start by alura

Habilidade trabalhada nesta aula:

(EF08MA18) Reconhecer e construir figuras obtidas por composições de transformações geométricas (translação, reflexão e rotação), com o uso de instrumentos de desenho ou de softwares de geometria dinâmica.

Aula 2

Correr ou deslizar?

▶ Unidade

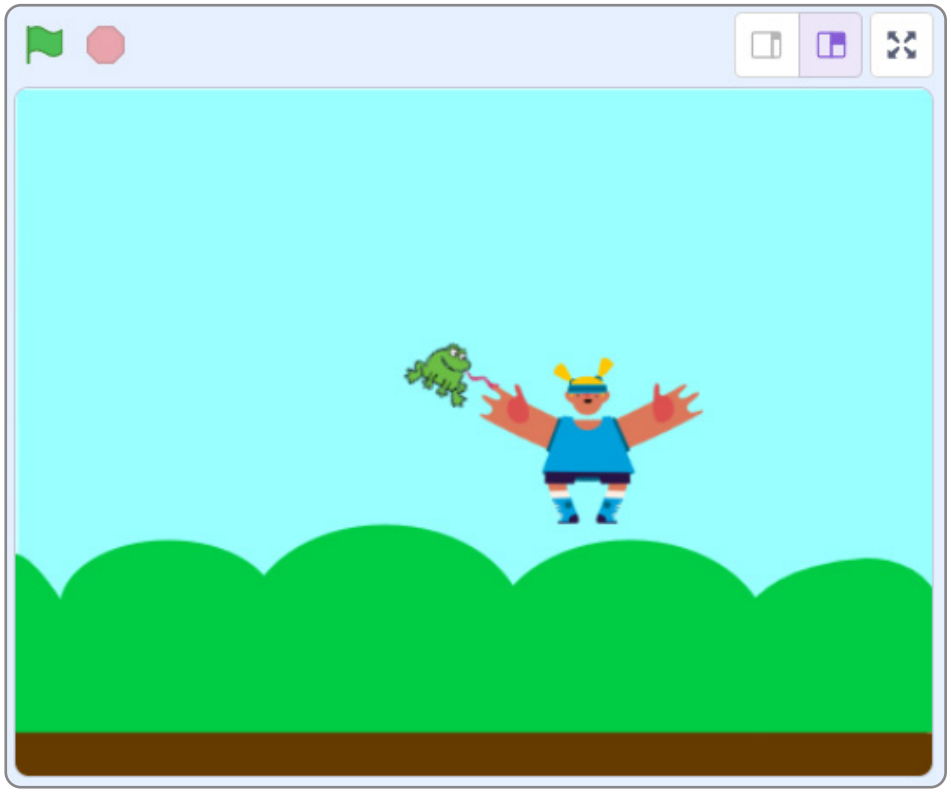
Lógica de programação: jogos, arte e criatividade - Parte 1

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

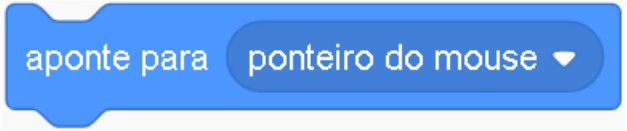


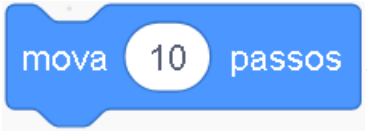
Na aula passada, criamos as personagens *Casey* e *Frog* e escolhemos um cenário divertido para o jogo de pega-pega.
Nesta aula, vamos suavizar os movimentos de *Frog* para que acompanhem os de *Casey*, programar a pausa do jogo quando eles se encostarem e ajustar o tamanho das personagens para que seus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

Para iniciar a aula, sugere-se que o professor compare as melhorias que faremos no jogo de pega-pega com uma partida de futebol. Assim como no futebol, onde todos os jogadores precisam se mover juntos e de forma coordenada para o jogo fluir bem, suavizaremos os movimentos das personagens no StartLab para que sigam o mesmo ritmo. Além disso, vamos programar o jogo para parar no momento certo, assim como o árbitro apita o final da partida. Isso envolve montar uma estratégia, usando o pensamento computacional para garantir que o jogo funcione de forma divertida e organizada.

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:

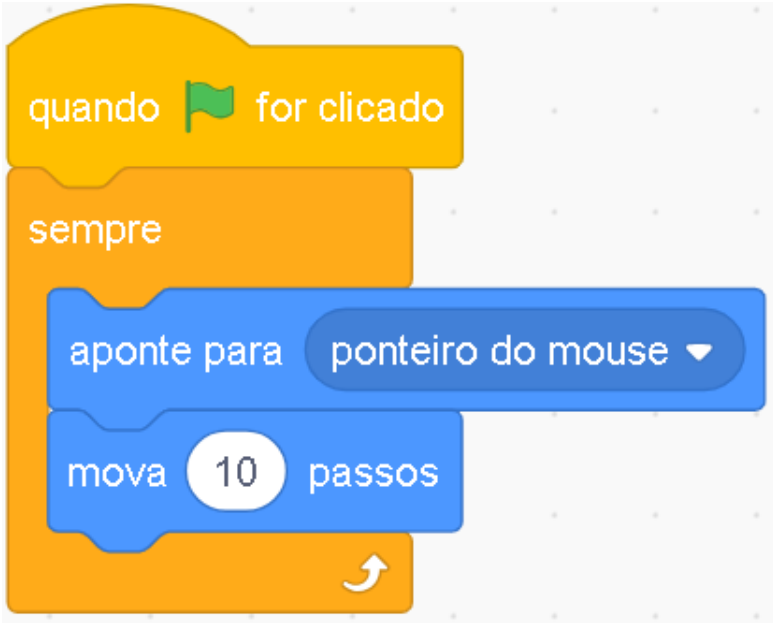
Professor, para iniciar a aula, é importante lembrar aos estudantes que, para continuarem construindo seus projetos, eles precisam acessar a seção *Meu Projeto*, disponível no menu *Aula Atual*, localizado ao lado esquerdo da tela, na plataforma Alura Start. Neste momento da aula, é importante destacar que, além de arrastar o bloco até o menu de blocos no canto esquerdo da tela para excluí-lo, eles também podem clicar com o botão direito do mouse sobre o bloco e, no menu suspenso que aparecerá, selecionar a opção *Apagar Bloco*.

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.

Neste caso, o *Frog* continua em movimento porque usamos o bloco *sempre*, o que faz que ele continue correndo atrás do *Casey* mesmo após encostar nele. Isso causa o efeito de chacoalhar. Para resolver isso, precisamos definir um final para o jogo que interrompa esse movimento contínuo. Faremos isso na área de código do ator *Frog*, programando o fim do jogo assim que ele encostar em *Casey*, mesmo sem mexermos no cursor.

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

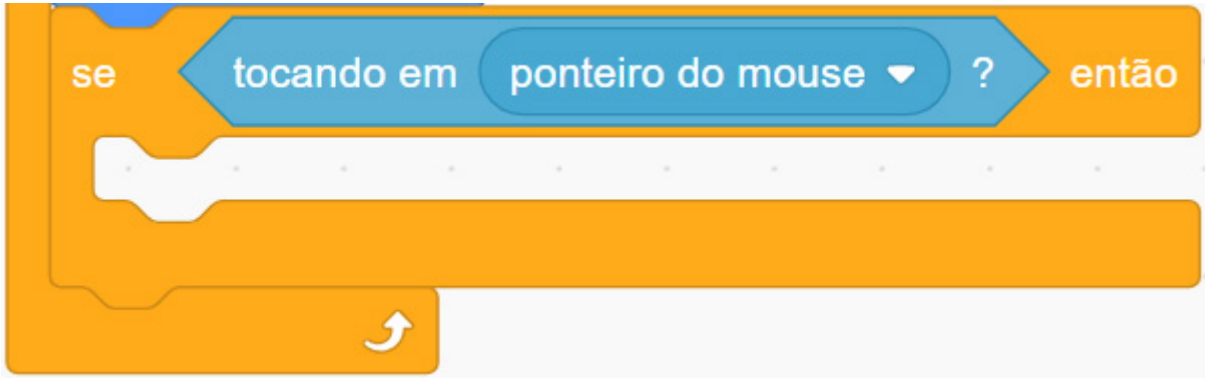
da seção *Controle*, arrastaremos o bloco 

, 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:



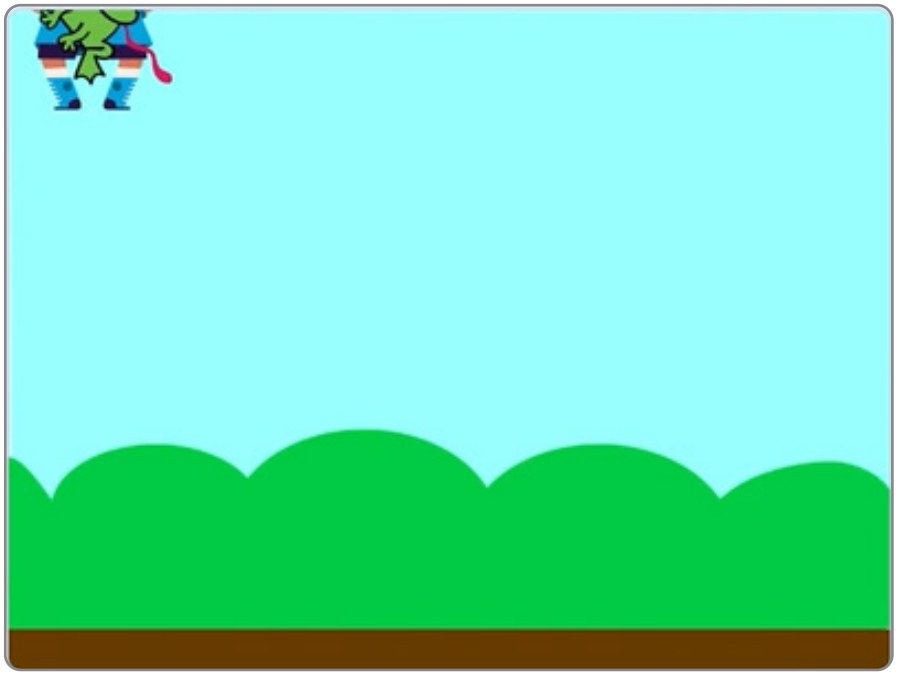
Esse conjunto de blocos que adicionamos ao código verifica se o ator *Frog* encostou ou não no *ponteiro do mouse*. Então, quando isso acontecer, o jogo precisa parar. Para isso, retornaremos à seção *Controle* e arrastaremos o bloco 

, 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:

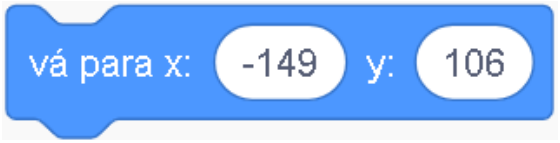
Vamos testar novamente!

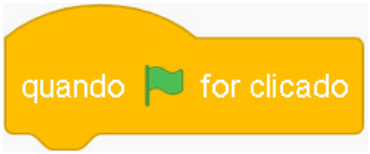
Ao testarmos o código, veremos que *Frog* está encurralando *Casey*. Com isso, não conseguimos movimentar os atores; precisamos de um espaço entre eles.



O *Frog* e o *Casey* não conseguem se movimentar corretamente porque o código faz que as personagens se movam rapidamente e parem abruptamente ao tocar o ponteiro do mouse. Isso resulta em movimentos bruscos e em um *chacoalhar* quando as personagens colidem. Para corrigir, será preciso ajustar o movimento para que seja mais suave e modificar a condição de parada para não interromper todos os scripts.

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.

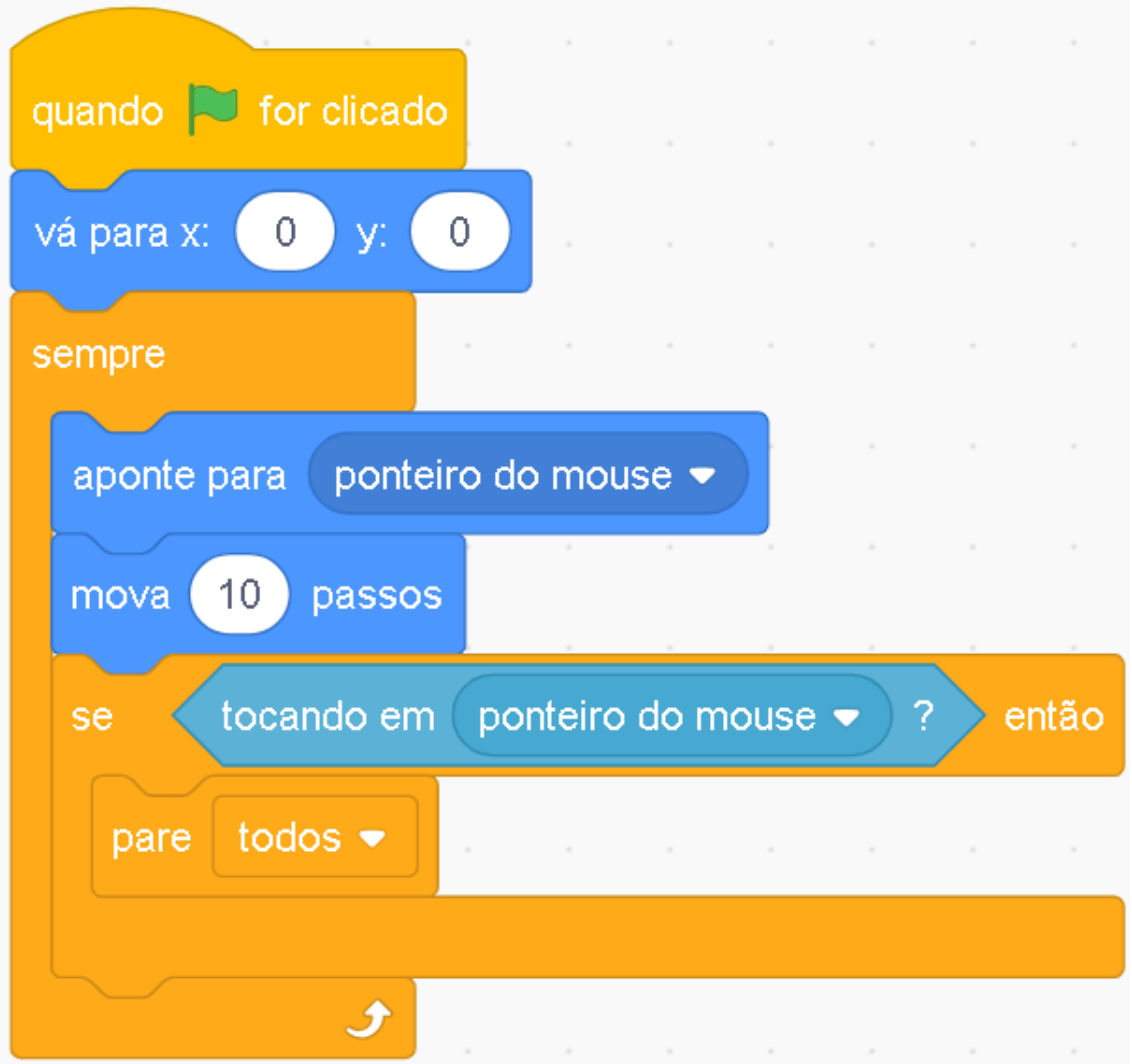
Para isso, retornaremos à seção *Movimento* e arrastaremos o bloco 

para o início do código, logo abaixo do bloco 

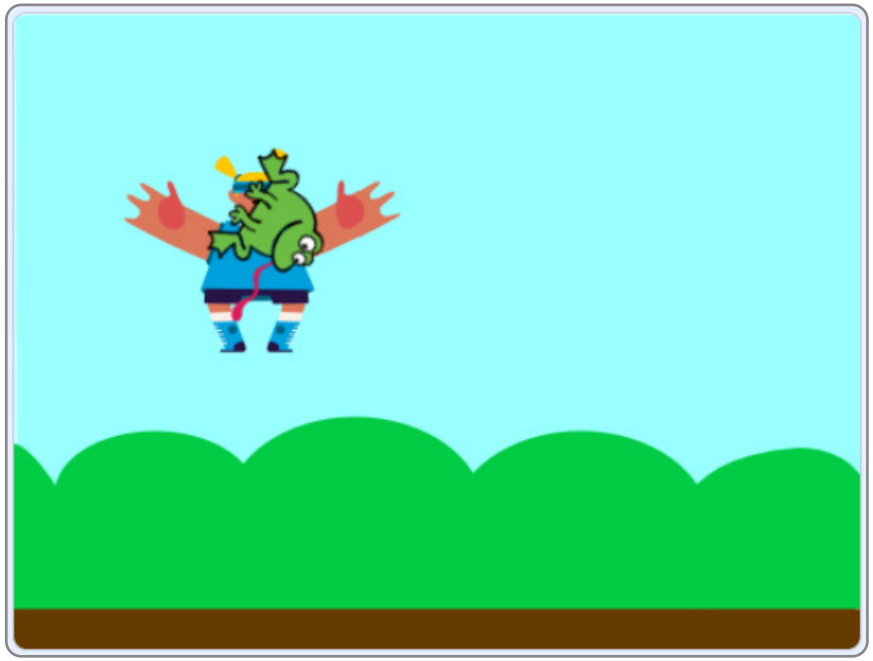
. Após posicioná-lo, ajustaremos seus valores para x:0 e y:0, para que o ator comece o jogo centralizado na tela.

O bloco de movimento mencionado define a posição inicial da personagem no cenário. Ao usar 0 para ambos os valores x e y, estamos colocando a personagem exatamente no centro do cenário.

Após realizarmos essas modificações, nosso script ficará assim:



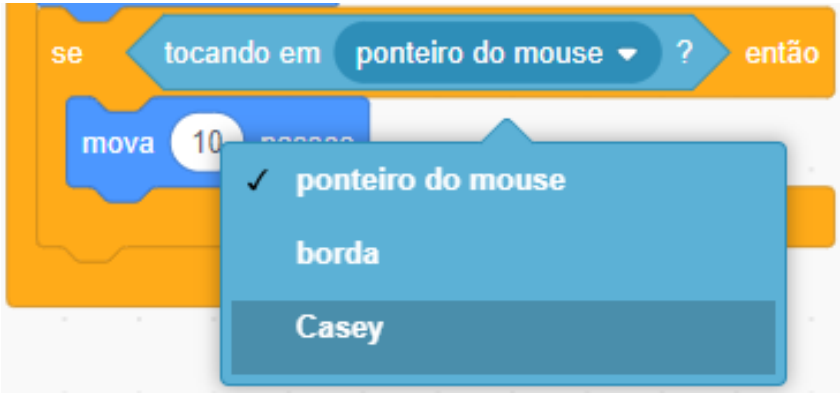
Agora, ao testarmos clicando na bandeira verde, observaremos que o ator inicia a cena no centro da tela e, em seguida, começa a correr atrás da personagem principal. Se eles se encostam, o jogo para; contudo, percebemos que *Frog* está praticamente em cima de *Casey*. Observe:



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

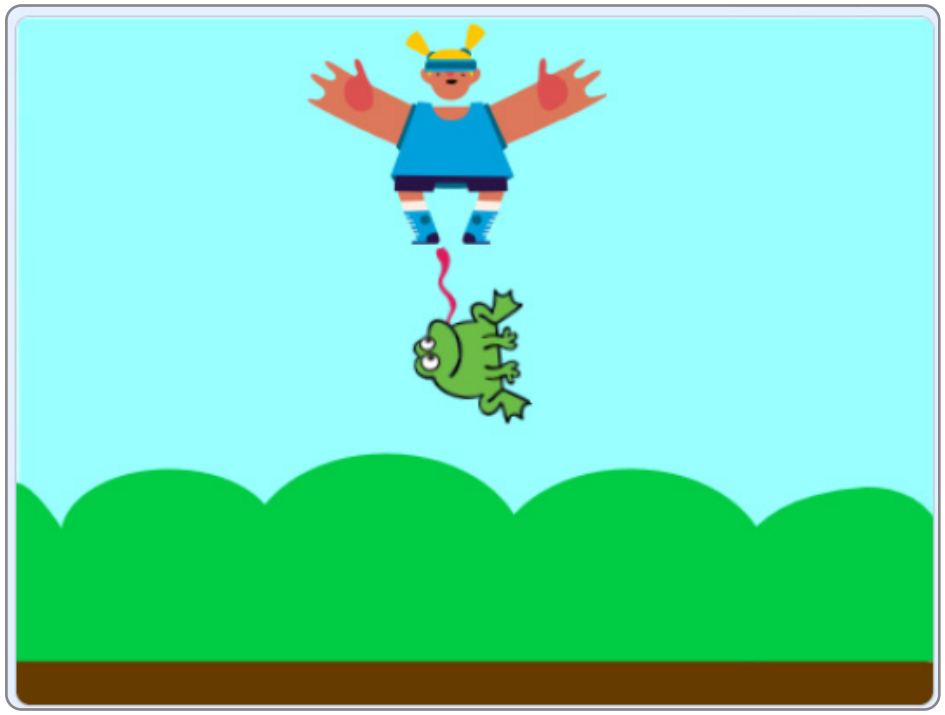
Para isso, clicaremos no menu suspenso do bloco

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



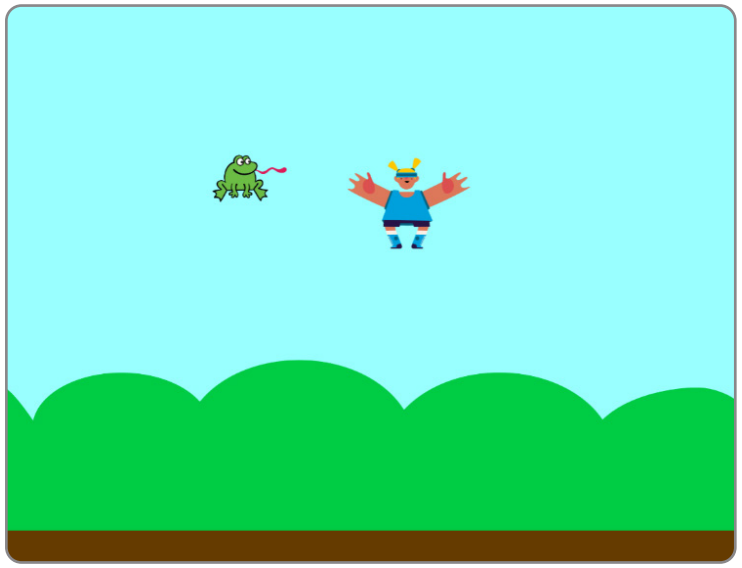
Agora, ao clicarmos na bandeira verde, observaremos que, ao iniciarmos o movimento da personagem, não conseguiremos escapar do *Frog*. Precisaremos ser mais rápidos para fugir, pois se a ponta da língua do sapo tocar nas extremidades dela, o jogo para.

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

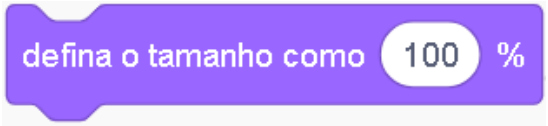


Dessa forma, precisamos diminuir o tamanho das personagens para que fiquem proporcionais ao tamanho do cenário.

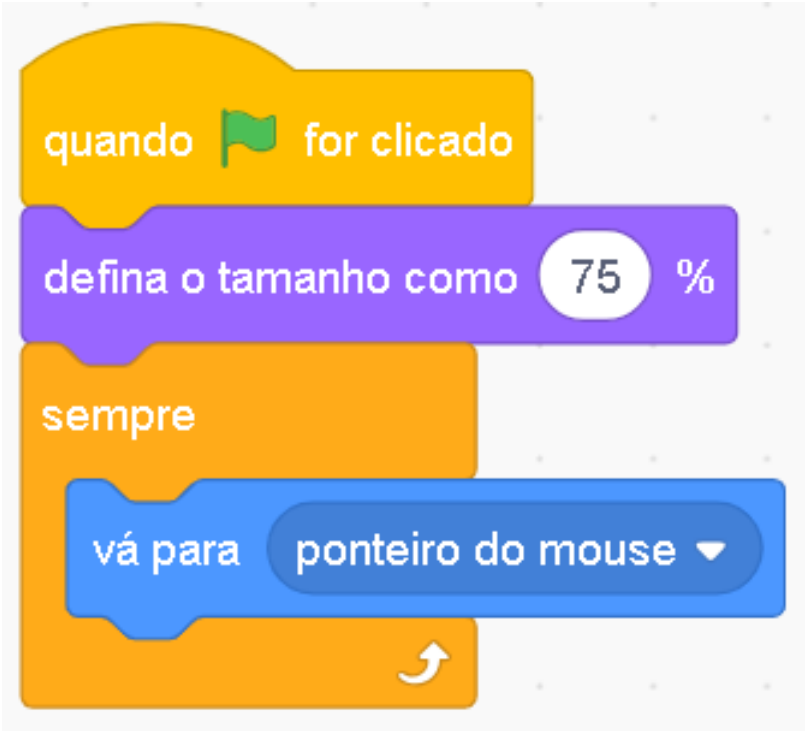
Para isso, retornaremos ao painel de atores e, na parte superior, com o ator selecionado, identificaremos o campo *Tamanho* que, por padrão, está definido como 100. Alteraremos esse valor para 50 em ambos os atores, deixando-os menores em relação ao cenário. Observe a sequência de imagens:



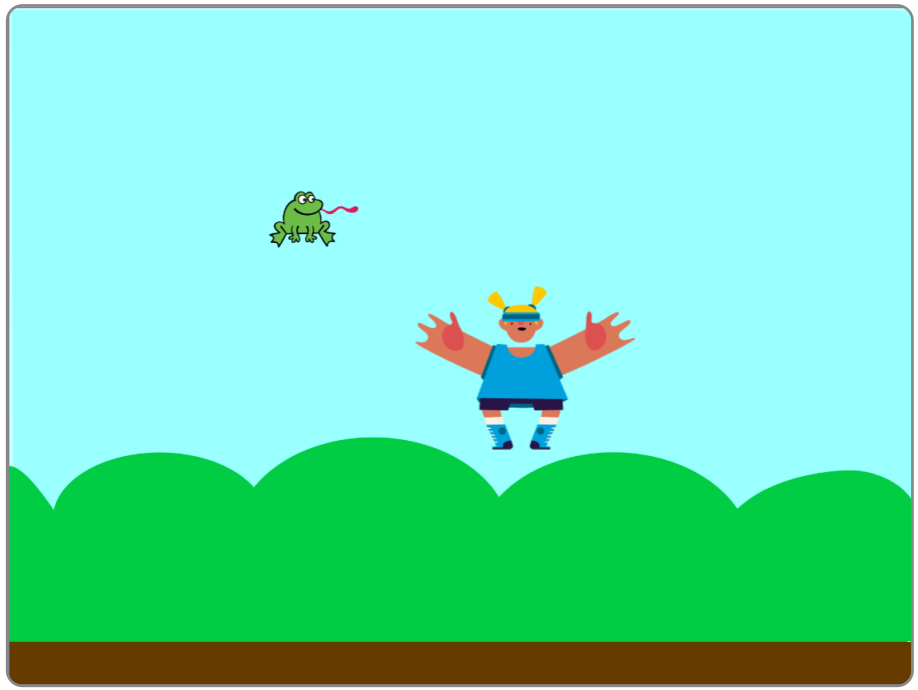
Também podemos configurar o código para realizar essa alteração automaticamente.

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 

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!

start by alura

CLIQUE AQUI PARA AVALIAR ESTE MATERIAL