

**Habilidades trabalhadas nesta aula:**

(EF07MA05) Resolver um mesmo problema utilizando diferentes algoritmos.

(EF07CO03) Construir soluções computacionais de problemas de diferentes áreas do conhecimento, de forma individual e colaborativa, selecionando as estruturas de dados e técnicas adequadas, aperfeiçoando e articulando saberes escolares.

(EF07CO11) Criar, documentar e publicar, de forma individual ou colaborativa, produtos (vídeos, podcasts, web sites) usando recursos de tecnologia. (basicamente a publicação do jogo).




Aula 4

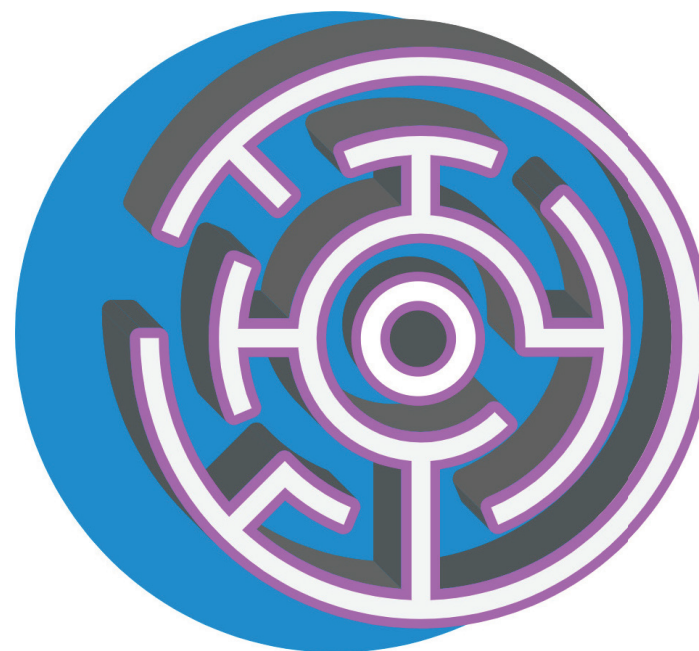
Usando condições

► **Unidade**

**Lógica de programação:
desenvolvendo a missão
labirinto**

O que vamos aprender?

-  Utilizar o conceito de condicional para definir situações.
-  Aumentar a possibilidade de opções dentro de um jogo.
-  Definir pontos de mudança e retomada no jogo, aumentando sua dinâmica.

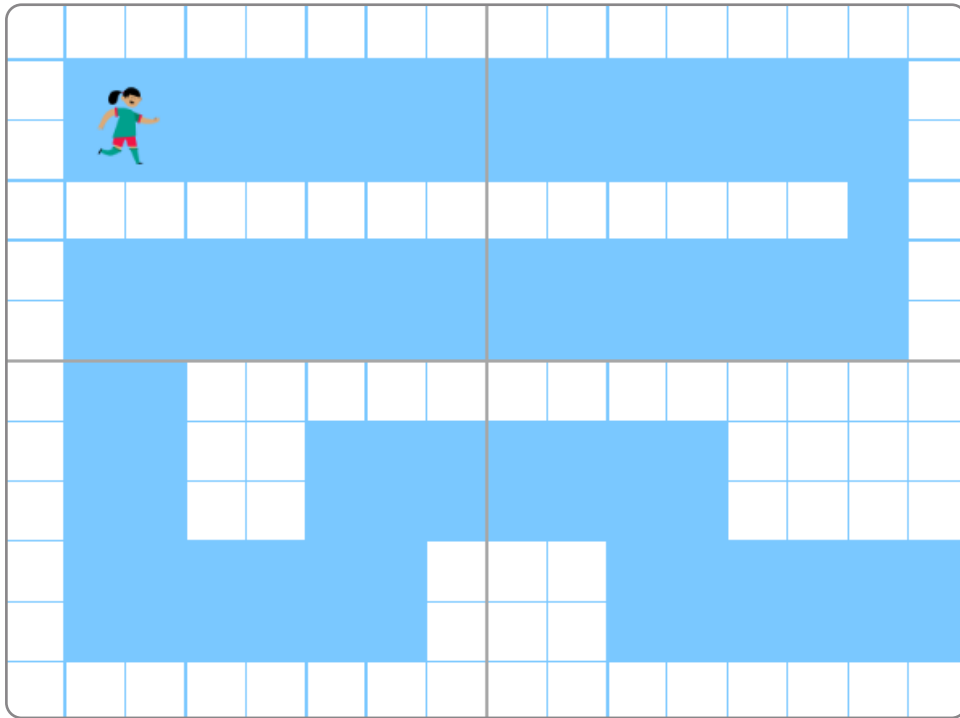


▶ ACOMPANHE O VÍDEO DA AULA



Usando condições para movimentos

Anteriormente, programamos a movimentação da nossa personagem para cima, para baixo e para os lados. Portanto, ela pode, até o momento, movimentar-se por todo o *Palco* sem respeitar os limites do labirinto. Nesta aula, adicionaremos mais alguns blocos aos scripts do projeto, possibilitando a criação de desafios para o jogador durante a movimentação da personagem pelo *Palco*.





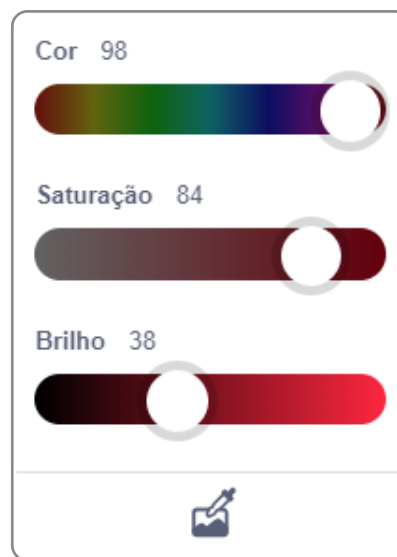
! Professor, para iniciar a aula, peça para que os estudantes apresentem brevemente as ideias que tiveram sobre novos labirintos. Oriente-os a acessarem a plataforma e utilizarem o mesmo projeto criado. Nesta aula, definiremos o espaço de movimento da personagem no *Palco*, para que o jogo tenha sentido.

Assim, começaremos a modificação dos scripts adicionando os blocos condicionais da categoria *Controle*. Essa categoria possui blocos que apresentam diferentes tipos de controle, como tempo, fantasia, situações ou resultados, repetição e criação de clones.


! Professor, converse com os estudantes sobre o conceito de condicional. As condicionais, em programação e lógica, são estruturas de controle que permitem que um programa execute diferentes ações com base em uma condição específica. Essa condição é avaliada como verdadeira ou falsa, e o programa toma decisões com base nessa avaliação. Assim, se queremos que a personagem se movimente apenas pelo caminho do Labirinto, utilizaremos os blocos de condicional para definir ações diferentes para quando ela estiver ou não sobre o Labirinto.

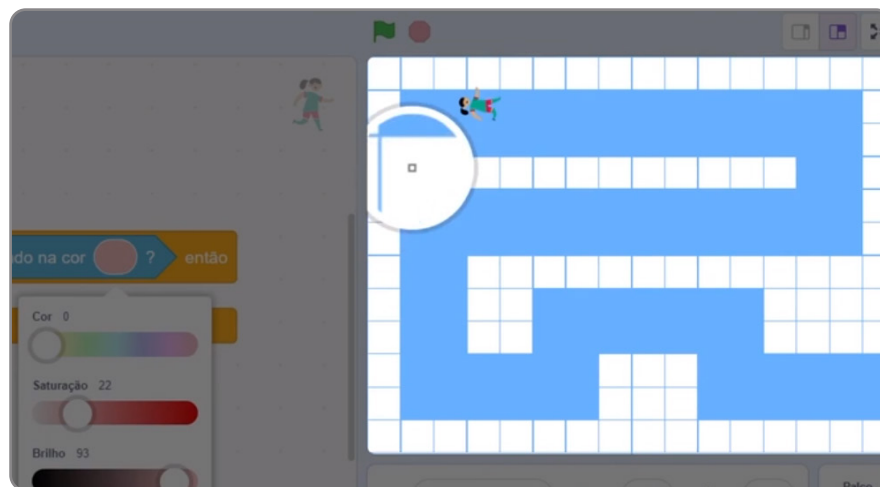


Desse modo, da categoria *Controle*, arraste o bloco  para a área de código. Agora, para detectar se a personagem está tocando a parte branca do cenário, ou seja, se ela está fora do labirinto, precisamos adicionar um bloco de sensor. Da categoria *Sensores*, arraste o bloco , encaixando-o na lacuna de mesmo formato do bloco de controle. Em seguida, clique na cor desse bloco. Teremos o seguinte menu:



! Professor, como estamos trabalhando com blocos que envolvem condicionais, realize a leitura do script com os estudantes, explicando o que está sendo definido pelos blocos. Queremos que, se a personagem tocar na cor branca, a plataforma entenda que ela está fora do Labirinto.

A condição que queremos criar é: se a personagem tocar na parte branca do *Palco*, algo deve acontecer. Essa condição definirá uma mudança no jogo. Para definirmos a cor branca do *Palco* como condicional para mudança, selecione o ícone , no menu de cores do bloco de sensor. Apenas o *Palco* do Scratch ficará iluminado, permitindo que possamos selecionar a cor desejada para o bloco sensor diretamente do cenário, como mostra a imagem a seguir:

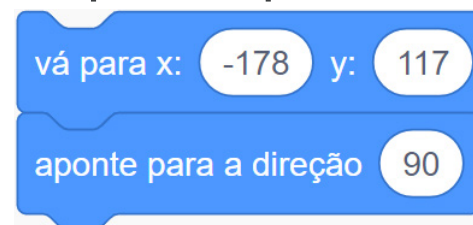


Observe que o ícone do mouse ficará no modo “lupa”. Agora, basta clicar sobre a parte branca do Palco para definir a cor branca. Feito isso, o bloco de sensor ficará da seguinte forma:

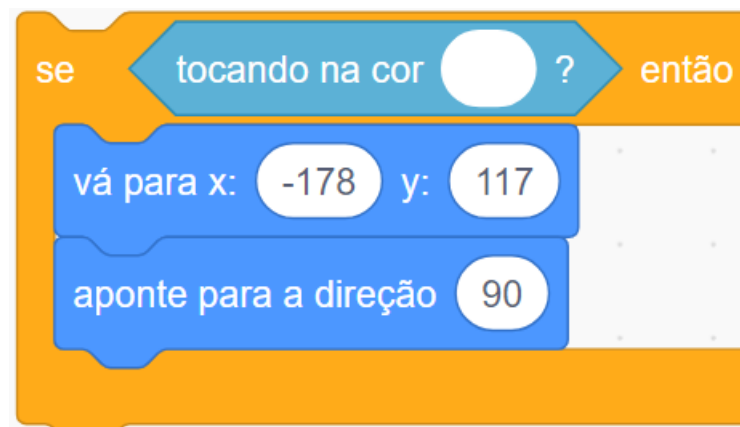



Definido o sensor, precisamos descrever o que deve acontecer quando a personagem tocar a cor branca no cenário. Neste caso, definiremos que, ao tocar a cor branca, a personagem retorne à posição inicial do jogo, obrigando o jogador a recomeçar, e aponte para a direção 90

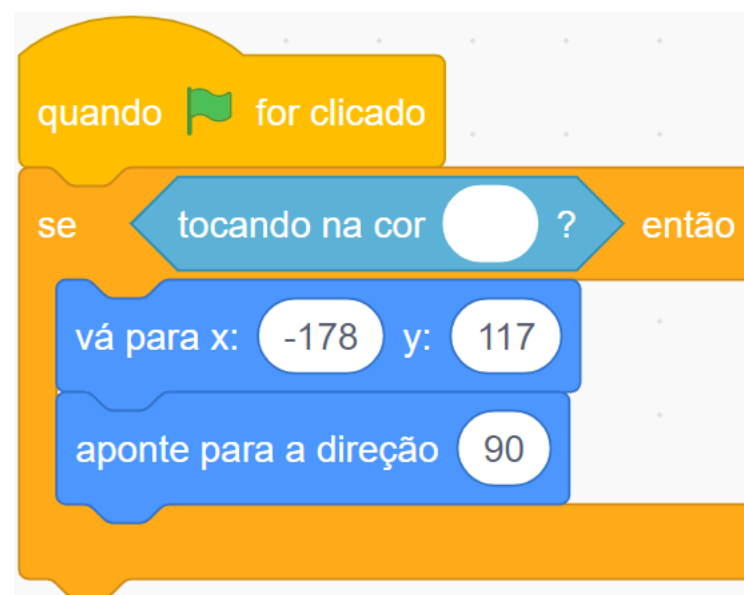
graus. Para isso, vamos copiar os blocos



no script que inicia o jogo. Para isso, clique sobre o primeiro bloco com o botão direito do mouse, selecionando o ícone **Duplicar**. Uma cópia desses blocos aparecerá na área de código. Arraste esse conjunto de blocos para dentro da chave do bloco condicional. Seu conjunto de blocos ficará da seguinte forma:



Por fim, para que essa ordenação ocorra sempre, da categoria *Eventos*, arraste o bloco **quando  for clicado** e coloque-o acima do bloco condicional. Feito isso, teremos o seguinte script:



Faça o teste do seu jogo! Observe que ainda há falhas na identificação da cor que a personagem toca, visto que ela não retorna de imediato à posição inicial. Mas esse problema será resolvido na próxima aula.

! Professor, continue realizando a leitura do script com os estudantes, para que a ordenação tenha um sentido lógico. Realize o teste do script com eles e peça para que digam qual bloco precisa ser adicionado. Incentive-os a observarem outras categorias e blocos presentes na plataforma, identificando novas possibilidades de programação.

► Desafio

Nesta aula, ordenamos o script que indicará uma condicional: quando a personagem encostar na cor branca do *Palco*, voltará para o ponto de início do labirinto. O que está faltando no script para termos um jogo mais competitivo e dinâmico?

Como desafio, pense em quais outras barreiras podem ser acrescentadas ao jogo, aumentando sua diversão e competitividade.

💡 Espera-se que os estudantes consigam prever barreiras que possam ser adicionadas ao jogo e testar novos blocos livremente para tentar criar essas dificuldades. Indicamos que os estudantes compartilhem suas experiências de jogos, possibilitando, assim, que utilizem seus conhecimentos prévios e gostos particulares para o desenvolvimento do Pensamento Computacional em sala de aula.



CLIQUE AQUI PARA AVALIAR ESTE MATERIAL