




## Aula 2

# Correr ou deslizar?

► Unidade

Lógica de programação: jogos,  
arte e criatividade - Parte 1

# O que vamos aprender?

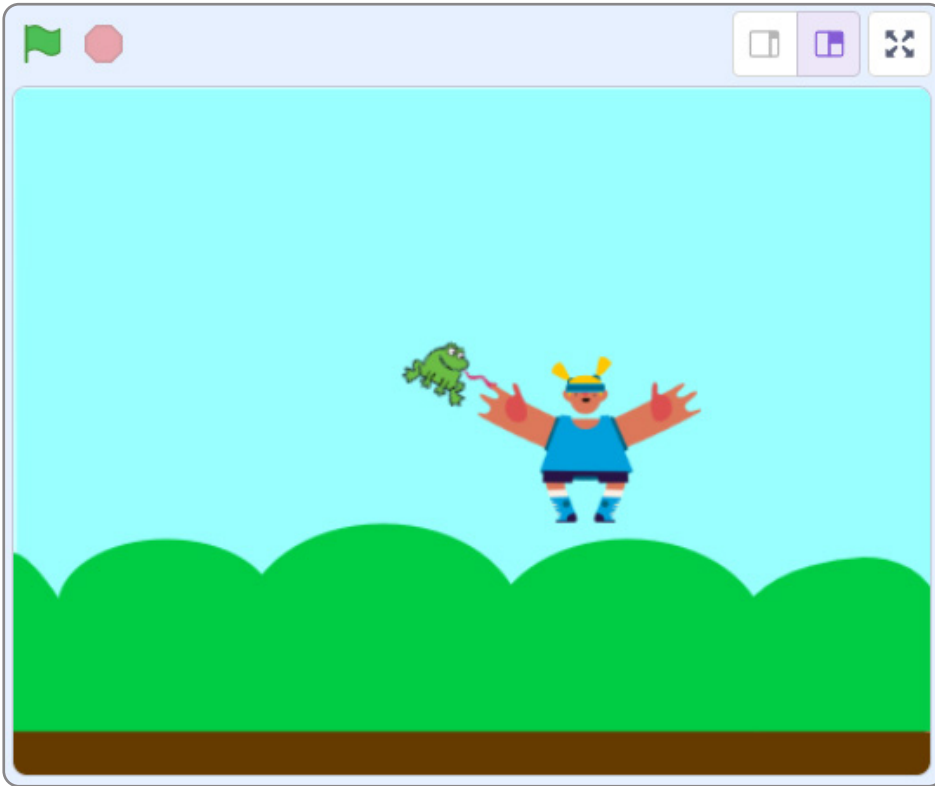
-  Compreender o uso do bloco **mova** ( ) **passos** e aplicá-lo para controlar o movimento de personagens.
-  Identificar diferenças entre os blocos de movimento.
-  Entender a importância do critério de parada em algoritmos e sua aplicação prática.



 [ACESSE A PLATAFORMA START](#)

## Se encostou, parou

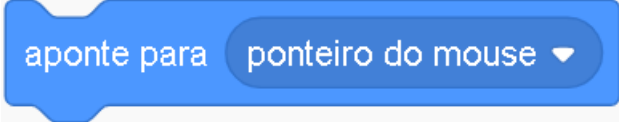

Na aula passada, criamos as personagens *Casey* e *Frog* e escolhemos um cenário divertido para o jogo de pega-pega. Nesta aula, vamos suavizar os movimentos de *Frog* para que acompanhem os de *Casey*, programar a pausa do jogo quando eles se encostarem e ajustar o tamanho das personagens para que seus movimentos fiquem mais fluidos em relação ao cenário.



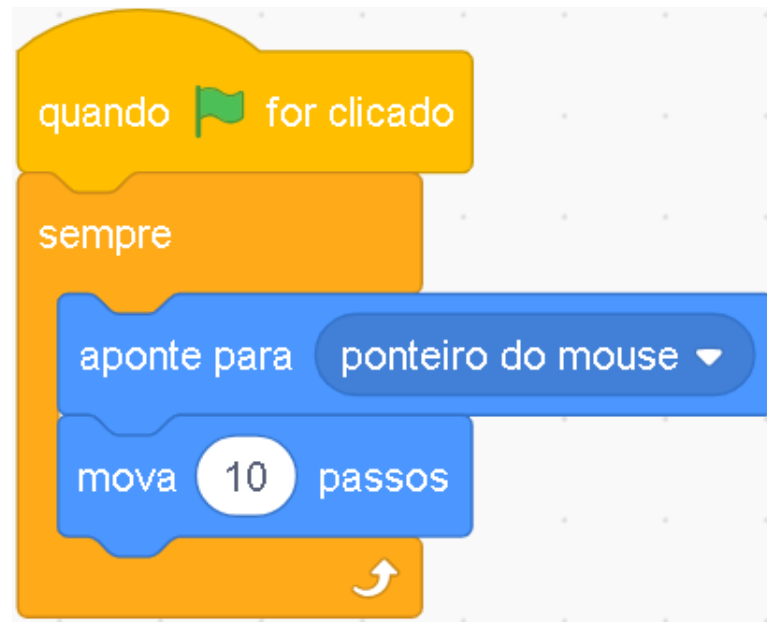
Começaremos ajustando os movimentos do *Frog* para que ele siga o ritmo de *Casey* de forma mais suave. Para isso, selecionaremos a personagem no painel de atores, localizado no canto inferior direito da tela. Com o código na área de programação, excluiremos o bloco de movimento, arrastando-o para a esquerda da tela e soltando-o para que seja excluído, como vemos na imagem a seguir:



Como nosso objetivo é que o ator *Frog* corra atrás do *ponteiro do mouse*, substituiremos o bloco de movimento que excluímos por outros dois blocos da mesma seção. Temos duas opções que realizam essa função:

o bloco  e o bloco .

Dessa forma, retornaremos à seção *Movimento* e arrastaremos esses blocos, encaixando-os um abaixo do outro, respectivamente, dentro da chave do bloco de eventos. Observe como o script:



Agora, vamos testar clicando na bandeira verde.

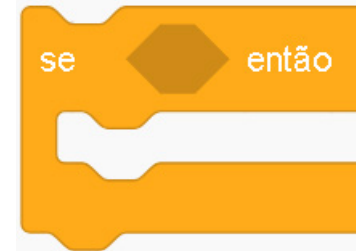
Ao testarmos, perceberemos que o ator corre atrás da personagem principal o tempo todo. E, quando as personagens se encostam, notamos que *Frog* começa a chacoalhar.



Precisamos definir um final para o jogo, de modo que, quando as personagens se encostarem, ele pare.

Para isso, precisaremos de um bloco que faça essa verificação. Assim,

da seção *Controle*, arrastaremos o bloco




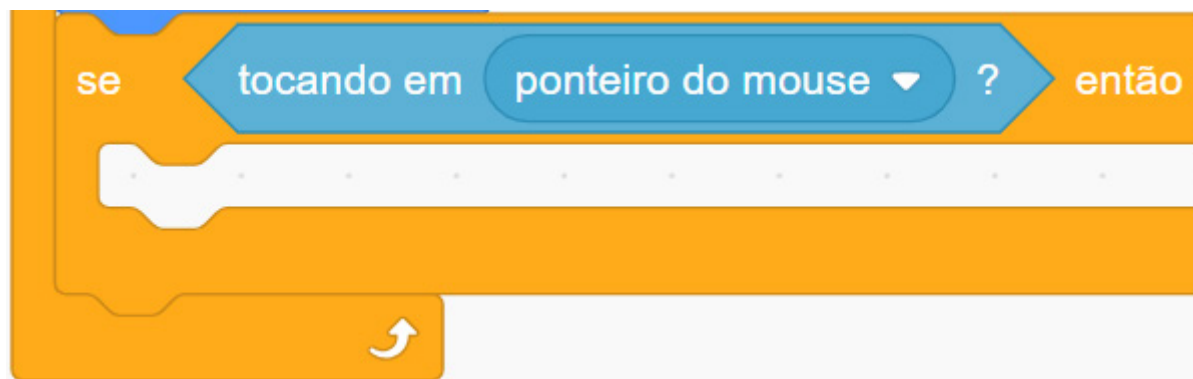
, encaixando-o


dentro da chave do primeiro bloco de controle, logo abaixo dos blocos de movimento. Observe:



Em seguida, da seção *Sensores*, arrastaremos o bloco

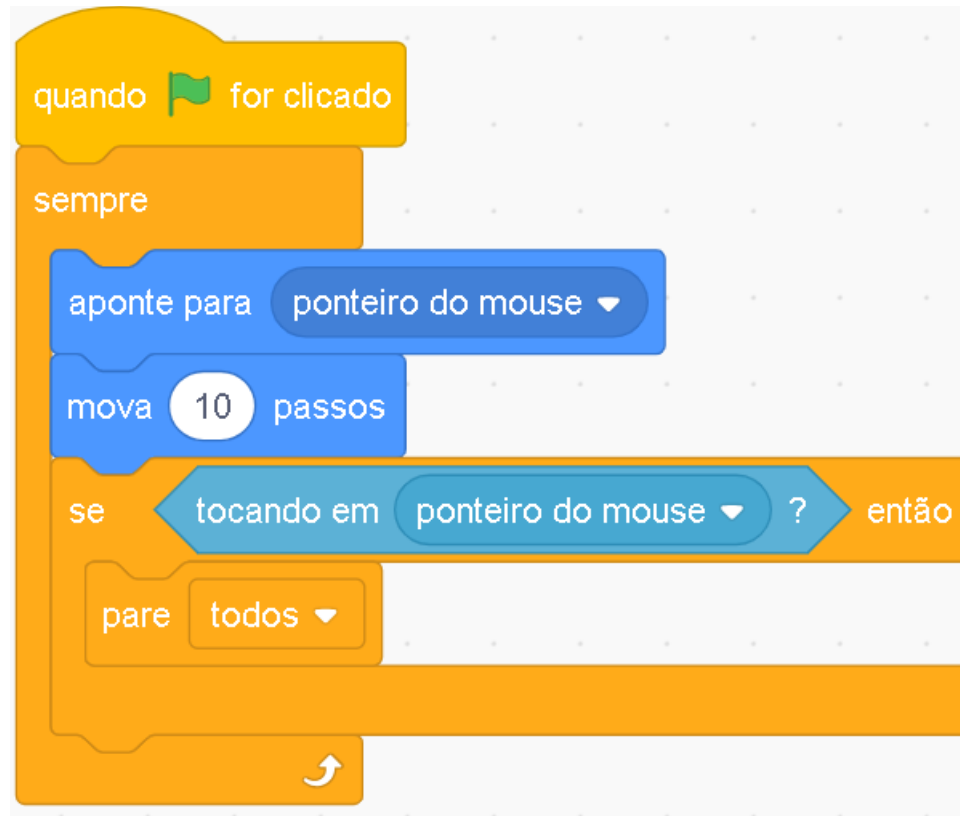
 e o encaixaremos na lacuna do segundo bloco de controle que acabamos de adicionar. Observe:



Esse conjunto de blocos que adicionamos ao código verifica se o ator *Frog* encostou ou não no *ponteiro do mouse*. Então, quando isso acontecer, o jogo precisa parar. Para isso, retornaremos à seção *Controle* e arrastaremos o bloco , encaixando-o dentro do segundo bloco de controle.

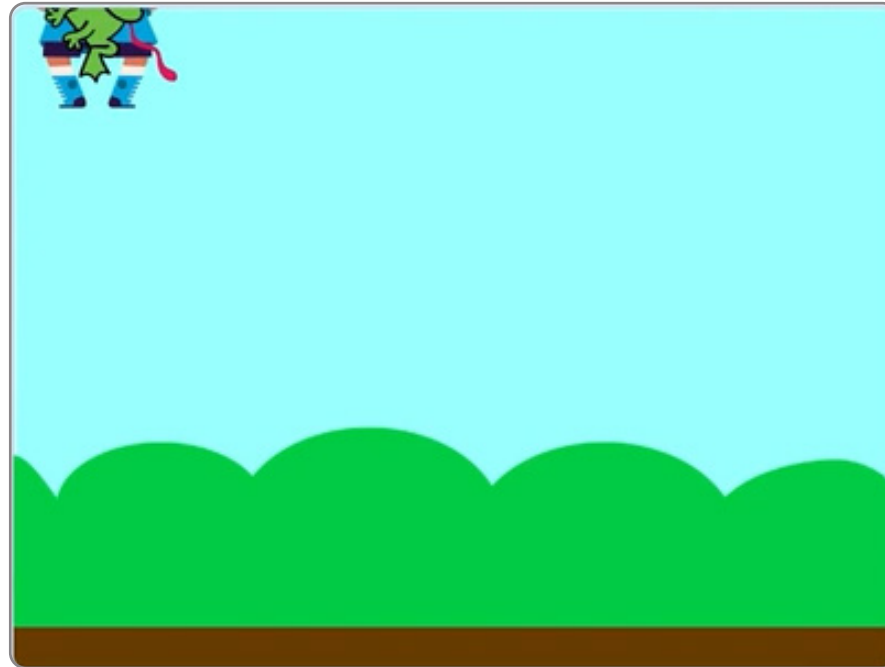



Dessa forma, informamos que, se o *Frog* estiver tocando no ponteiro do mouse, o jogo para. Feito isso, teremos o seguinte script:



Vamos testar novamente!

Ao testarmos o código, veremos que *Frog* está encurralando *Casey*. Com isso, não conseguimos movimentar os atores; precisamos de um espaço entre eles.





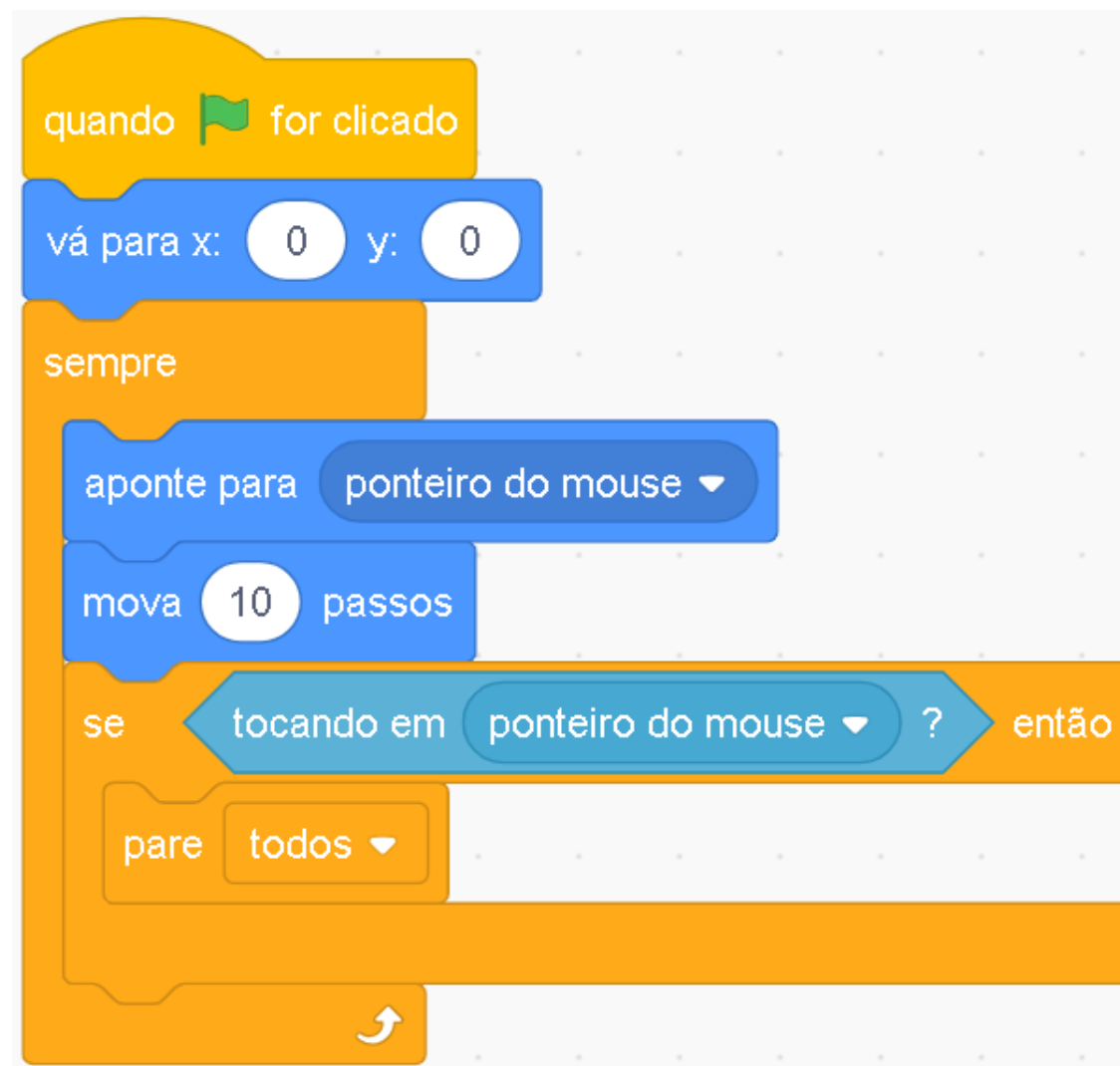
Para resolver esse bug (erro, em inglês), precisamos que o ator inicie o jogo posicionado no centro da tela. Dessa forma, ele só começará a correr atrás de Casey após o jogo começar, permitindo que a personagem principal se mova livremente.

Para isso, retornaremos à seção *Movimento* e arrastaremos o bloco

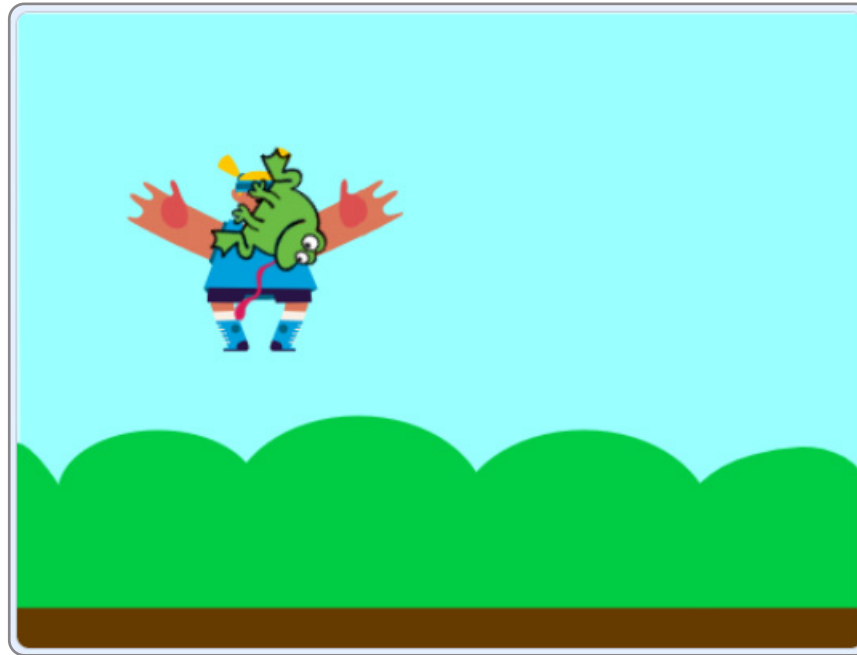
vá para x: -149 y: 106 para o início do código, logo abaixo do bloco

quando  for clicado. Após posicioná-lo, ajustaremos seus valores para x:0 e y:0, para que o ator comece o jogo centralizado na tela.

Após realizarmos essas modificações, nosso script ficará assim:




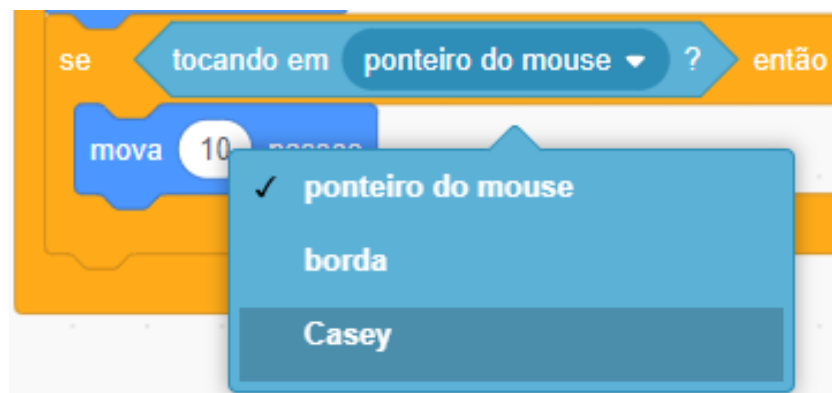
Agora, ao testarmos clicando na bandeira verde, observaremos que o ator inicia a cena no centro da tela e, em seguida, começa a correr atrás da personagem principal. Se eles se encostam, o jogo para; contudo, percebemos que *Frog* está praticamente em cima de Casey. Observe:



Para resolvermos isso, precisaremos ajustar o jogo para que ele pare assim que, *Frog* tocar as extremidades de Casey.

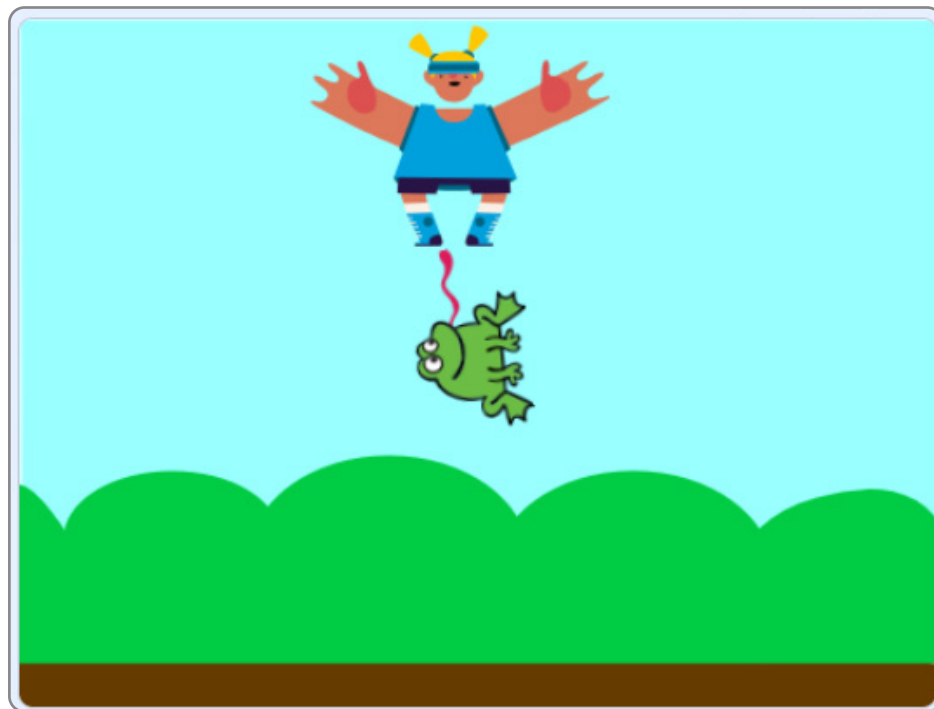
Para isso, clicaremos no menu suspenso do bloco

 e selecionaremos a opção *Casey*. Observe a imagem a seguir:



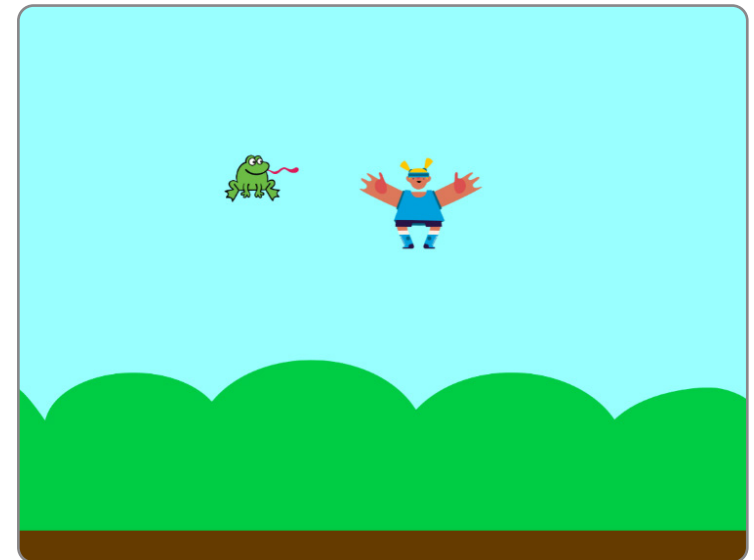
Agora, ao clicarmos na bandeira verde, observaremos que, ao iniciarmos o movimento da personagem, não conseguiremos escapar do *Frog*. Precisaremos ser mais rápidos para fugir, pois se a ponta da língua do sapo tocar nas extremidades dela, o jogo para.

Isso acontece porque o tamanho dos atores é muito grande em relação ao cenário. Observe:



Dessa forma, precisamos diminuir o tamanho das personagens para que fiquem proporcionais ao tamanho do cenário.

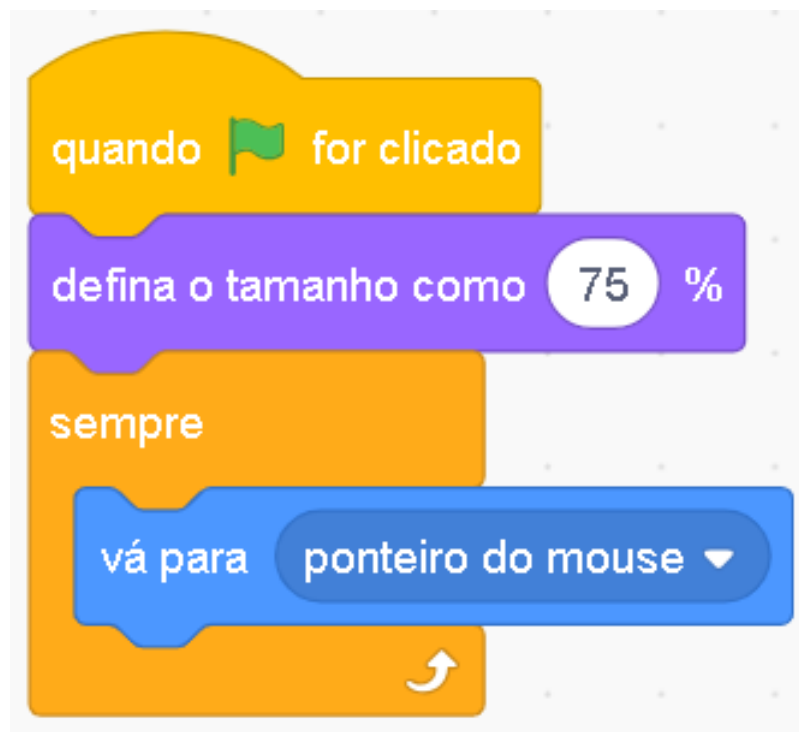
Para isso, retornaremos ao painel de atores e, na parte superior, com o ator selecionado, identificaremos o campo *Tamanho* que, por padrão, está definido como 100. Alteraremos esse valor para 50 em ambos os atores, deixando-os menores em relação ao cenário. Observe a sequência de imagens:



Também podemos configurar o código para realizar essa alteração automaticamente.



Para isso, no código da personagem Casey, adicionaremos um bloco que execute as mesmas alterações feitas manualmente, mas agora dentro do código. Na área de código, da seção *Aparência*, arrastaremos o bloco **defina o tamanho como 100 %** e o encaixaremos entre os blocos de eventos e de controle, para evitar que essa ação se repita. Após encaixá-lo, alteraremos seu valor para 75%. Observe como ficará o código:



Feito isso, ao clicarmos na bandeira verde, Casey aumentará ligeiramente de tamanho, passando de 50% para 75%. Observe:



Até o momento, nosso projeto está muito divertido. Agora, precisamos pensar em como deixar o sapo mais rápido, mas esse é um assunto para a próxima aula!

**Bons estudos!**