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

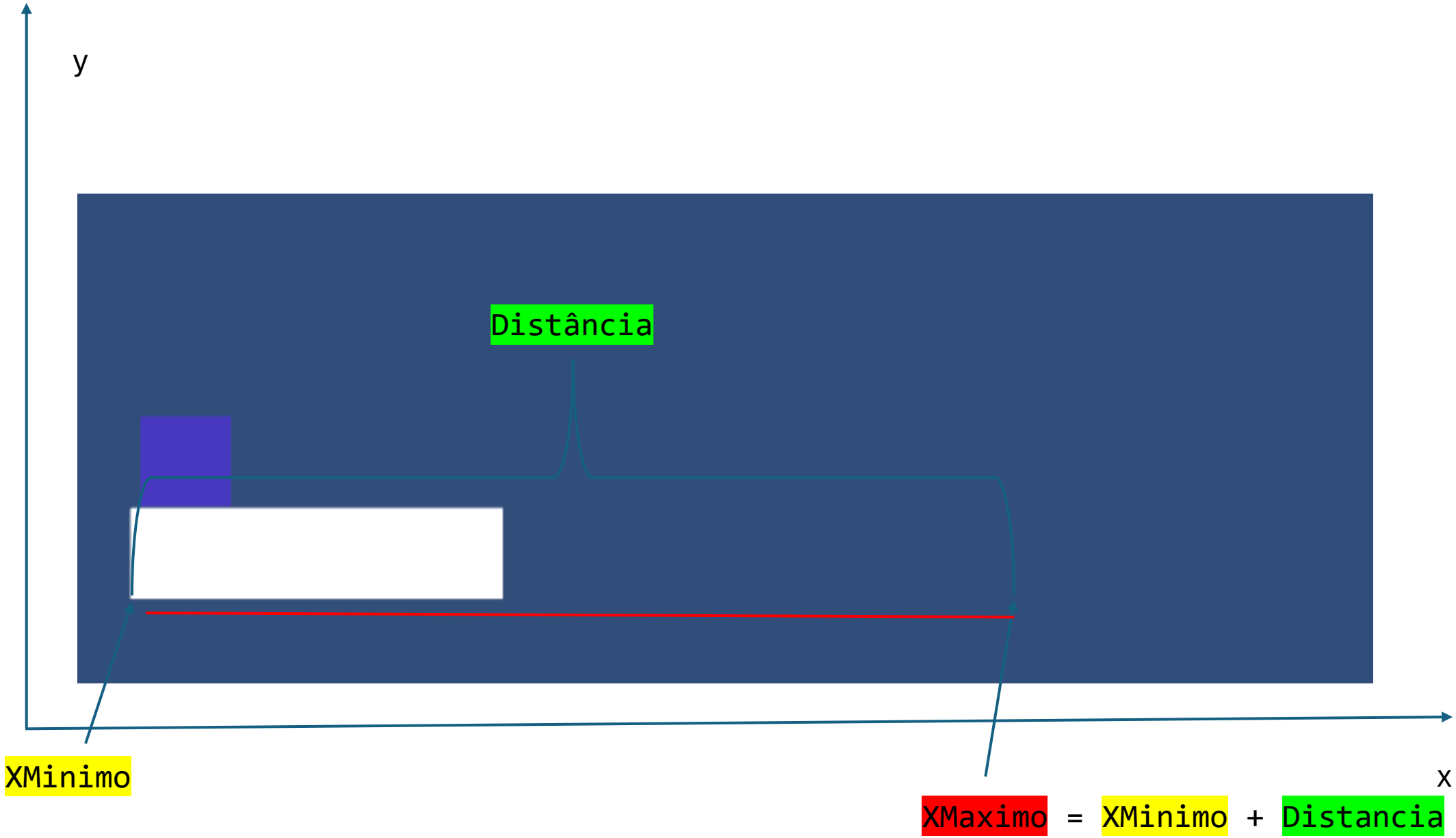
- As plataformas geralmente têm colisores (colliders) que detectam quando um personagem ou objeto interage com elas, permitindo que os personagens se movam ou saltem sobre elas de forma realista.
- A física das plataformas pode ser ajustada para criar diferentes experiências de jogo, como plataformas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

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


y
Distância
XMinimo
x
XMaximo = XMinimo + Distancia

Plataforma.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Plataforma : MonoBehaviour{
    //distancia máxima da posição inicial;
    public float Distancia;
    //velocidae de descolamento da plataforma
    public float Velocidade;
    // valor maximo em x que a plataforma pode chegar.
    float XMaximo;
    //valor minimo em X que a plataforma pode chegar.
    float XMinimo;
    //direcao do movimento da plataforma;
    float Direcao;
    //posição inicial em X da plataforma
    float XInicial;
    //posição atual em X da plataforma
    float XAtual;
    //posição atual em Y da plataforma
    float YAtual;
    //objeto que determina o posicionamento da plataforma
    Transform transformPlataforma;
```

```
void Start(){
        //distancia que a plataforma irá percorrer.
        Distancia = 2.5f;
        //velocidade de deslocamento da plataforma
        Velocidade = 5;
        //inicializa o transform da plataforma
        transformPlataforma = gameObject.GetComponent<Transform>();
        //no inicio do jogo recupera a posição inicial x e posição y.
        XAtual = transformPlataforma.position.x;
        YAtual = transformPlataforma.position.y;
        //é igual a posição inicial da plataforma
        XMinimo = transformPlataforma.position.x;
        XMaximo = XMinimo + Distancia;
        // a plataforma inicializa indo para direita.
        Direcao = 1;
    }
```

Update()

```
void Update()
    {
        MvimentoHorizontalPlataforma();
    }
```

```
void MvimentoHorizontalPlataforma(){
        //a cada frame modifica o valor de XAtual;
        //Time.deltaTime normaliza para que independente da taxa de frames
        //a movimentação tenha a mesma velocidade.
        XAtual = XAtual + (Velocidade * Direcao ) * Time.deltaTime;
        //se a posição atual  é maior que o x Maximo é invertido o lado.
        if (XAtual > XMaximo){
            Direcao = -1;
        }
        //se a posição atual for menor que o mínimo inverte o sentido
        if (XAtual < XMinimo){
            Direcao = 1;
        }
        //determina a nova posição da plataforma.
        transformPlataforma.position = new Vector2(XAtual, YAtual);
    }
```

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.

```
void OnCollisionEnter2D(Collision2D objetoTocado){
        //recupera a tag do objeto tocado
        string tag = objetoTocado.gameObject.tag;
        if (tag.Contains("Player") == true){
            //define o parent do tranform do personagem como igual
            //ao transform do objeto tocado
            //o objeto tocado(player) passa a influenciar o movimento
            //do personagem.
            //ap entrar na plataforma o personagem se movimenta junto com a plataforma.
            objetoTocado.transform.parent = transformPlataforma;
        }
    }
```

```
void OnCollisionEnter2D(Collision2D objetoTocado){
        //recupera a tag do objeto tocado
        string tag = objetoTocado.gameObject.tag;
        if (tag.Contains("Player") == true){
            //define o parent do tranform do personagem como igual
            //ao transform do objeto tocado
            //o objeto tocado(player) passa a influenciar o movimento
            //do personagem.
            //ap entrar na plataforma o personagem se movimenta junto com a plataforma.
            objetoTocado.transform.parent = transformPlataforma;
        }

    }
```

```
 void OnCollisionExit2D(Collision2D objetoParouTocar)
    {
        //recupera a tag do objeto que parou de ser tocado

        string tag = objetoParouTocar.gameObject.tag;
        //verifica se o objeto que parou de ser tocado é o Player
        if (tag.Contains("Player") == true){
            //define o parent do personagem como null
            //ao sair da plataforma, a plataforma vai embora
            //e o personagem fica.
            objetoParouTocar.transform.parent = null;
        }

    }
}//fecha a classe
```

Simplificando o script

- É possível utilizar um script simplificado para movimentar uma plataforma.
- Vantagem: Fácil de programar.
- Desvantagem: menos controle do programador quanto a posição da plataforma.

Vector2.MoveTowards()

- É uma função do Unity que move um ponto atual em direção a um ponto alvo em uma taxa constante.
- Isso é útil para criar movimento linear suave entre dois pontos.
- A função garante que o movimento não ultrapasse o ponto alvo, mesmo que a velocidade de movimento seja alta.

- MoveTowards(Vector2 current, Vector2 target, float maxDistanceDelta);
- current: A posição atual do objeto.
- target: A posição alvo para a qual o objeto se moverá.
- maxDistanceDelta: A distância máxima que o objeto pode se mover em um único frame.

Novo script: PlataformaSimples

```
using UnityEngine;
public class PlataformaSimples : MonoBehaviour{
    //Objeto que define o posicionamento da plataforma.
    Transform TransformPlataforma;
    //vetor com a posição inical da flataforma.
    Vector2 PosicaoInicial;
    //vetor com a posição final da flataforma.
    Vector2 PosicaoFinal;
    //contem a posição para onde deve se mover.
    //hora é posição inical, hora a posição final;
    Vector2 PosicaoDestino;
    //Velocidade de movimentação da plataforma
    float Velocidade;
    //distancia que deve ser percorrida
    float Distancia;
```

```
void Start() {
        //inicializa o transform do objeto
        TransformPlataforma = gameObject.GetComponent<Transform>();
        //velocidade inicial
        Velocidade = 5.0f;
        //distancia que será percorrida pela plataforma.
        Distancia = 2.5f;
        //recupera a posição inical da plataforma.
        float xInicial = TransformPlataforma.position.x;
        float yInicial = TransformPlataforma.position.y;
        //cria um vetor com a posição inical.
        PosicaoInicial = new Vector2(xInicial, yInicial);
        //utilizando a distância, cria o vetor com a posição final.
        PosicaoFinal = new Vector2(xInicial + Distancia, yInicial);
        //determina a posição de destino inical.
        PosicaoDestino = PosicaoFinal;
    }
```

void Update()

```
void Update()
    {
        MovimentoSimplesPlataforma();


    }
```

```
void MovimentoSimplesPlataforma(){
        // Move a plataforma em direção ao destino
        transform.position = Vector2.MoveTowards(TransformPlataforma.position,
PosicaoDestino, Velocidade * Time.deltaTime);

        // Troca o destino quando a plataforma atinge um dos pontos
        if ((Vector2)TransformPlataforma.position == PosicaoDestino)
        {
            if (PosicaoDestino == PosicaoInicial)
            {
                PosicaoDestino = PosicaoFinal;
            }
            else
            {
                PosicaoDestino = PosicaoInicial;
            }
        }
    }
```

MoveTowards(Vector2 current, Vector2 target, float maxDistanceDelta);

```
void OnCollisionEnter2D(Collision2D objetoTocado){
        //recupera a tag do objeto tocado
        string tag = objetoTocado.gameObject.tag;
        if (tag.Contains("Player") == true){
           //define o parent do tranform do personagem como igual
           //ao transform do objeto tocado
           //o objeto tocado(player) passa a influenciar o movimento
           //do personagem.
           //ao entrar na plataforma o personagem se movimenta junto com a plataforma.
            objetoTocado.transform.parent = TransformPlataforma;
        }

    }
```

```
 void OnCollisionExit2D(Collision2D objetoParouTocar){
        //recupera a tag do objeto que parou de ser tocado
        string tag = objetoParouTocar.gameObject.tag;
        //verifica se o objeto que parou de ser tocado é o Player
        if (tag.Contains("Player") == true){
            //define o parent do personagem como null
            //ao sair da plataforma, a plataforma vai embora
            //e o personagem fica.
            objetoParouTocar.transform.parent = null;
        }

    }

}
```