado -= Time.deltaTime;
            }
        }


        if(pulando && !Input.GetKey(KeyCode.C) && movimentacaoPadrao){
            rb.gravityScale = gravidadeMultiplicador;
            pulando = false;
        }
        if (rb.velocity.y < -0.5f && movimentacaoPadrao)
        {
            rb.gravityScale = gravidadeMultiplicador;
            pulando = false;
        }


        if(tempoCoyotePassado > 0){
            tempoCoyotePassado -= Time.deltaTime;
        }
```

```
    }

    void FixedUpdate(){
        if(movimentacaoPadrao){
            rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }
    }

    void Pulo(){
        if(dashing)
            return;
        if(tempoCoyotePassado > 0){
            pulando = true;
            rb.velocity = new Vector2(rb.velocity.x, 0);
            rb.AddForce(Vector2.up * forcaPulo, ForceMode2D.Impulse);
        } else if(tocandoParede){
            rb.gravityScale = 1;
            movimentacaoPadrao = false;
            pulando = true;
            Vector2 direcaoPulo = new Vector2(direcaoParede,
1).normalized;
            if (Input.GetKey(KeyCode.Z) && escalando && direcao !=
direcaoParede)
            {
                direcaoPulo = Vector2.up;
            }
            escalando = false;
            rb.velocity = new Vector2(0, 0);
            rb.AddForce(direcaoPulo * forcaPulo, ForceMode2D.Impulse);
            StartCoroutine(PosWallJump());
        }
```

```
        tempoCoyotePassado = 0;
        tempoBufferingPuloPassado = 0;
    }
    IEnumerator PosWallJump(){
        yield return new WaitForSeconds(tempoPuloParede);
        if (Input.GetKey(KeyCode.Z) && tocandoParede)
        {
            EscalarParede();
        }
        else
        {
            movimentacaoPadrao = true;
        }
    }

    bool IsGrounded(){
        float extraHeight = 0.1f;
        bool retorno = Physics2D.BoxCast(col.bounds.center, new
Vector2(col.bounds.size.x - 0.005f, col.bounds.size.y), 0, Vector2.down,
extraHeight, layerChao);
        return retorno;
    }

    void Dash(){
        if(dashing || !podeDarDash){
            return;
        }
        direcaoDash = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;
        if(direcaoDash == Vector2.zero){
            return;
        }
        dashing = true;
        movimentacaoPadrao = false;
        podeDarDash = false;
        rb.gravityScale = 0;
```

```
        StartCoroutine("PosDash");
    }
    IEnumerator PosDash(){
        Vector2 alvo = new Vector2(direcaoDash.x * velocidade,
direcaoDash.y * (forcaDash / 2));


        yield return new WaitForSeconds(tempoDash);


        rb.gravityScale = 1;
        movimentacaoPadrao = true;
        rb.velocity = alvo;
        dashing = false;
    }


    int ChecaTocandoParede(){
        float extraHeight = 0.15f;
        bool retorno = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.left, extraHeight, layerChao);
        bool retorno2 = Physics2D.BoxCast(col.bounds.center,
col.bounds.size, 0, Vector2.right, extraHeight, layerChao);
        if (retorno)
        {
            return 1;
        }
        else if (retorno2)
        {
            return -1;
        }
        else
            return 0;
    }
    void EscalarParede(){
        escalando = true;
        movimentacaoPadrao = false;
        dashing = false;
        rb.gravityScale = 0;
```

```
        StopAllCoroutines();
    }
    void PararEscalarParede(){
        escalando = false;
        movimentacaoPadrao = true;
        rb.gravityScale = 1;
    }
}
```

2.7 - Controle aéreo

Criamos uma 'float' puloParedeLerp e aplicamos esse código no FixedUpdate().

```
    void FixedUpdate(){
        if(movimentacaoPadrao){
            if (noChao)
                rb.velocity = new Vector2(direcao * velocidade,
rb.velocity.y);
            else
                rb.velocity = Vector2.Lerp(rb.velocity, new
Vector2(direcao * velocidade, rb.velocity.y), puloParedeLerp *
Time.deltaTime);
        } else if(dashing){
            rb.velocity = direcaoDash * forcaDash;
        } else if(escalando){
            rb.velocity = new Vector2(0, direcaoY * velocidadeParede);
        }
    }
```