# Aplicação de Derivadas no jogo

Derivadas foram aplicadas em diversas partes do jogo, para calcular a gravidade, ajustar tempo do movimento para que o personagem não vá mais rápido dependendo do frame rate do jogo, entre outras.

Um dos códigos que usa derivadas é o código de movimentação do personagem, mais especificamente na parte que controla o pulo do personagem.

```
using Unity.VisualScripting;
using UnityEngine;

public class Movimento : MonoBehaviour
{

    private float lastStepTime = 0;
    public float delayBetweenSteps = 0.5f;
    public  CharacterController character;
    private Animator animator;
    private Vector2 input;

    private float velocidade = 4f;
    private float forcaY;
    private bool podePular = true;

    public bool estaNoChao;
    [SerializeField] private LayerMask colisaoLayer;
    [SerializeField] public Transform peDoPersonagem;
    [SerializeField] private AudioClip passos;

    private AudioSource audioSource;
    private float gravidade = -20f;
    private float forcaPulo = 10f;

    void Start()
    {
        character = GetComponent<CharacterController>();
        animator = GetComponent<Animator>();
    }

    void Update()
    {
        // Movimento lateral no eixo X (2.5D)
        input.Set(Input.GetAxisRaw("Horizontal"), 0);
```

```
        Vector3 move = new Vector3(input.x * velocidade, 0, 0);



        // Verifica se está no chão
        estaNoChao = Physics.CheckSphere(peDoPersonagem.position, 0.2f,
colisaoLayer);
        animator.SetBool("estatNoChao", estaNoChao);

        if (estaNoChao && forcaY < 0)
        {
            forcaY = -2f;
            podePular = true;
        }

        // Pulo
        if (Input.GetKeyDown(KeyCode.Space) && estaNoChao && podePular)
        {
            forcaY = forcaPulo;
            animator.SetTrigger("pular");
            podePular = false;
        }

        // Gravidade
        forcaY += gravidade * Time.deltaTime;
        move.y = forcaY;

        // Move personagem
        character.Move(move * Time.deltaTime);

        // Espelha o personagem na direção correta (lado)
        if (input.x > 0)
        {
            transform.rotation = Quaternion.Euler(0, 90, 0);  // Direita
        }
        else if (input.x < 0)
        {
            transform.rotation = Quaternion.Euler(0, -90, 0); // Esquerda
        }

        // Animação de corrida
        if (Mathf.Abs(input.x) > 0.1f)
        {
            animator.SetBool("correndo", true);



        }
        else
        {
```

```
            animator.SetBool("correndo", false);
        }



        if ((Mathf.Abs(input.x) > 0.1f) && (estaNoChao) && (Time.time -
lastStepTime >= delayBetweenSteps)) //se estiver andando, estiver no chão
e se faz algum tempo desde que o som de passos tocou
        {
            AudioSource.PlayClipAtPoint(passos, transform.position, 1f);
// toca som de passos
            lastStepTime = Time.time; // atualiza tempo em que som passos
tocou
        }


    }



}
```

Onde na linha 60 há uma derivada que aplica aceleração ao percorrer do tempo por conta da gravidade.

Ela está fazendo uma integração numérica, que é baseado em uma derivada.

- “gravidade” é uma aceleração, que na física é a derivada da velocidade ao longo do tempo:

$$a(t) = \frac{dv(t)}{dt}$$

- Então quando você multiplica por uma etapa de tempo (Time.deltaTime), você está estimando a mudança da velocidade:

$$\Delta v = a \cdot \Delta t$$

- Então você adiciona a velocidade atual (forcaY), efetivamente integrando aceleração ao longo do tempo para atualizar a velocidade:

$$Vnew = Vold + a \cdot \Delta t$$

Em conclusão, a derivada é utilizada para incrementalmente atualizar a velocidade, fazendo com que o pulo seja mais natural.