



# Além de jogar, eu faço jogos!

Uma parceria de sucesso entre o CPDI e o  
Instituto Cooperforte



Comitê para Democratização  
da Informática

**Professora Juliana Oliveira**



# Pauta – Aula 18

O que vamos aprender hoje

**01**

Finalização das propriedades  
de corpos físicos

**02**

Tap, touch e multitouch



# Propriedades do corpo

## **object.bodyType**

É um valor para o tipo de corpo físico. Incluem dynamic, static ou kinematic.

## **object.isAwake**

É o valor booleano para determinar se o corpo tem permissão para dormir. Manter um corpo acordado tem uma sobrecarga de desempenho maior e geralmente não é necessário porque uma colisão com outro corpo o acordará automaticamente.



## **object.isBodyActive**

usado para definir ou obter o estado ativo atual do corpo. Corpos inativos não são destruídos, mas são removidos da simulação física e deixam de interagir com outros corpos. Não podemos alterar o estado ativo do corpo durante um evento de colisão, portanto, devemos enfileirar esse evento após um pequeno e imperceptível atraso.

## **object.isSensor**

Define a propriedade interna de sensor em todos os elementos do corpo. Como atua em todos os elementos do corpo, substitui incondicionalmente qualquer configuração de sensor nos elementos individuais.

# Métodos do corpo

## **object:setLinearVelocity()**

Define os componentes x e y para a velocidade linear do corpo, em pixels por segundo.

## **object:getLinearVelocity()**

Retorna os componentes x e y para a velocidade linear do corpo, em pixels por segundo. Esta função aproveita o fato de que Lua pode retornar vários valores, neste caso ambas as velocidades lineares.



## **object:applyForce()**

Aplica os componentes xey especificados de uma força linear em um determinado ponto dentro das coordenadas mundiais. Se o ponto alvo for o centro de massa do corpo, ele tenderá a empurrar o corpo em linha reta. Se o ponto alvo estiver deslocado do centro, o corpo girará em torno de seu centro de massa. Para objetos simétricos, o centro de massa e o centro do objeto terão a mesma posição.



## **object:applyLinearImpulse()**

semelhante a `object:applyForce()`, exceto que um impulso é uma única e momentânea sacudida de força. O impulso pode ser aplicado a qualquer ponto do corpo (seja o centro de massa ou um ponto de deslocamento).

## **object:applyTorque()**

Aplica uma força rotacional ao corpo. Valores positivos resultarão em torque no sentido horário; valores negativos resultarão em torque no sentido anti-horário. O corpo irá girar em torno de seu centro de massa.

## **object:applyAngularImpulse()**

Semelhante a `object:applyTorque()`, exceto que um impulso é uma única e momentânea sacudida de força.

## **object:resetMassData()**

É útil se os dados de massa padrão para o corpo foram substituídos. Esta função redefine para a massa calculada a partir das formas.

## **Força e impulso**

Uma pergunta comum é se aplicamos força ou impulso a um corpo. A diferença é que um impulso serve para simular um choque/chute imediato no corpo, enquanto força (e torque) é algo exercido ao longo do tempo. Portanto, para obter uma simulação de força/torque realista, você deve aplicá-la continuamente em cada ciclo de aplicação, enquanto desejar que a força continue.



# Detecção de tap

Eventos de tap são a forma mais básica de interatividade da tela. Um tap é essencialmente representado pelo usuário tocando a tela e decolando no mesmo ponto aproximado. O evento de tap só é considerado bem sucedido se o usuário tocar e soltar nesse ponto.

Podemos ouvir eventos de tap registrando um ouvinte "tap" nos objetos de exibição mais comuns. Por exemplo:

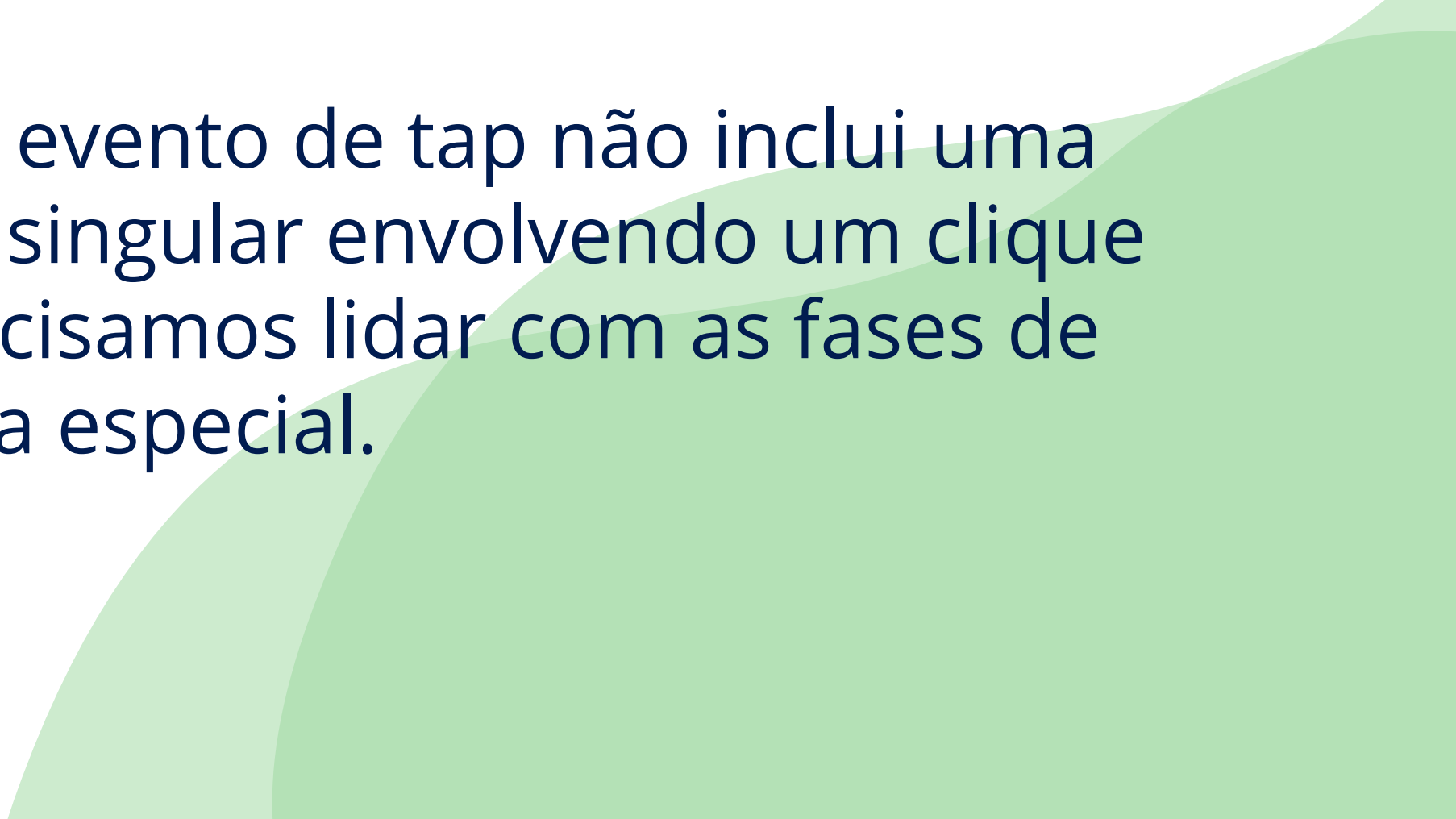
```
local function detectarTap( event )  
    -- Código executado quando o botão é tocado.  
  
    print( "Objeto tocado: " .. tostring(event.target) ) -- "event.target" é o objeto tocado  
    -- tostring: para sequenciar  
  
    return true  
end  
  
local botaoTap = display.newRect( 200, 200, 200, 50 )  
botaoTap:addEventListener( "tap", detectarTap ) -- Adicione um ouvinte "tap" ao objeto
```



## As propriedades de event retornadas de um tap incluem:

- `event.target` — referência ao objeto que foi tocado.
- `event.name` — valor da cadeia de `"tap"`.
- `event.numTaps` — o número de toques na tela. O atraso padrão entre o que é considerado o próximo toque na sequência é `0`, mas esse tempo pode ser ajustado com a função [system.setTapDelay\(\)](#).
- `event.x` / `event.y` — a posição **x** e **y** da **torneira**, em coordenadas de conteúdo.

Ao contrário dos eventos de touch, o evento de tap não inclui uma propriedade `phase`. O tap é uma ação singular envolvendo um clique (toque e liberação), portanto não precisamos lidar com as fases de nenhuma maneira especial.



# Filtrando tap duplos

Utilizando a propriedade `event.numTaps` podemos determinar facilmente se um objeto foi tocado várias vezes e ignorar simultaneamente toques únicos no objeto. Para conseguir isso, basta garantir que `event.numTaps` seja igual a 2 ou superior e ignorar `return true` em todos os outros casos:

```
local function tapDuplo( event )  
    if ( event.numTaps == 2 ) then  
        print( "Objeto tocado duas vezes: " .. tostring(event.target) )  
    else  
        return true  
    end  
end  
  
local botaoTapDuplo = display.newRect( 100, 100, 200, 50 )  
botaoTapDuplo:addEventListener( "tap", tapDuplo )
```

# DESAFIO!!!

Adicione o evento TAP na imagem A com a detecção de tap e na imagem B com a detecção de tap duplo. Os nomes de variáveis não podem ser iguais aos dos exemplos.

**20 minutos para execução.**

