Plataformas

Prof. Me. Hélio Esperidião

Plataformas

- Em jogos, uma "plataforma" refere-se a uma superfície ou objeto sobre o qual os personagens ou elementos do jogo podem se mover, pular ou interagir.
- Plataformas são componentes essenciais em muitos tipos de jogos, especialmente nos gêneros de plataformas (platformers) e jogos de ação e aventura.

Características das Plataformas em Jogos

- Estáticas: Plataformas que não se movem e permanecem em uma posição fixa no jogo.
- Dinâmicas: Plataformas que se movem, mudam de posição, rotação ou até mesmo escala ao longo do tempo ou em resposta a eventos no jogo. Elas podem mover-se de forma linear, rotativa, aparecer e desaparecer, entre outras variações

Colisão e Física

- As plataformas geralmente têm colisores (colliders) que detectam quando um personagem ou objeto interage com elas, permitindo que os personagens se movam ou saltem sobre elas de forma realista.
- A física das plataformas pode ser ajustada para criar diferentes experiências de jogo, como plataformas escorregadias, trampolins ou plataformas que quebram sob peso excessivo.

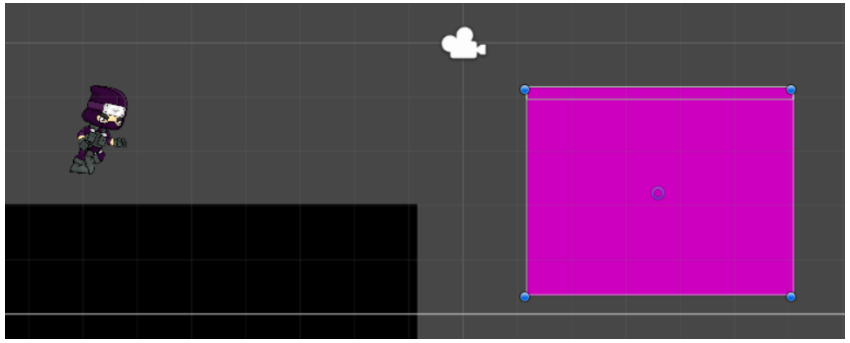
Transform

É um componente fundamental no Unity que representa a posição, rotação e escala de um objeto no espaço 3D ou 2D. Cada GameObject

No Unity o Transform é associado automaticamente, permite manipular esses aspectos do objeto de forma intuitiva e eficaz.

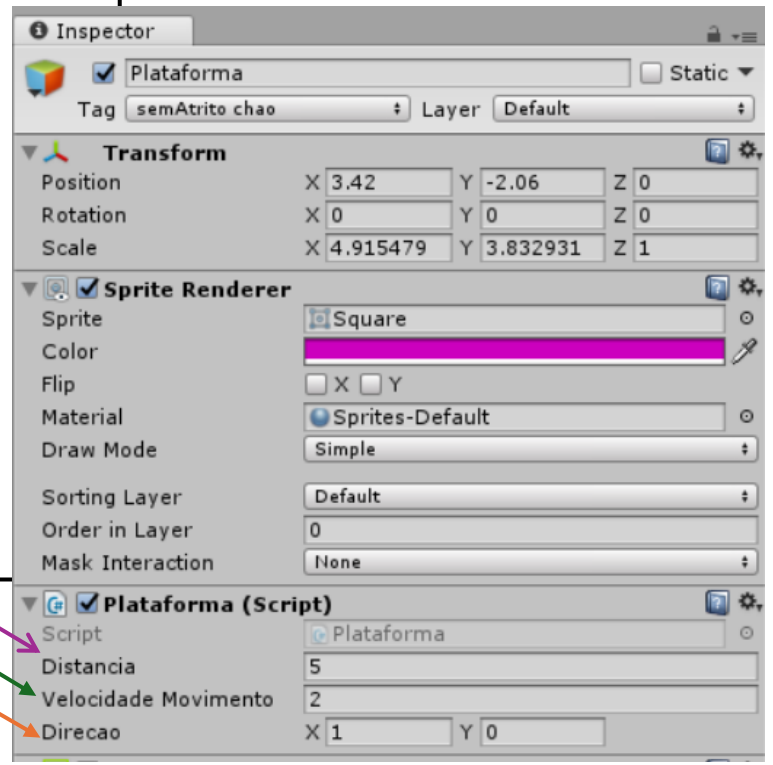
Configuração inicial

- Personagem:
  - Colisor
  - Corpo Rígido
  - Tag: **Player**
- Plataforma:
  - Scritp: Plataforma.cs
  - Colisor;
  - Tag: Plataforma



Atributos

```
using UnityEngine;
public class Plataforma : MonoBehaviour{
    Transform TransformPlataforma; // Referência ao componente Transform da plataforma
    Vector2 PosicaoInicial; // Armazena a posição inicial da plataforma
    float TempoDecorrido; // Acumula o tempo passado para controle do movimento
    // Variáveis públicas que podem ser ajustadas no Inspector
    public float Distancia; // Distância máxima do movimento
    public float VelocidadeMovimento; // Velocidade do movimento
    public Vector2 Direcao; // Direção do movimento (vetor normalizado ou não):
    // - (1, 0) → Movimento horizontal para direita
    // - (-1, 0) → Movimento horizontal para esquerda
    // - (0, 1) → Movimento vertical para cima
    // - (0, -1) → Movimento vertical para baixo
    // - (1, 1) → Movimento diagonal superior direito (45°)
    // - (-1, 1) → Movimento diagonal superior esquerdo (135°)
    // - (-1, -1) → Movimento diagonal inferior esquerdo (225°)
    // - (1, -1) → Movimento diagonal inferior direito (315°)
    // - (0.5f, 1) → Movimento em ângulo de ~63.4°
    // - (1, 0.5f) → Movimento em ângulo de ~26.6°
    // Qualquer combinação de valores x e y para direções personalizadas
```



Start()

```
    void Start()
    {
        // Obtém o componente Transform do objeto atual
        TransformPlataforma = GetComponent<Transform>();
        // Armazena a posição inicial do objeto
        PosicaoInicial = TransformPlataforma.position;
        // Inicializa o tempo decorrido como zero
        TempoDecorrido = 0f;
    }
```

Update()

```
// Método chamado a cada frame
void Update()
{
    mover(); // Chama a função de movimento a cada frame
}
```

Mover()

```
    // Função que controla o movimento da plataforma
    void mover()
    {
        // Incrementa o tempo decorrido multiplicado pela velocidade
        TempoDecorrido += Time.deltaTime * VelocidadeMovimento;
        // Calcula um valor que vai e volta entre 0 e Distancia (movimento de vai-e-vem)
        float movimento = Mathf.PingPong(TempoDecorrido, Distancia);

        // Atualiza a posição do objeto: posição inicial + direção * valor do movimento
        transform.position = PosicaoInicial + Direcao.normalized * movimento;
    }
```

transform.parent

- O transform.parent no Unity é uma propriedade do componente Transform que permite definir ou obter o pai de um objeto na hierarquia de objetos da cena.
- A hierarquia de objetos é fundamental para organizar e manipular a cena, especialmente para agrupar objetos e criar relações de dependência entre eles.
- Definindo o Pai de um Objeto
  - Você pode definir o pai de um objeto ao atribuir um novo Transform à propriedade transform.parent. Isso faz com que o objeto se torne um filho do novo pai, herdando a posição, rotação e escala relativas ao pai.

OnCollisionEnter2D ()

```
// Método chamado quando ocorre uma colisão 2D
void OnCollisionEnter2D(Collision2D objetoTocado)
{
    // Recupera a tag do objeto que colidiu
    string tag = objetoTocado.gameObject.tag;
    // Verifica se a tag contém "Player"
    if (tag.Contains("Player") == true)
    {
        // Faz o jogador se tornar filho da plataforma (move-se junto com ela)
        objetoTocado.transform.parent = TransformPlataforma;
    }
}
```

OnCollisionExit2D ()

```
// Método chamado quando uma colisão 2D termina
void OnCollisionExit2D(Collision2D objetoParouTocar)
{
    // Recupera a tag do objeto que parou de colidir
    string tag = objetoParouTocar.gameObject.tag;
    // Verifica se a tag contém "Player"
    if (tag.Contains("Player") == true)
    {
        // Remove o parentesco, fazendo o jogador parar de seguir a plataforma
        objetoParouTocar.transform.parent = null;
    }
}
```

Valores possíveis de Direção

// - (1, 0) → Movimento horizontal para direita
// - (-1, 0) → Movimento horizontal para esquerda
// - (0, 1) → Movimento vertical para cima
// - (0, -1) → Movimento vertical para baixo
// - (1, 1) → Movimento diagonal superior direito (45°)
// - (-1, 1) → Movimento diagonal superior esquerdo (135°)
// - (-1, -1) → Movimento diagonal inferior esquerdo (225°)
// - (1, -1) → Movimento diagonal inferior direito (315°)
// - (0.5f, 1) → Movimento em ângulo de ~63.4°
// - (1, 0.5f) → Movimento em ângulo de ~26.6°