



# 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 19

O que vamos aprender hoje

**01**

Tap, touch e multitouch

# 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:

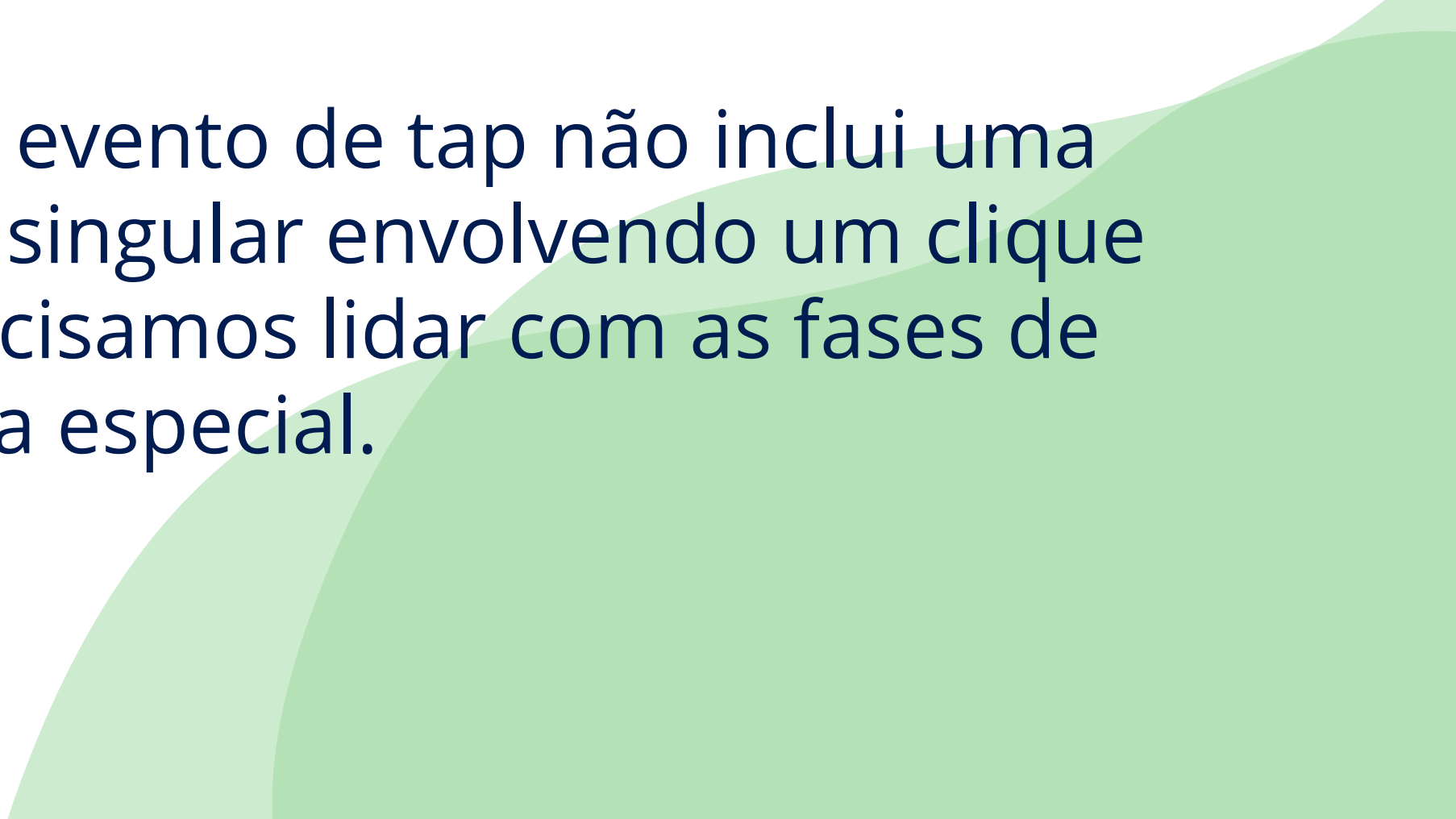
```
local function detectarTap( event )  
    -- Código executado quando o botão é tocado.  
  
    print( "Objeto tocado: " .. tostring(event.target) ) -- "event.target" é o objeto tocado  
    -- tostring: para sequenciar  
  
    return true  
end  
  
local botaoTap = display.newRect( 200, 200, 200,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

# Detecção de touch

Os eventos de touch fornecem um nível muito maior de interatividade da tela. Usando o touch, podemos detectar quando o usuário toca a tela pela primeira vez e quando o toque é retirado da tela. Também podemos acompanhar o movimento do toque conforme ele se move pela tela.

Para conseguir isso, temos a propriedade `event.phase` em quatro estados:

- `"began"` — indica que um toque foi iniciado na tela.
- `"moved"` — indica que um toque se moveu na tela.
- `"ended"` — indica que um toque foi retirado da tela.
- `"cancelled"` — indica que o **sistema** cancelou o rastreamento do toque (não confundir com `"ended"`).

Para ouvir um evento de touch registramos "touch" nos objetos de exibição. Como por exemplo:

```
local function detectarTouch (event)
    print ("Localização X do toque" .. event.x)
    print ("Localização Y do toque" .. event.y)
end

local botao = display.newRect (100, 300, 200, 50)
botao:addEventListener ("touch", detectarTouch)
```

Aqui temos um exemplo das fases de touch:

```
local function fasesToque( event )

    if ( event.phase == "began" ) then
        print( "Objeto tocado = " .. tostring(event.target) )
    elseif ( event.phase == "moved" ) then
        print( "localização de toque nas seguintes coordenadas = X:" .. event.x .. ", Y:" .. event.y )
    elseif ( event.phase == "ended" ) then
        print( "Touch terminado no objeto: " .. tostring(event.target) )
    end

    return true
end

local botaoTouch = display.newRect( 200, 400, 200, 50 )
botaoTouch:addEventListener( "touch", fasesToque )
```



As propriedades de event retornadas de um touch incluem:

- **event.id** - um identificador exclusivo que distingue entre vários toques em diferentes eventos de toque.
- **event.target** - referência ao objeto que foi tocado.
- **event.name** - valor da cadeia de "touch".
- **event.phase** - a fase do toque.
- **event.pressure** - número que indica a pressão de um toque, normalmente usado em IOS compatíveis para detectar eventos "3D touch".
- **event.time** - o tempo em milissegundos desde o início do jogo, acessível a partir da função touch listener.
- **event.x/event.y** - a posição x e y do toque.
- **event.xStart/event.yStart** - a posição x e y do toque da fase de sequencia de toque (began).

# Desafio!!!!

Crie um evento de touch para que ao tocar no objeto ele suma da tela.



# Multitoque



Ativar um multitoque em um jogo permite detectar e lidar com vários toques do usuário na tela ao mesmo tempo.

**Por padrão o multitoque está desabilitado, então antes de usá-lo precisamos habilitar.**

Com o multitoque ativado, registramos um evento de touch e em seguida, comparamos a propriedade `event.id` para determinar qual sequência de eventos de toque específica está sendo retornada.



```
system.activate( "multitouch" )

local newRect1 = display.newRect( display.contentCenterX, display.contentCenterY, 280, 440 )
newRect1:setFillColor( 1, 0, 0.3 )

local function touchListener( event )

    print( "Fase de toque: " .. event.phase )
    print( "Localização X: " .. tostring(event.x) .. ", Localização Y: " .. tostring(event.y) )
    print( "ID de toque exclusivo: " .. tostring(event.id) )
    print( "-----" )
    return true
end

newRect1:addEventListener( "touch", touchListener )
```

# Propagação de tap/touch



Quando o usuário toca na tela, o evento é despachado para a hierarquia de exibição. Somente os objetos de exibição que cruzam o local de toque na tela receberão o evento.

Os eventos de tap e touch se propagam por meio desses objetos em uma ordem específica. O padrão é: o primeiro objeto a receber o evento é o mais a frente na hierarquia de exibição que cruza o local de toque.

Esses eventos se propagam até serem tratados, ou seja, se tivermos vários objetos sobrepostos na hierarquia de exibição e um ouvinte de evento de tap ou touch foi aplicado a cada um, o evento será propagado por todos esses objetos. Para interromper essa propagação é só informar ao Corona que o evento foi tratado.

```
local function myTouchListener( event )  
    if ( event.phase == "began" ) then  
        print( "object touched = " .. tostring(event.target) )  
    end  
    return true  
end  
  
local myButton = display.newRect( 100, 100, 200, 50 )  
myButton:addEventListener( "touch", myTouchListener )
```

# Configuração de foco de toque

Podemos direcionar todos os eventos de toque para um objeto de exibição específico definindo o foco no objeto. Isso instrui o sistema a entregar todos os eventos de toque futuros a esse objeto, tornando-o "dono" do primeiro toque que recebe durante a vida útil desse toque.

Para definir o foco em um objeto de exibição, passamos a referência dele para `StageObject:setFocus()`.

Por outro lado quando desejamos liberar o foco no objeto, basta passar `nil` para o `StageObject:setFocus()`.

```
local function touchListener( event )  
  
    if ( event.phase == "began" ) then  
        event.target.alpha = 0.5  
  
        display.getCurrentStage():setFocus( event.target )  
  
    elseif ( event.phase == "ended" or event.phase == "cancelled" ) then  
        event.target.alpha = 1  
        display.getCurrentStage():setFocus( nil )  
    end  
    return true  
end  
  
retangulo1:addEventListener( "touch", touchListener )  
retangulo2:addEventListener( "touch", touchListener )
```



# Desafio!!!!

Aplique a mesma técnica utilizada no exemplo anterior para definir o foco de toque de cada uma das imagens.

