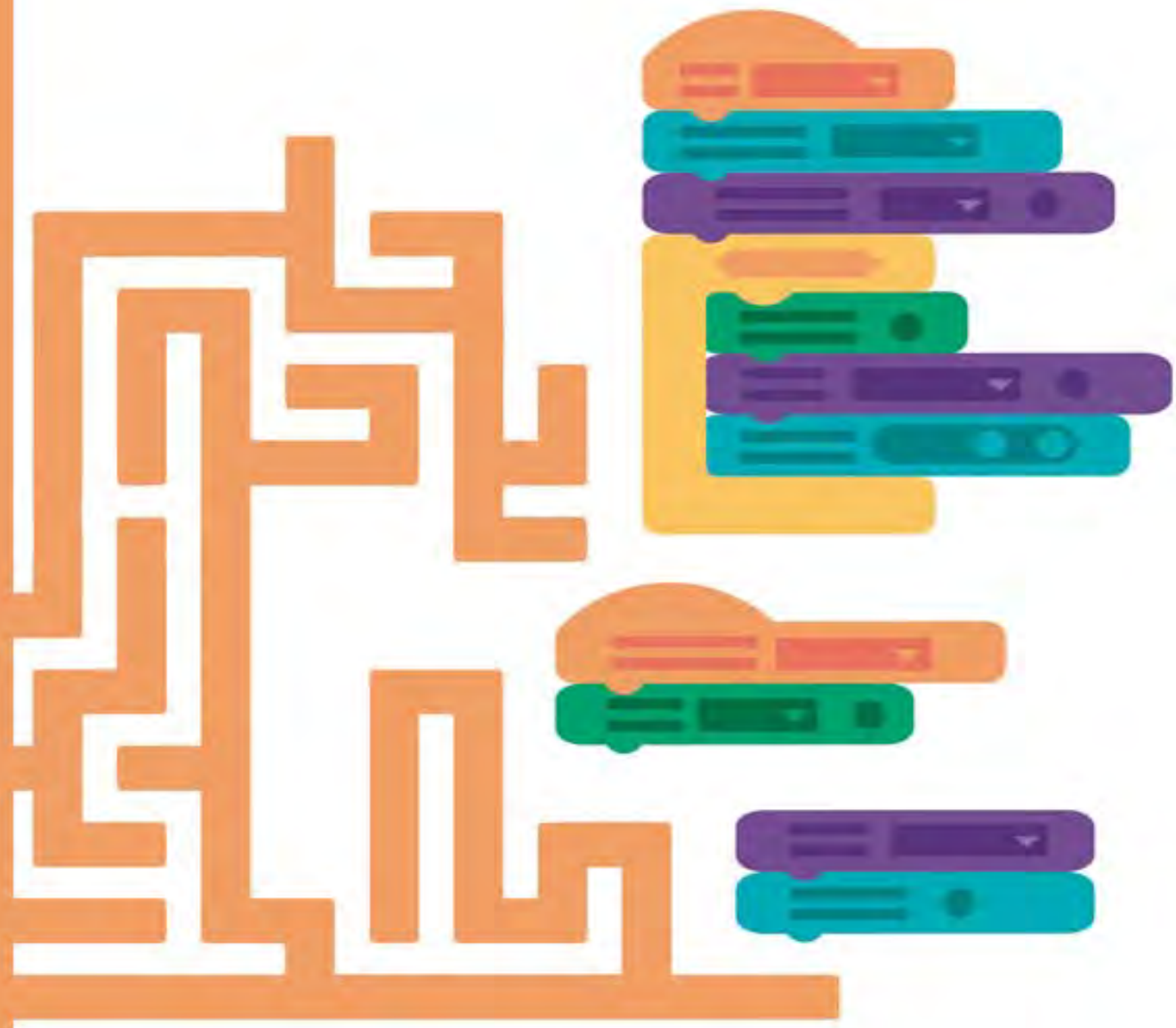
Scratch

Um jeito divertido de aprender programação



Casa do Código

HELTON VARELA

Table of Contents

Sumário

ISBN

Impresso e PDF: 978-85-5519-279-1

EPUB: 978-85-5519-280-7

MOBI: 978-85-5519-281-4

Você pode discutir sobre este livro no Fórum da Casa do Código: http://forum.casadocodigo.com.br/.

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

Prefácio

Ao desenvolver um projeto de extensão em uma escola pública sobre o ensino de programação, tivemos muitas dificuldades na coleta de materiais para prepararmos nossas aulas e nos deparamos com a falta de bibliografia sobre esse assunto. Com o surgimento de centenas de pesquisas desenvolvidas e aplicadas em todo o Brasil, sentimos a necessidade de produzir um material introdutório que pudesse ser utilizado.

O Scratch é uma ferramenta interessante para aqueles que desejam aprender a programar e para aqueles que desejam ensinar. Além de estudantes, educadores também podem usá-la no planejamento de suas aulas para complementar o processo de ensino e aprendizagem de forma criativa, ajudando os estudantes a raciocinar sistematicamente e a trabalhar colaborativamente.

No Brasil, é utilizado tanto na Educação Básica como nas universidades. O objetivo na elaboração deste material foi proporcionar aos estudantes, educadores, desenvolvedores e aos curiosos por programação a criação de um jogo do início ao fim. Nossa proposta foi criar um jogo de labirinto com todos os passos, de maneira simples e didática. Além de despertar o interesse pela área de computação e do pensamento computacional.

Após ler este livro, esperamos que você seja capaz de desenvolver outros jogos, produzir histórias e animações conforme sua vontade, necessidade e conteúdo. A programação aprimora o raciocínio lógico, a criatividade e a resolução de problemas, habilidades importantes para os cidadãos do século XXI.

Organizadora: Claudia T. Peviani **Público-alvo** Este livro foi escrito para aqueles que estão começando tanto na vida quanto na computação. O público-alvo deste livro é para aqueles entre 7 e 16 anos, mas qualquer um que queira aprender vai se dar muito bem.

O Scratch é uma ótima ferramenta de aprendizado e qualquer um que estiver disposto a ler este livro aprenderá o que é programação. Não há pré-requisito algum. O único pré-requisito é a vontade de aprender, tendo isso você já está com meio caminho andado.

Agradecimentos

Dedico este livro a todos aqueles que buscam novos conhecimentos e estão sempre dispostos a aprender.

Agradeço a Deus, por dar força e energia necessárias para vencer os obstáculos da vida.

Sou grato à professora Claudia pela oportunidade no projeto, e à UFGD. Aos meus pais, Sueli e Manoel, por sempre me apoiarem, e aos amigos, pelos bons momentos.

CAPÍTULO 1

O que é o Scratch?

O Scratch é um programa de computador no qual é possível escrever uma série de instruções que dirá ao computador o que deve ser feito. Com ele, podemos criar nossas próprias histórias, jogos ou animações, e tudo isso é feito através de uma linguagem de programação. Foi desenvolvido pelo Media Lab do MIT em 2007, e tem como principais características a facilidade e acessibilidade, permitindo que pessoas no mundo todo comecem seus estudos em programação.

Como o Scratch utiliza uma interface gráfica e blocos que são montados como Lego, é muito mais fácil aprender programação através dele. Muitas escolas em países desenvolvidos já utilizam o Scratch como parte da grade de ensino, mostrando para crianças a ciência da computação logo cedo.

Iniciativas como o *Hour of Code* têm levado o ensino de programação para crianças no mundo todo, e utilizando muitas vezes o Scratch. Alguns cursos introdutórios de Ciência da Computação têm usado o Scratch também como uma ferramenta para introduzir a programação.

Há uma comunidade online que se ajuda para desenvolver alguns projetos e compartilha. Veremos no final deste capítulo alguns projetos disponíveis online e os seus códigos. O Scratch permite visualizar o código de qualquer projeto, não impedindo o usuário de conhecer como funciona qualquer programa. O slogan da comunidade é: *Imagine, Programe, Compartilhe*, o que significa que a imaginação e o compartilhar são conceitos fundamentais para todo desenvolvedor Scratch.

O Scratch é um software de código aberto, o que significa que você pode baixar, examinar e alterar o código conforme queira. A maioria das linguagens de programação são baseadas em texto, isto é, você teria de dar comandos ao computador, escrevendo-os em um programa. O Scratch é uma linguagem de programação visual, portanto, nós apenas precisamos arrastar blocos para dentro da área de programação.

A programação é algo bastante divertido e tenho certeza de que você, caro leitor, que está começando agora no mundo da programação vai gostar do conteúdo apresentado através deste livro. Aqui vamos abordar todos os conceitos básicos em programação e estaremos desenvolvendo um jogo no qual será aplicado o conhecimento adquirido de cada capítulo. Para começar, vamos entender qual a diferença do Scratch para as outras linguagens de programação.

Veja este exemplo do clássico *Hello World!* Os trechos de códigos apresentados a seguir exibem uma simples mensagem na tela do computador. Tente entender o que cada trecho de código está "dizendo" ao computador.

Linguagem C

printf(“Hello World!”);

Linguagem Python print "Hello World!"

Linguagem Java System.out.print(“Hello World!”);

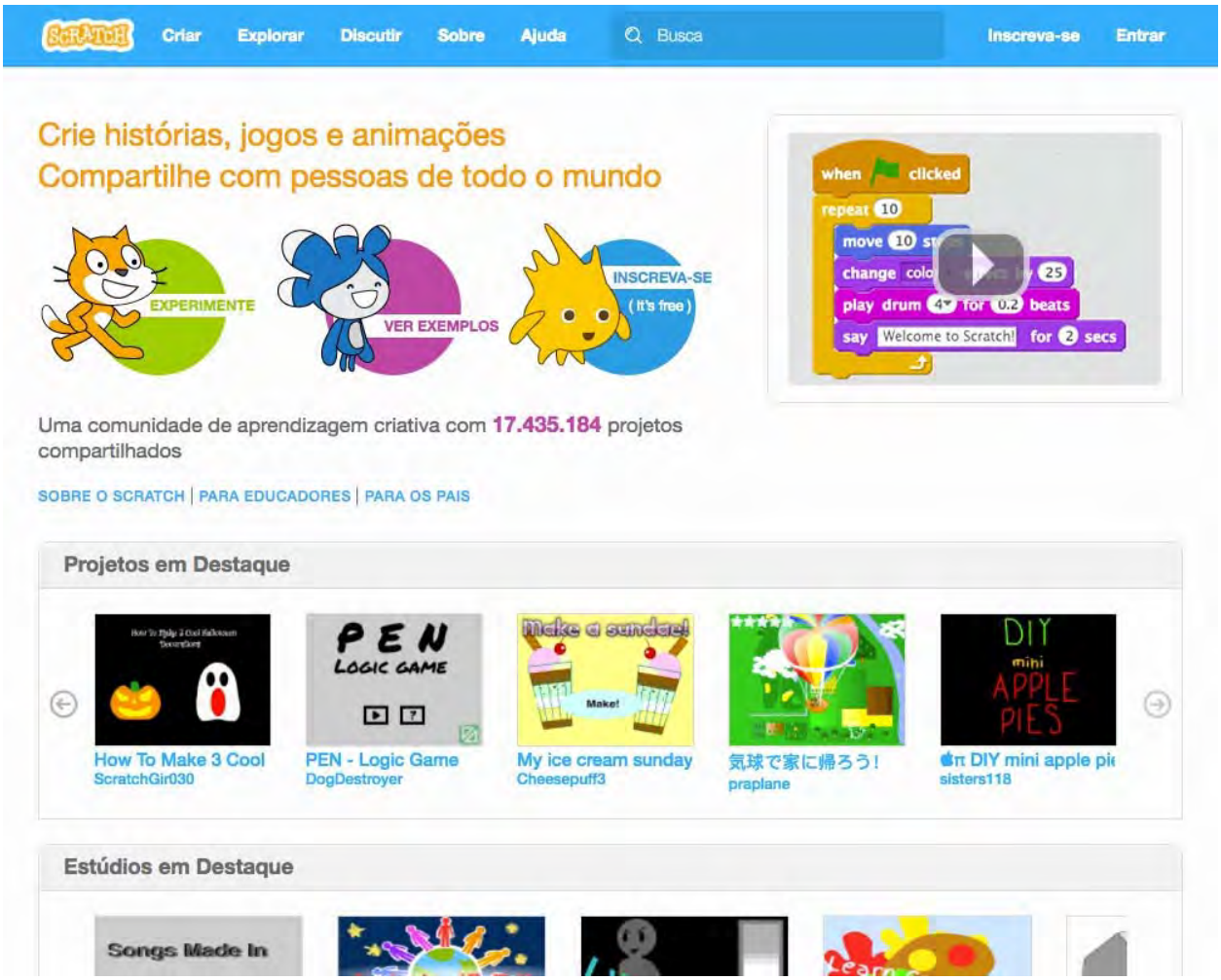
Figura 1.1: Linguagem Scratch

Veja que, em todas as linguagens de programação, o resultado será o mesmo: **Hello World!** Porém no Scratch será um pouco diferente, pois o resultado é sempre apresentado por um Ator. A mensagem é a mesma, porém a forma de apresentação é diferente:

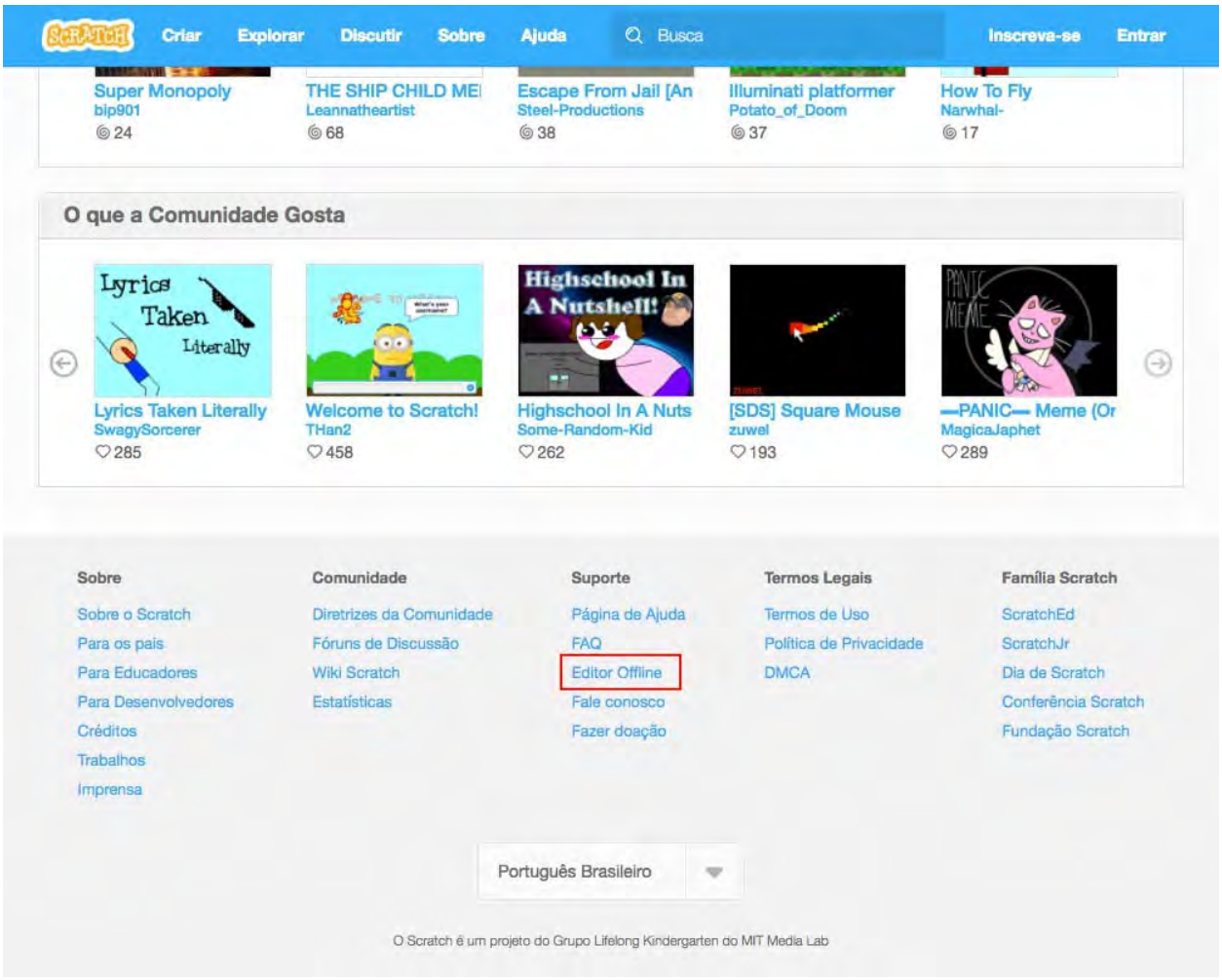
1.1 Instalação

Antes de começarmos a programar, precisamos fazer a instalação da ferramenta. Você pode também usar a versão online através do endereço: http://scratch.mit.edu. Recomenda-se a instalação, pois assim não há dependência de internet, o que permite estudar com mais tranquilidade.

1. Acesse http://scratch.mit.edu para fazer o download.

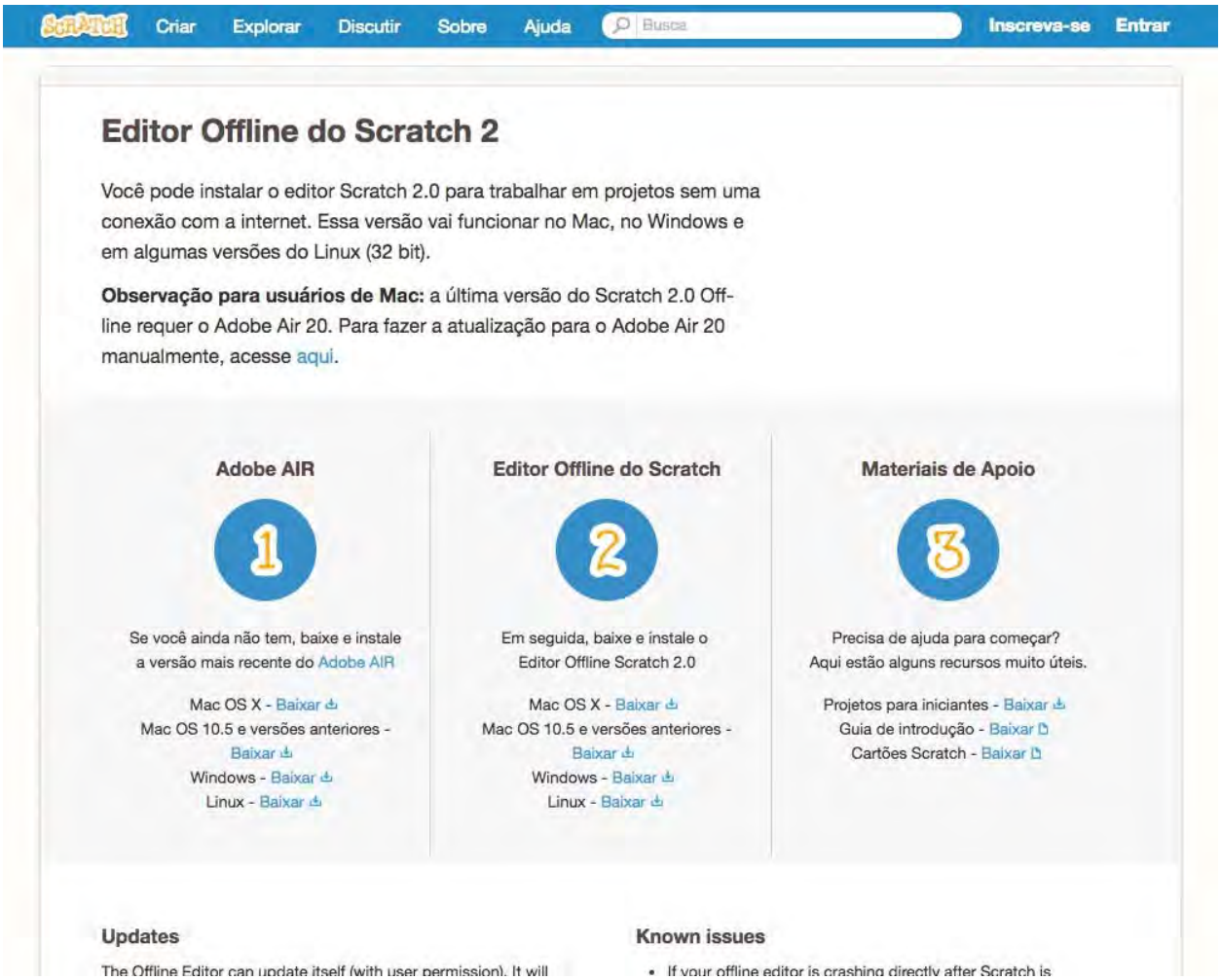


2. Desça a barra de rolagem e clique em Editor Offline para baixarmos a última versão.

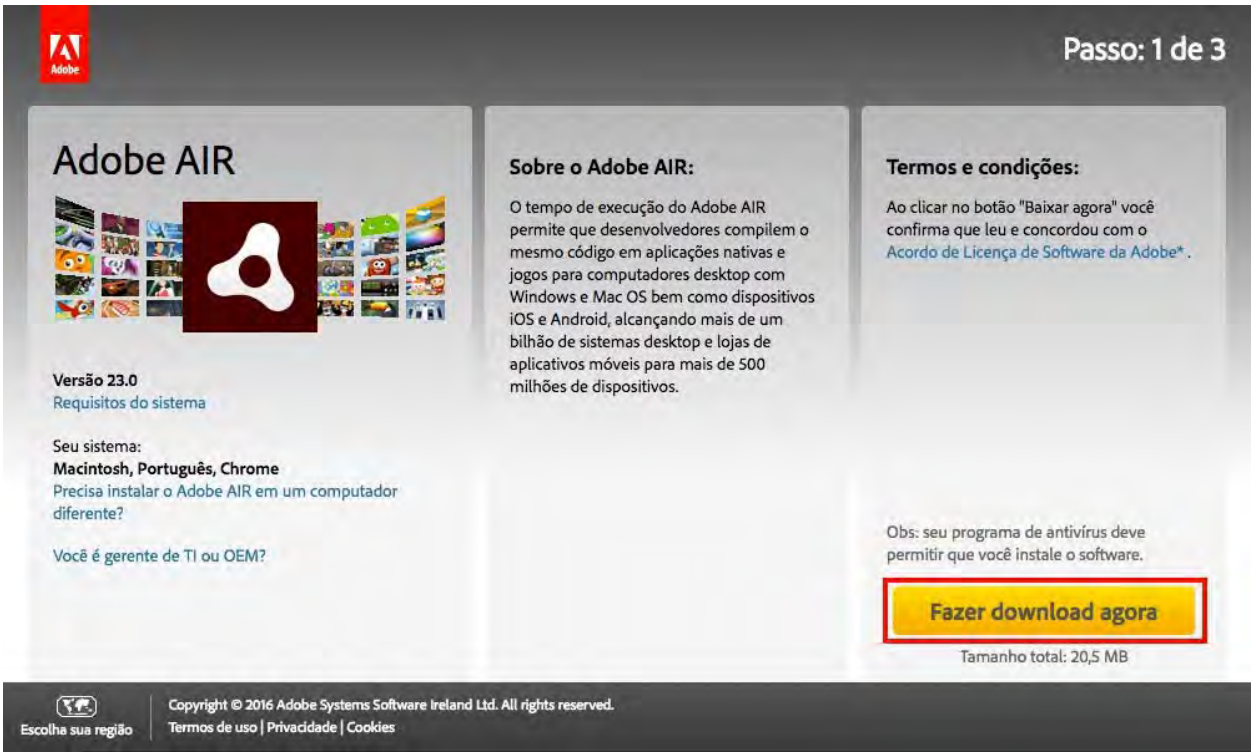


Para prosseguir, é necessária a instalação da plataforma Adobe AIR. Ele é o responsável por permitir que o Scratch seja executado.

3. Clique no link correspondente ao seu sistema operacional no passo número 1, para baixar o Adobe AIR.

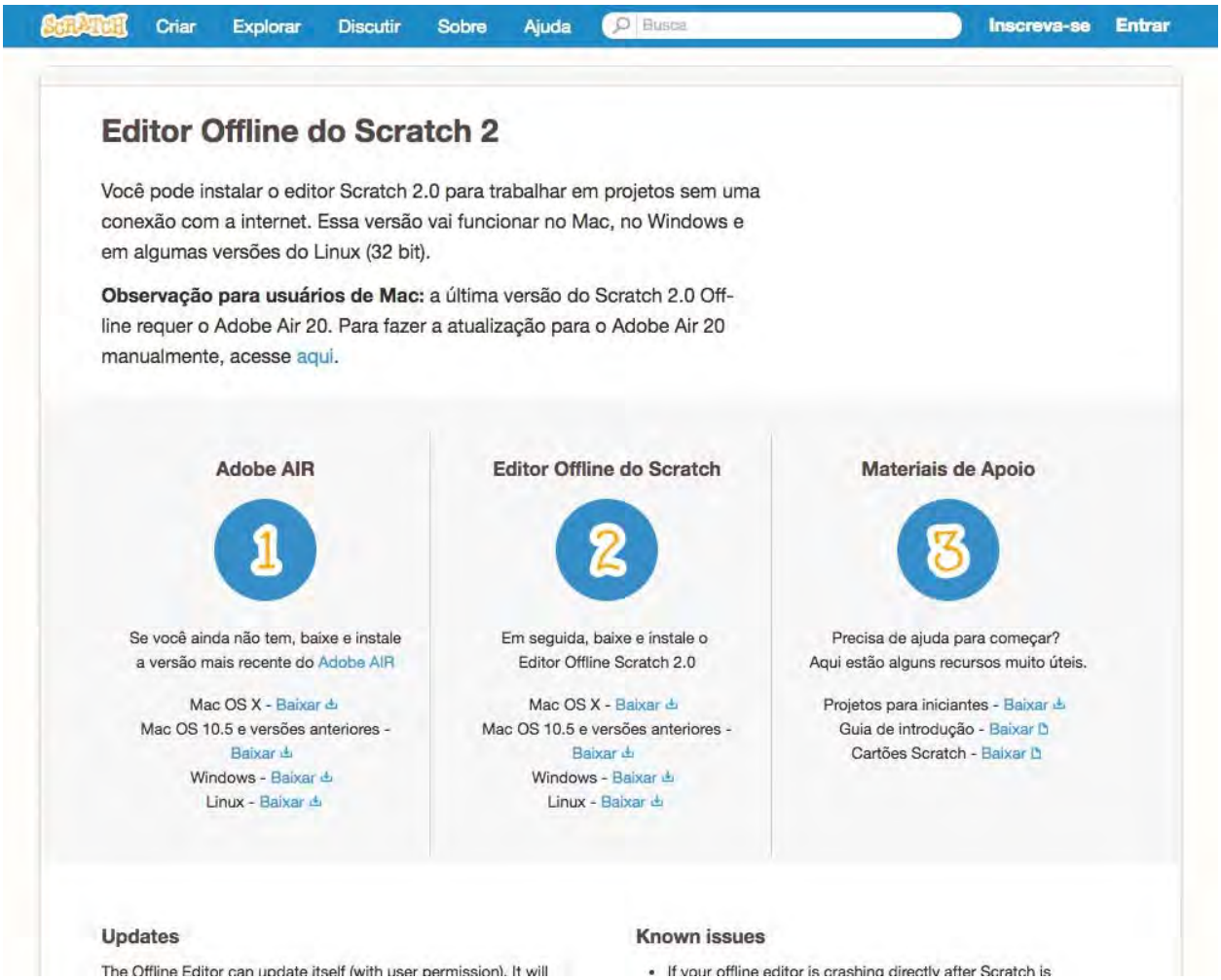


4. Depois, clique em Fazer download agora.

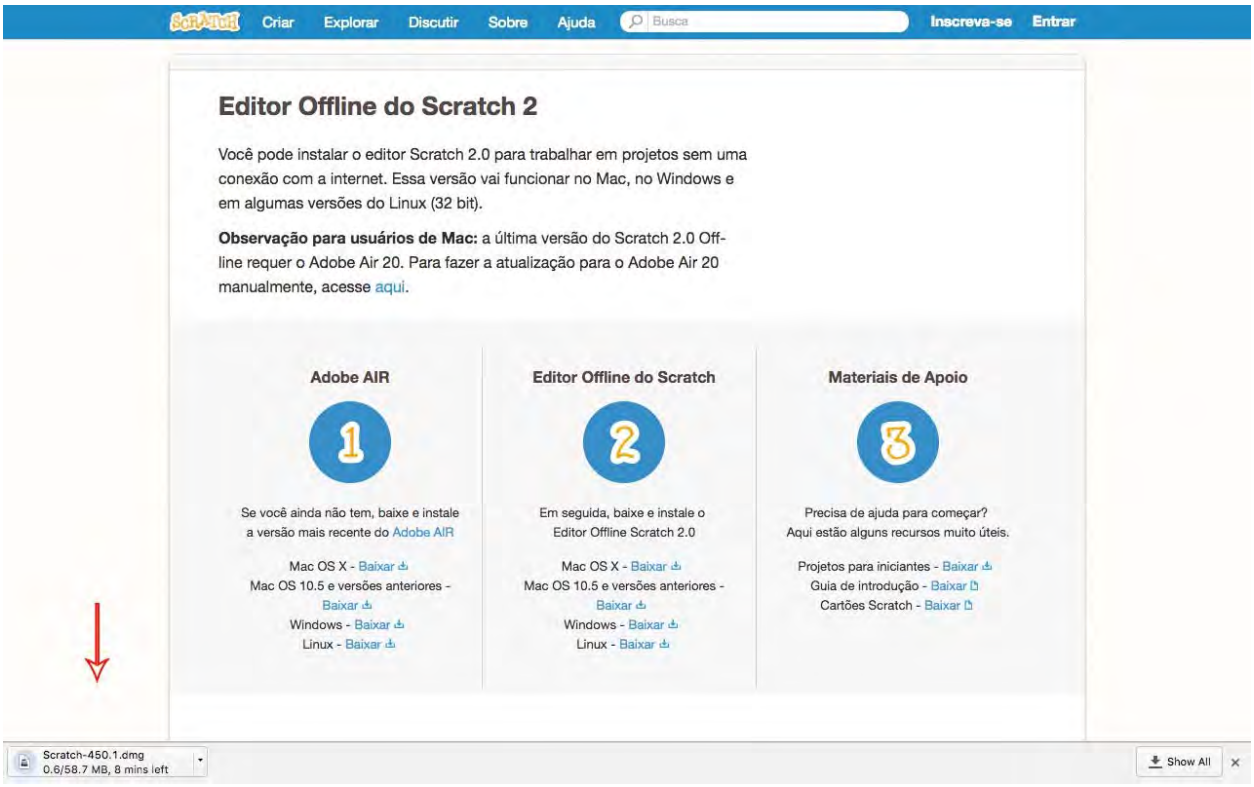


Prossiga a instalação normalmente.

5. Agora vamos baixar o Scratch. Selecione a versão corresponde ao seu Sistema Operacional.

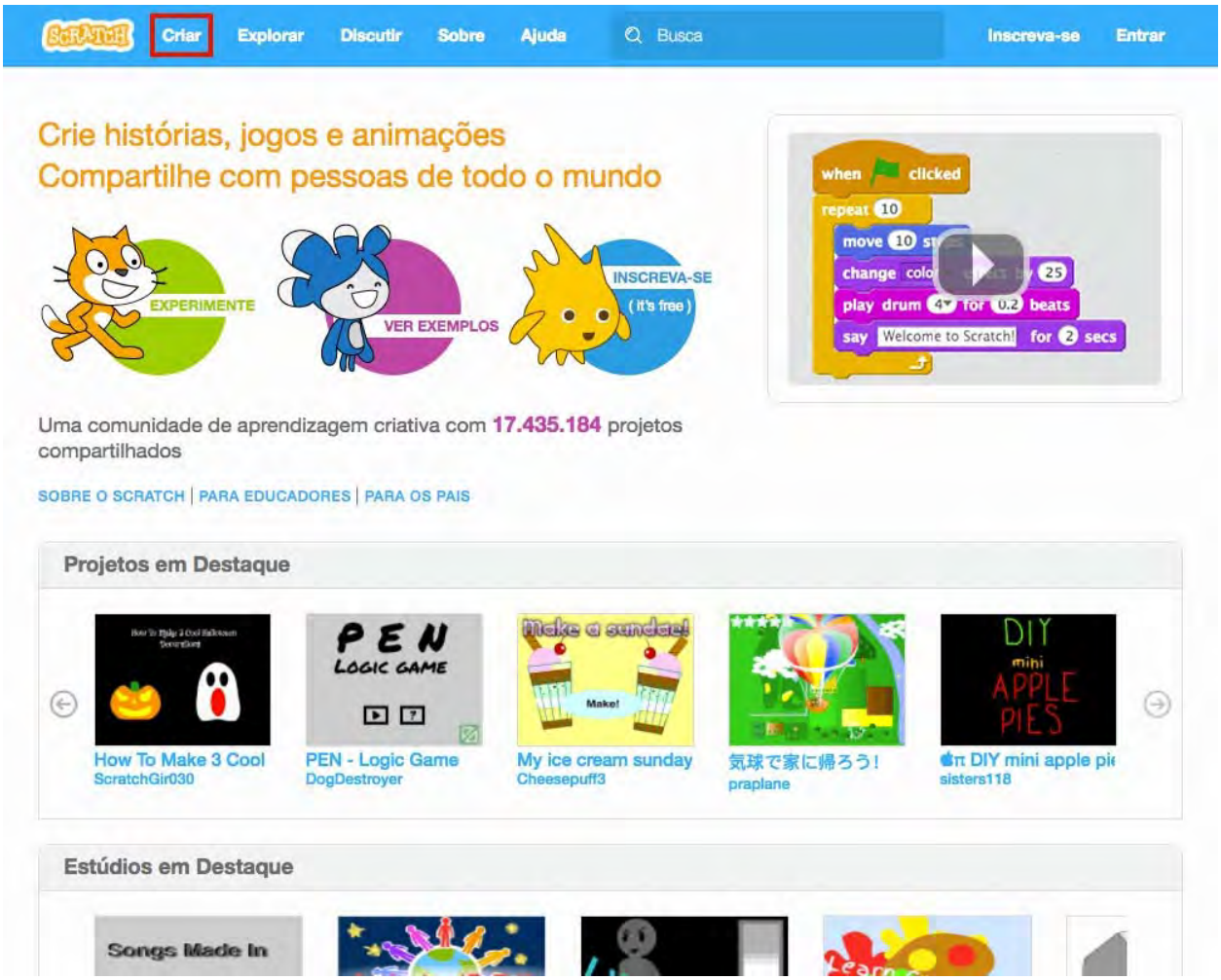


6. Veja que, logo após clicarmos, o Scratch começa a ser baixado. Prossiga a instalação normalmente.



Se você quiser, pode também usar a versão online que roda direto no navegador.

1. Acesse http://scratch.mit.edu e clique em Criar.



2. Espere o carregamento e, logo em seguida, aparecerá a interface do Scratch. Muito simples mesmo.

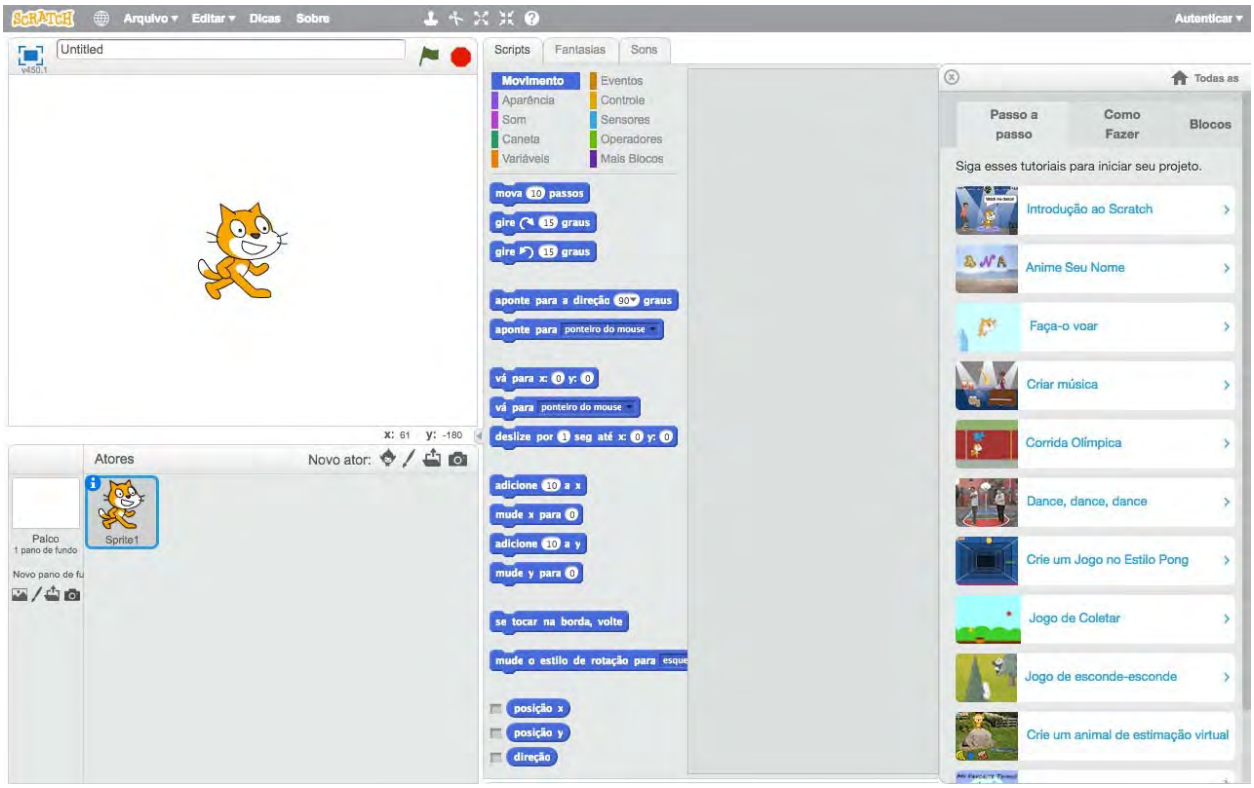


Figura 1.10: Interface Scratch versão web

1.2 Conhecendo o Scratch

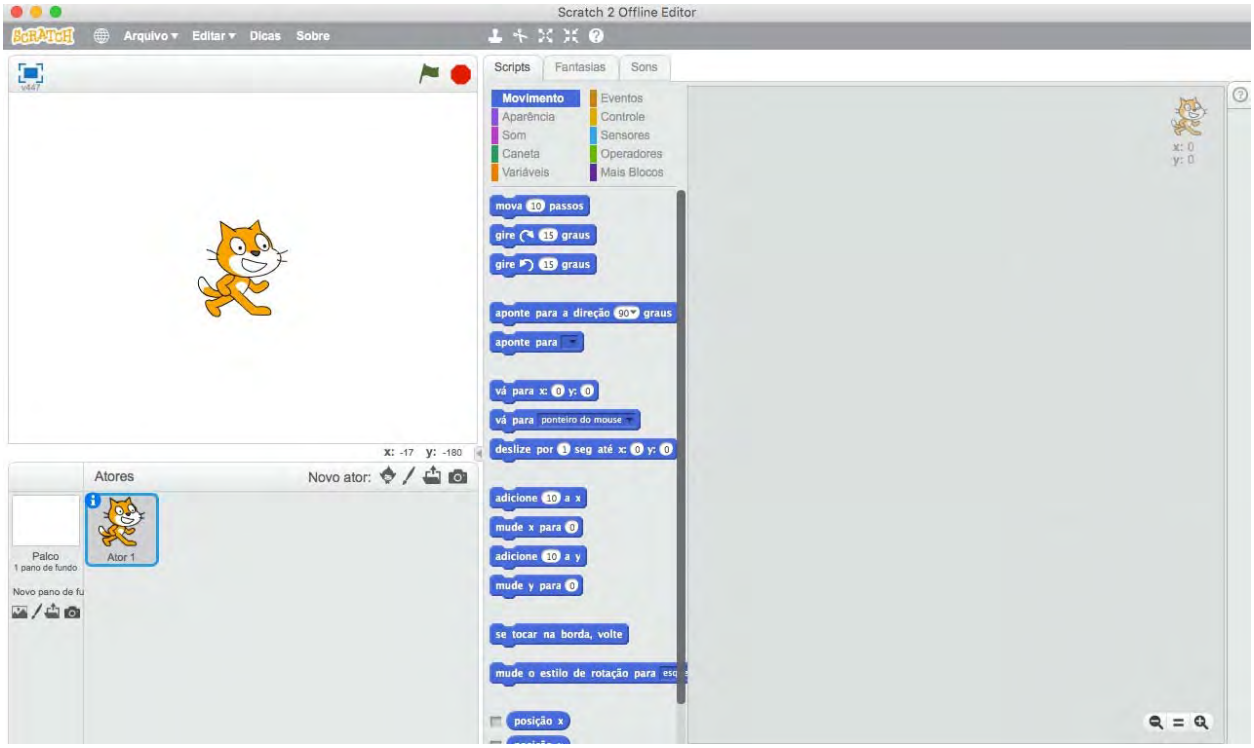
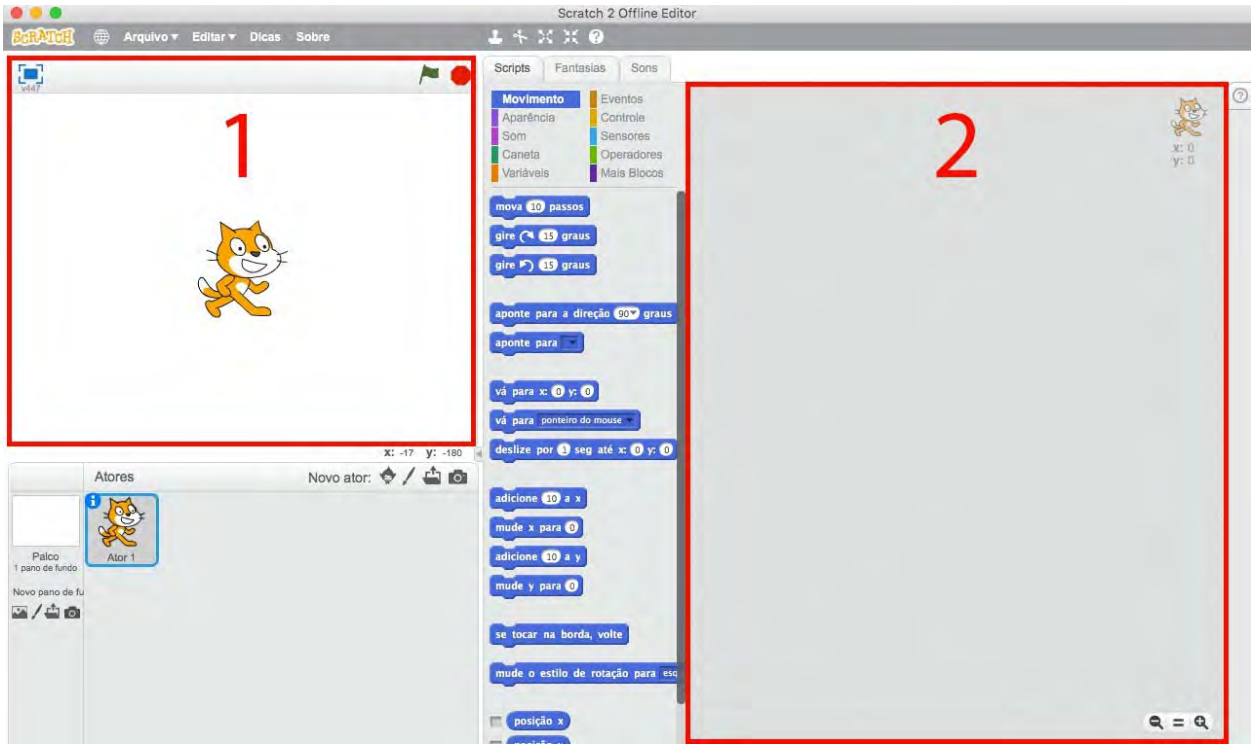
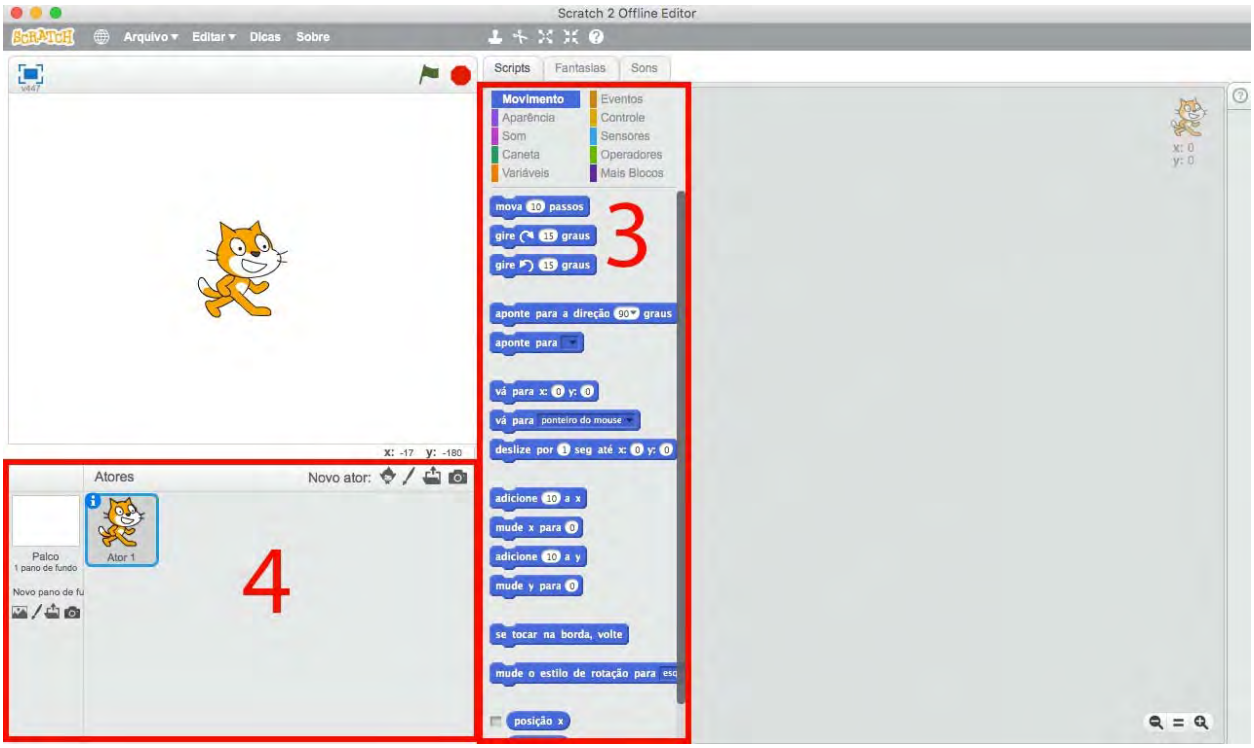


Figura 1.11: Interface do Scratch v450.1

O Scratch tem uma interface bastante intuitiva e simples de usar. Veremos a seguir em vermelho a divisão de algumas áreas que facilitarão nosso estudo.

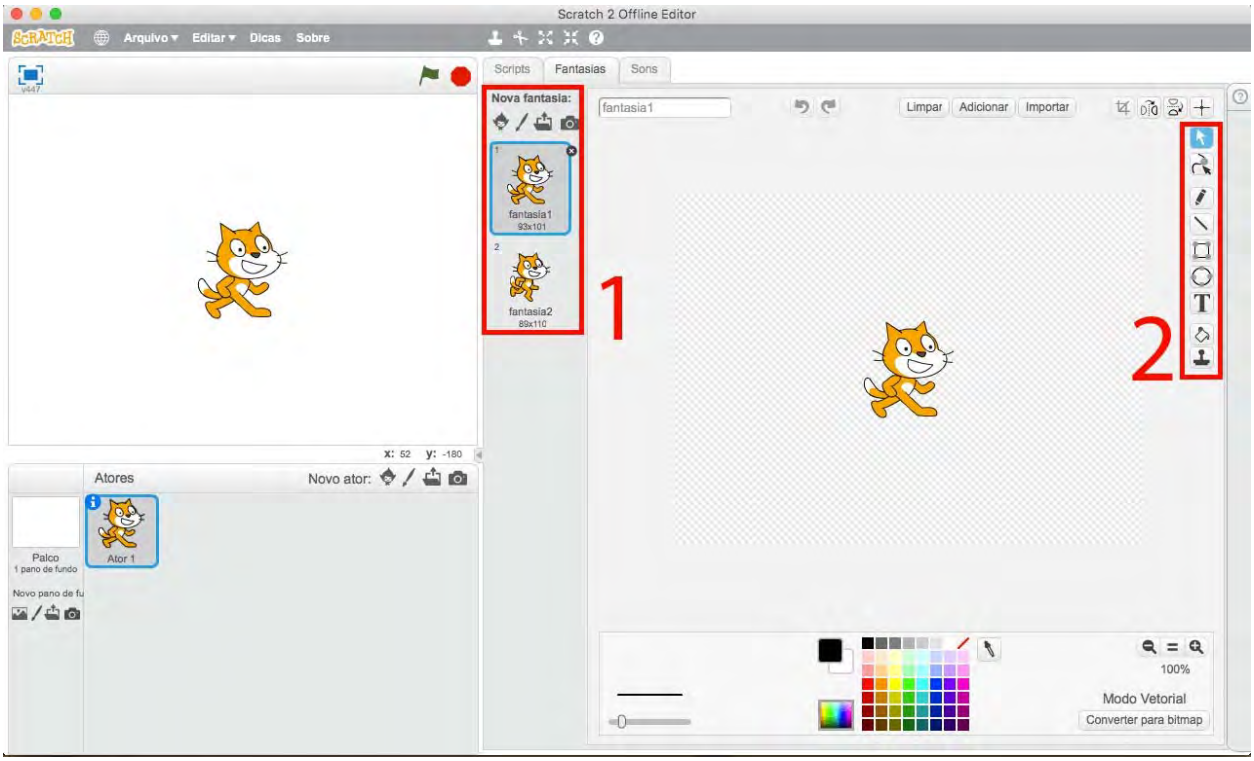


- **1 — Palco:** esta é a área que permite visualizar toda a parte gráfica e animada do projeto, e também é onde são realizados os testes.
- **2 — Área de Scripts:** aqui é onde vamos programar. Todo código que será desenvolvido e os blocos que vamos arrastar sempre ficarão nesta área.



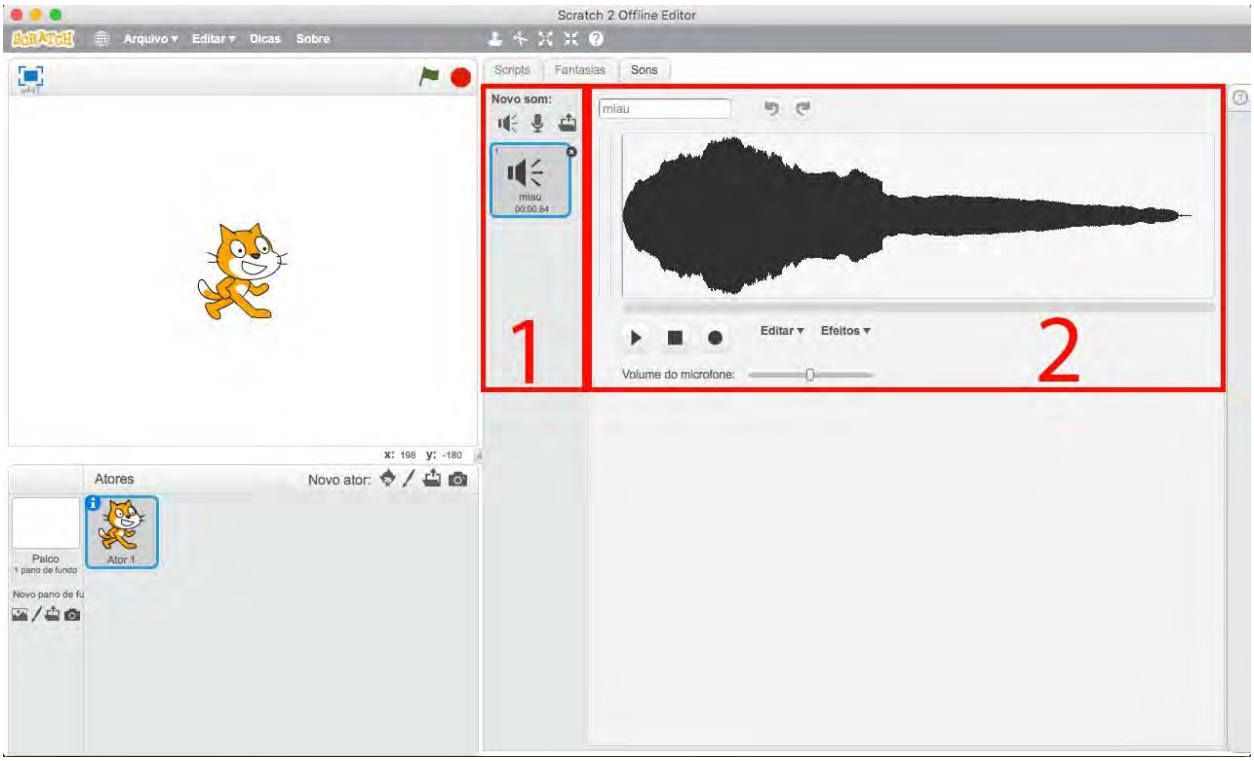
- **3 — Paleta de Blocos:** todos os blocos que usaremos estarão nesta área. Veja que eles são divididos por cores e, dentro de cada cor, os blocos têm diferentes formatos. Cada cor é responsável por uma função e, dependendo do seu formato, ele retorna ou lê um tipo de valor. Lembre-se de que na **Área de Scripts** estarão os blocos que serão *compilados*, ou seja, transformados em programa que visualizaremos através do **Palco**.
- **4 — Lista de Atores:** esta área permite visualizar, adicionar ou alterar imagens que aparecerão no Palco, também conhecido como **Atores**.

O Scratch permite também trabalhar com desenhos através da aba *Fantasias*. Nele é possível criar, editar e controlar a fantasia que o Ator ou o Palco estará utilizando. É possível criar ou carregar um desenho tanto para Atores como para o Palco.



- **1 — Fantasias:** veja que há duas fantasias para um mesmo *Ator1*. Mas por que usamos duas fantasias para um mesmo ator? Neste caso, estamos querendo criar uma animação. Observe que cada fantasia tem o ator em um movimento diferente. Cada vez que trocar a fantasia, o ator terá a sensação de estar se movimentando. Essa é uma das utilidades das múltiplas fantasias. No decorrer do livro, veremos isso na prática.
- **2 — Ferramentas de desenho:** aqui ficam as ferramentas básicas de edição que qualquer editor de imagens tem como selecionar, cortar, texto *etc.*

Para os projetos ficarem mais divertidos, podemos adicionar alguns sons, e isto é possível através da aba Sons. Você pode adicionar um som pré-definido do Scratch, carregar do seu computador ou gravar do microfone direto do Scratch. Sabe aquele jogo de RPG com aquela música sensacional? É possível criar um jogo neste estilo somente com os sons que já vêm no Scratch.



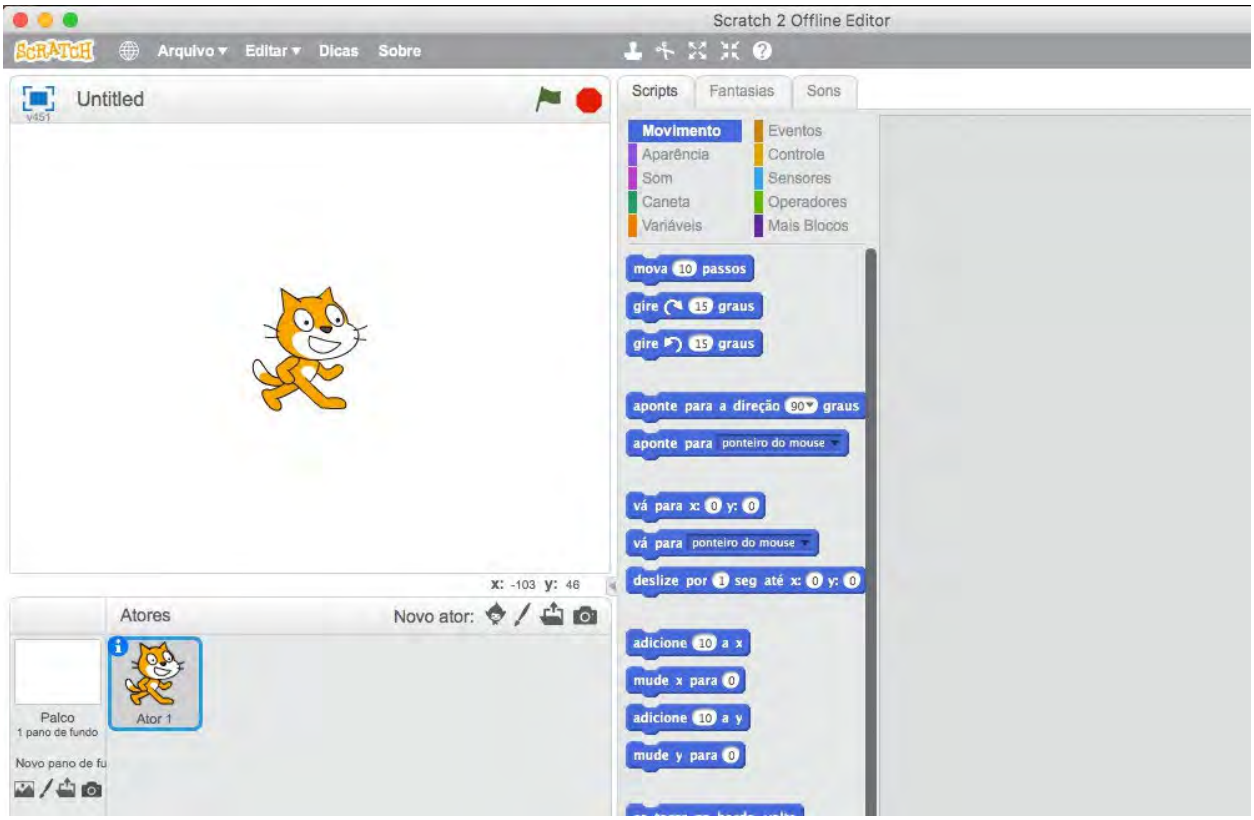
- **1 — Sons:** aqui ficam todos os sons carregados e disponíveis para utilização.
- **2 — Linha do tempo:** nesta área, é possível ver as ondas que compõem o som, além de poder editar, pausar e parar.

Não se preocupe em decorar todos esses nomes. Durante os próximos capítulos, estaremos falando sobre cada um deles de forma mais aprofundada.

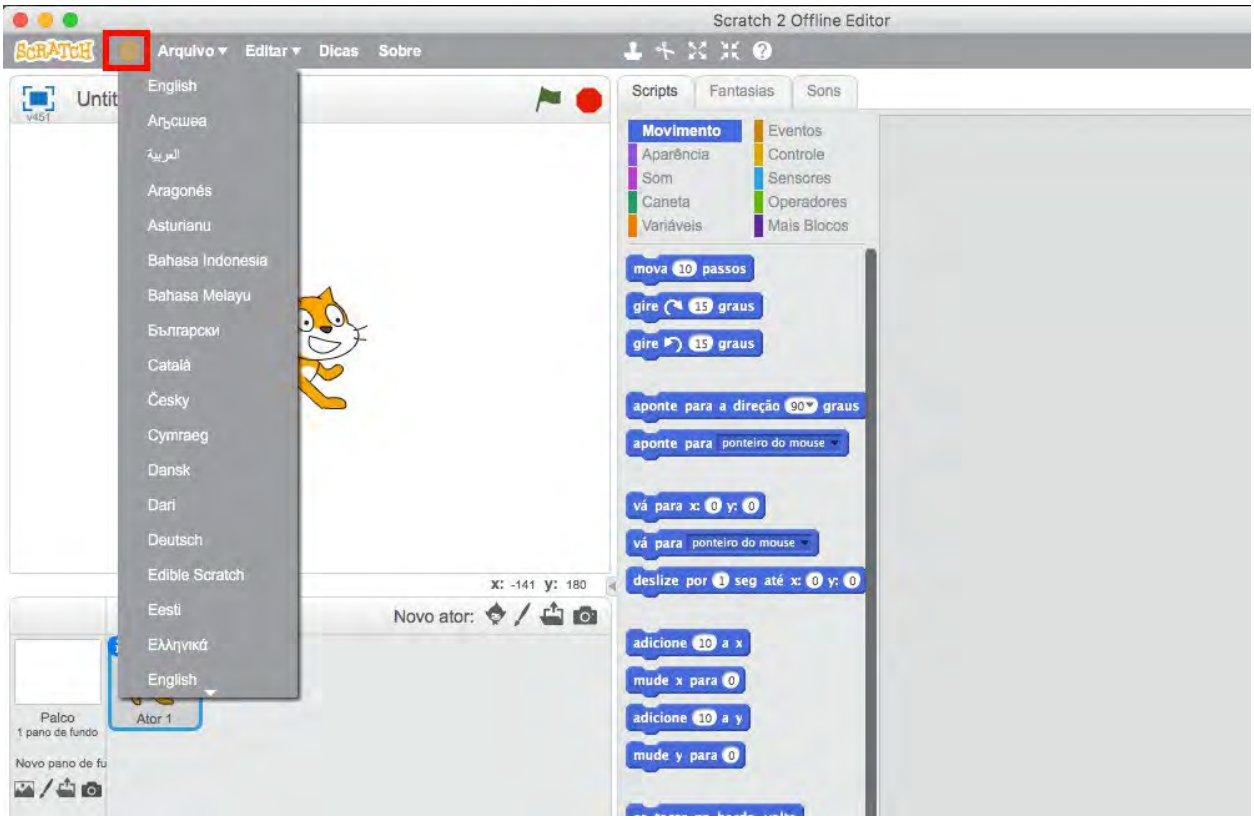
1.3 Scratch em português

Caso você esteja usando o Scratch em outro idioma e queira mudá-lo para português, é muito simples.

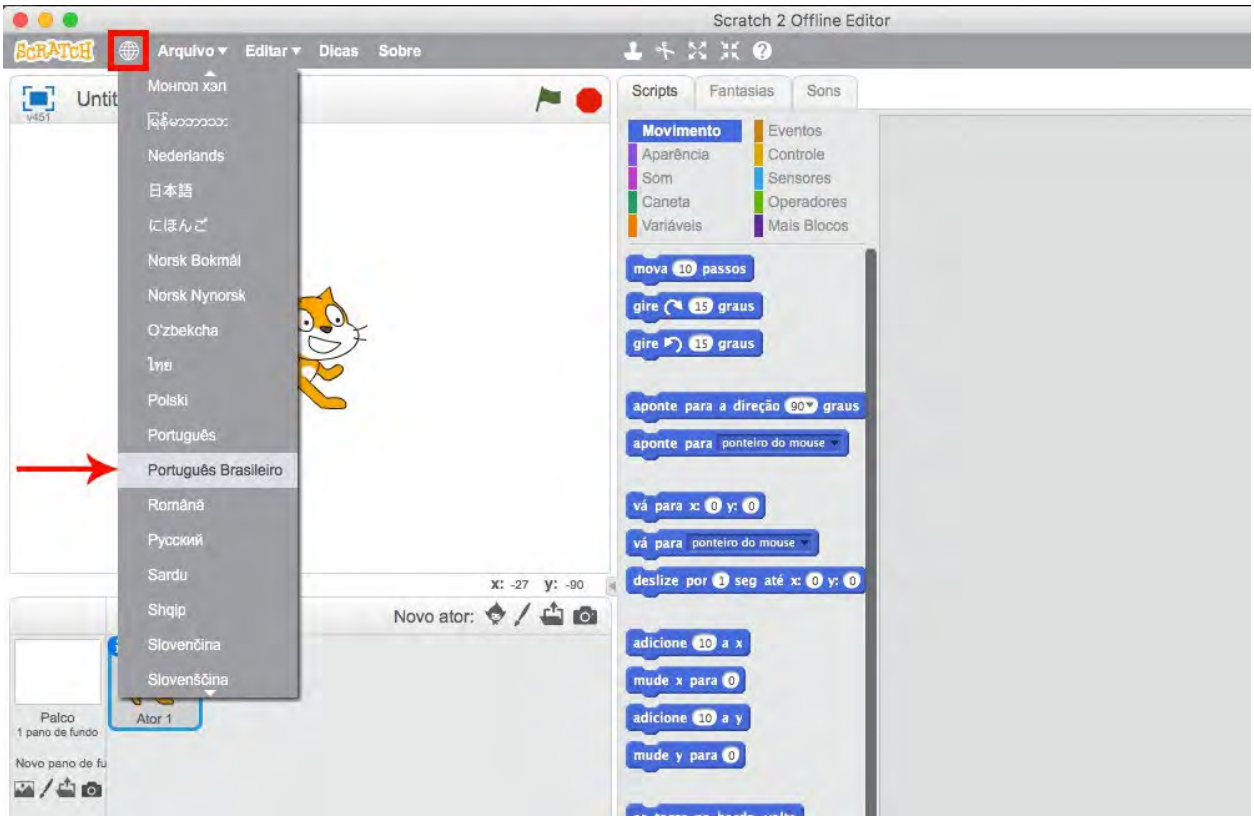
1. Abra o Scratch.



2. Clique no ícone do globo.



3. Selecione *português brasileiro*.



Durante o livro, estaremos sempre utilizando em português.

1.4 Exemplos

Veremos agora alguns projetos para conhecermos melhor o Scratch e o que ele é capaz de fazer. Alguns projetos serão mais simples e outros mais complexos. Neste momento, a fim de ganhar familiaridade com o Scratch, vamos conhecer a interface e funções básicas para, nos próximos capítulos, podermos trabalhar com ferramentas mais complexas.

Animate the Crab

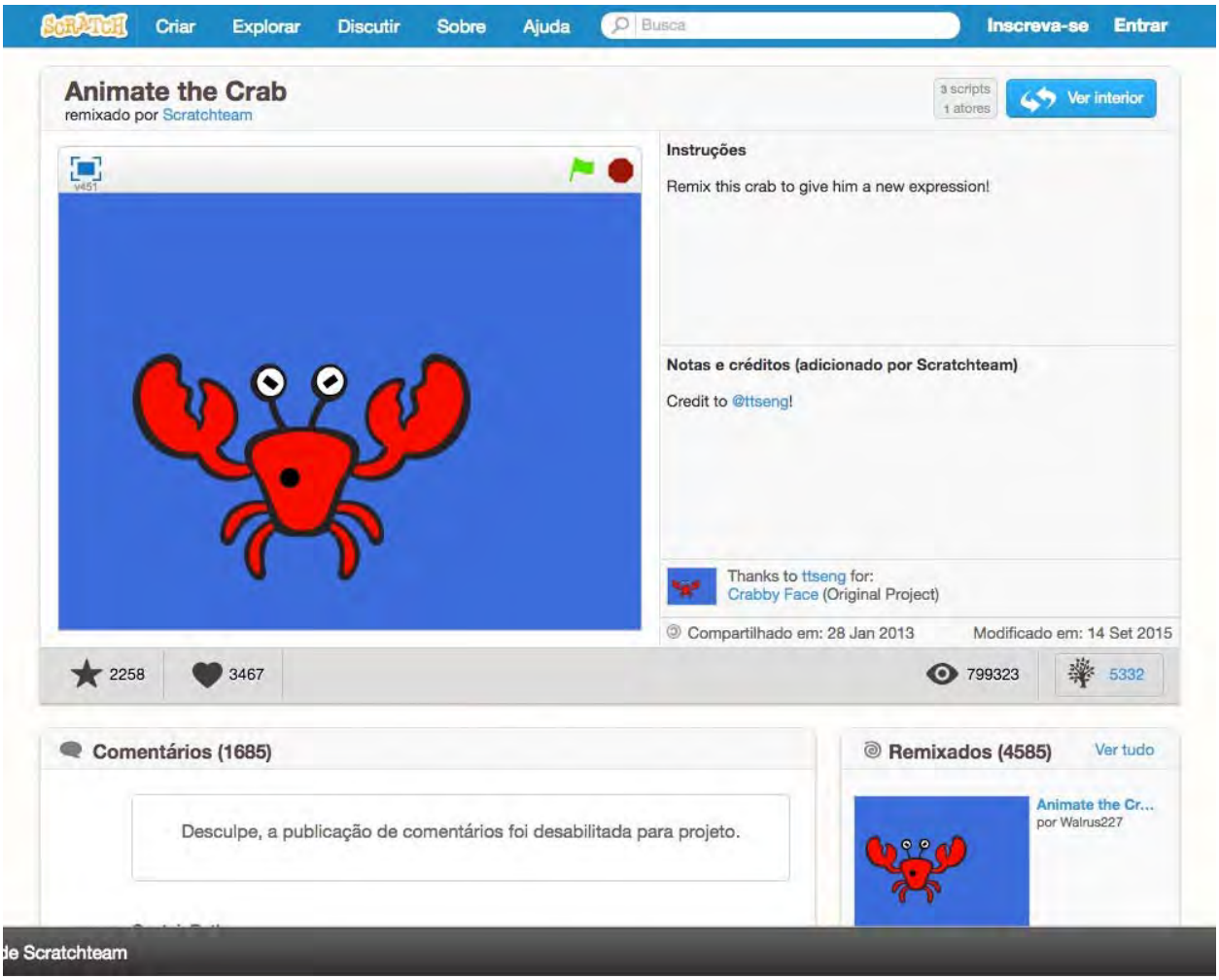


Figura 1.19: Animate the Crab

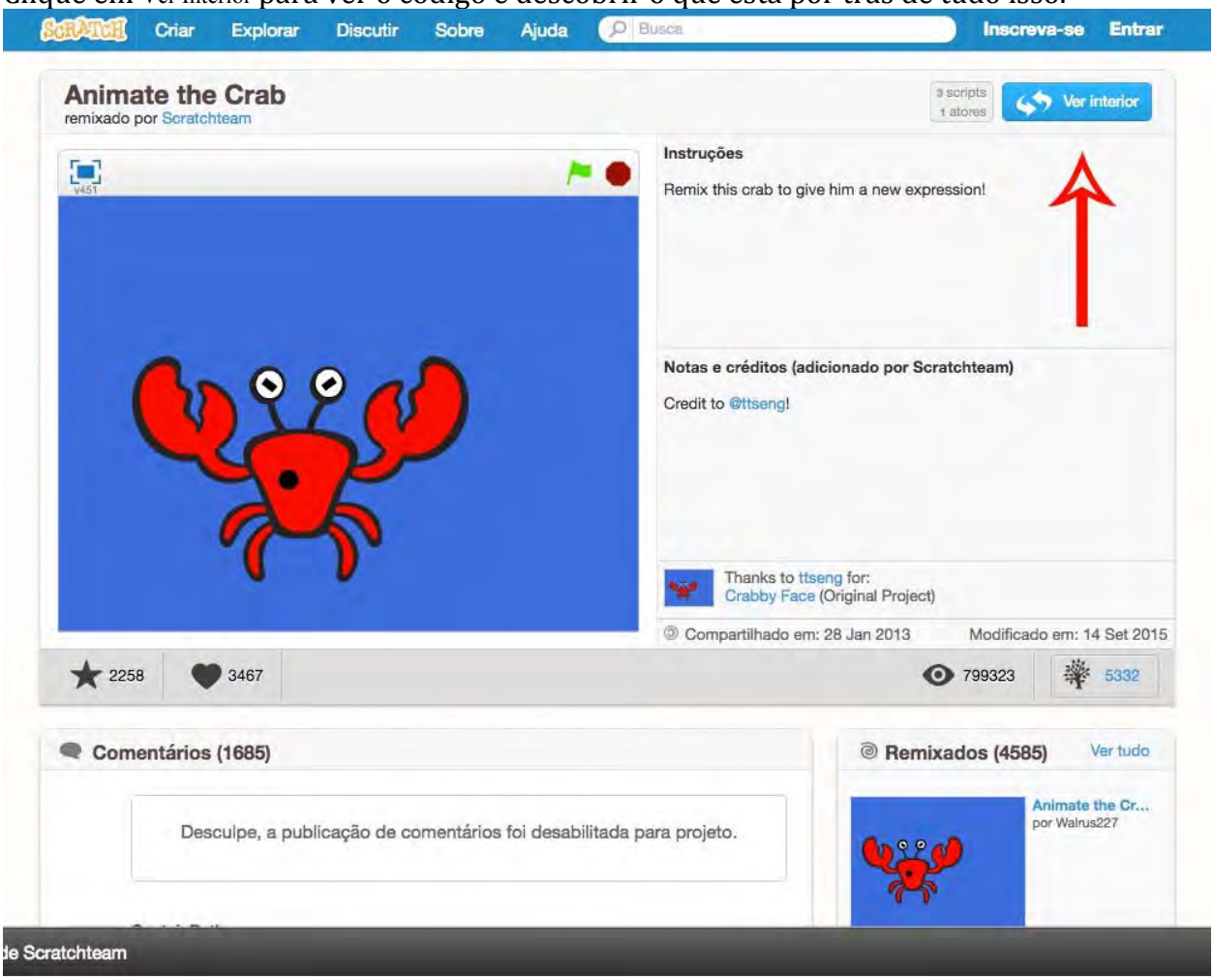
Para visualizar este projeto, acesse: https://scratch.mit.edu/projects/10015059/.

Veja o que a animação faz e tente interagir com ela. Muito interessante, não é mesmo? Neste projeto, temos um caranguejo fazendo um beatbox. Podemos ver que não é um projeto muito complexo e com grandes efeitos, pois é uma simples animação. Só que, por trás disso, há um código que é responsável por dizer ao computador o que deve ser feito.

Muitas pessoas falam que um mágico nunca revela ou não deve revelar seus segredos. Eu penso o contrário: é muito mais interessante ver como funciona e o que está por trás de toda beleza da mágica. E aqui é a mesma coisa.

No Scratch, é sempre possível ver como funciona cada programa. Não há mágica, mas sim lógica, montado como se fosse um quebra-cabeça. Então vamos ver como funciona o nosso querido caranguejo.

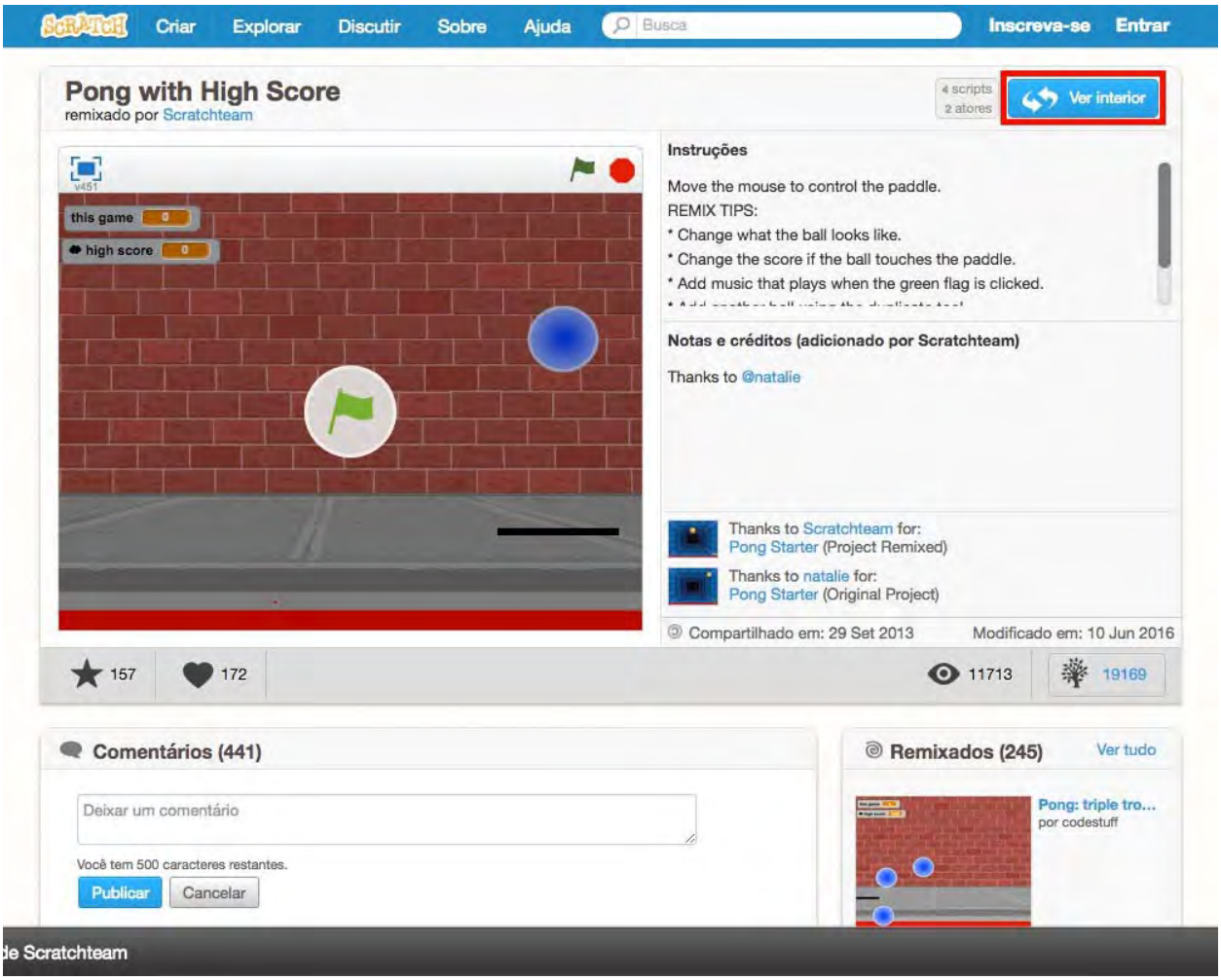
Clique em Ver Interior para ver o código e descobrir o que está por trás de tudo isso.



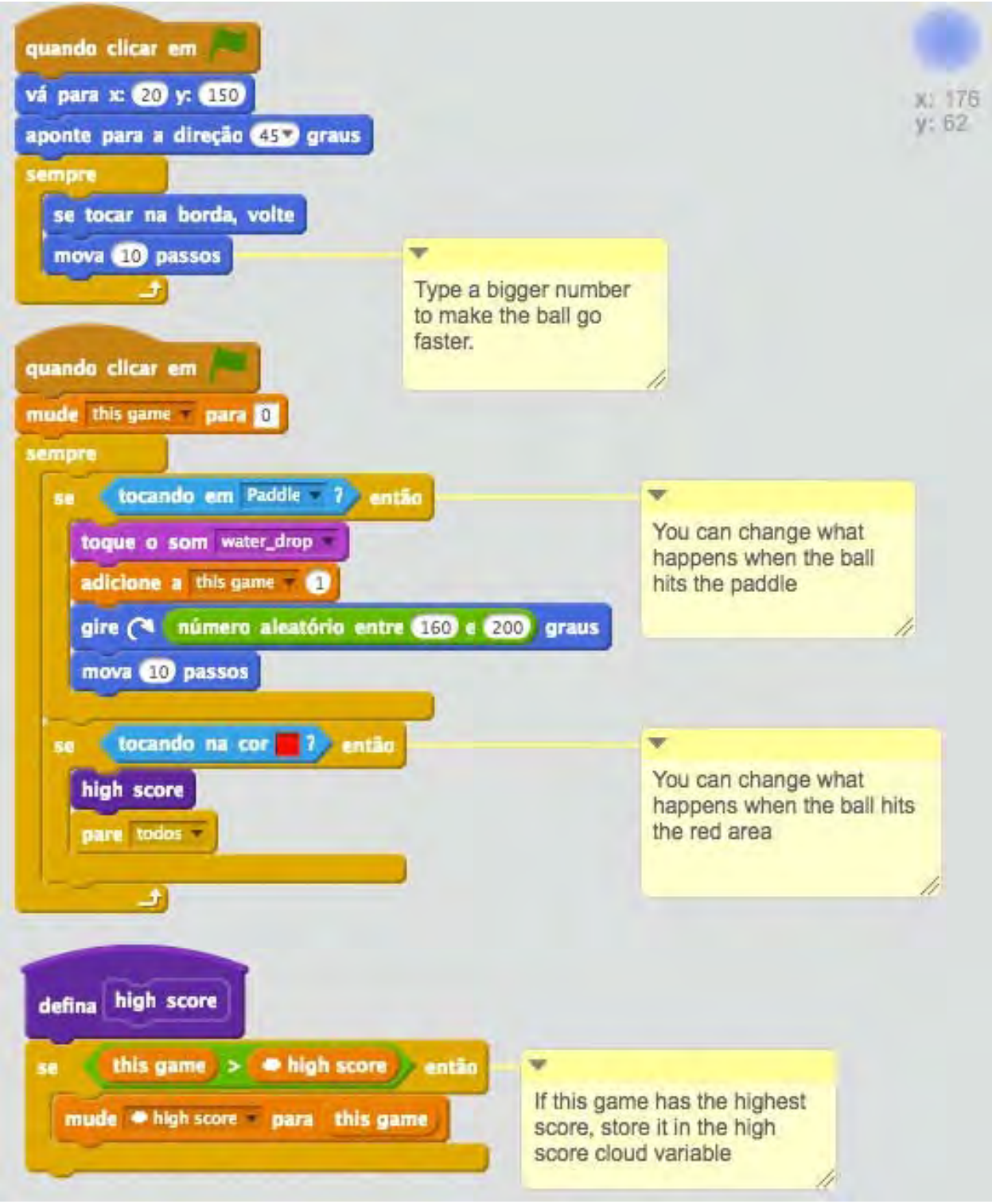
Veja que apareceram na Área de Scripts alguns blocos coloridos. A junção de todos estes blocos cria a animação, e cada bloco separadamente de acordo com a cor e formato é responsável por alguma ação específica na animação. Não entraremos em detalhes no código por enquanto, mas fique à vontade para testar e mexer no código como desejar. Teste os botões, altere alguns valores, troque as fantasias e tente descobrir o funcionamento de cada bloco.

Pong with High Score Neste projeto, temos um pequeno jogo de *Pong*, em que o objetivo é não deixar a bola tocar no chão utilizando o mouse.

1. Acesse: http://scratch.mit.edu/projects/12778537.



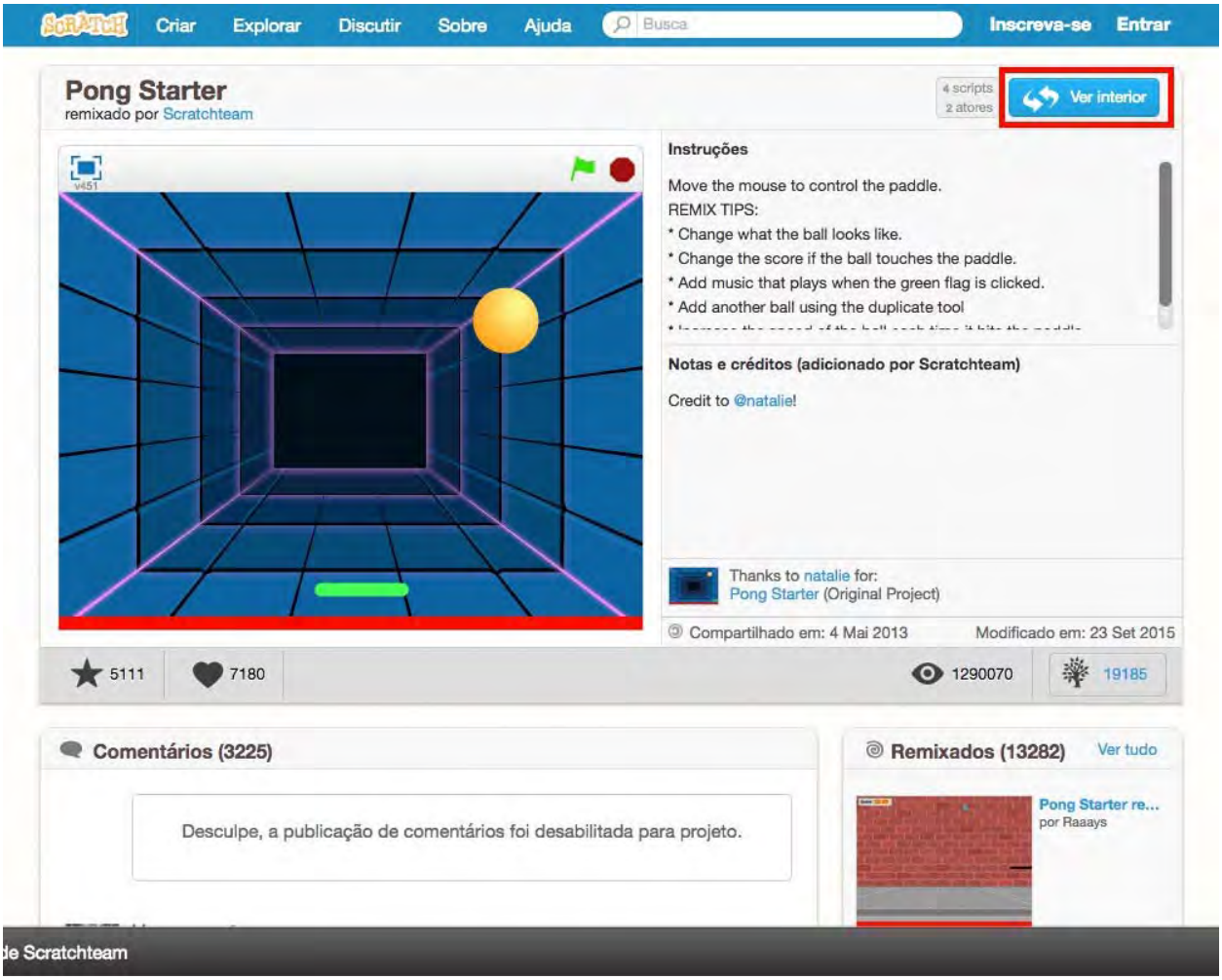
2. Clique em Ver interior.



Pode explorar o código e alterar como quiser. Para testar o jogo, clique na bandeira na parte superior do Palco.

Pong Starter Temos aqui outro exemplo de *Pong*. Veja como é possível criar o mesmo jogo com códigos diferentes.

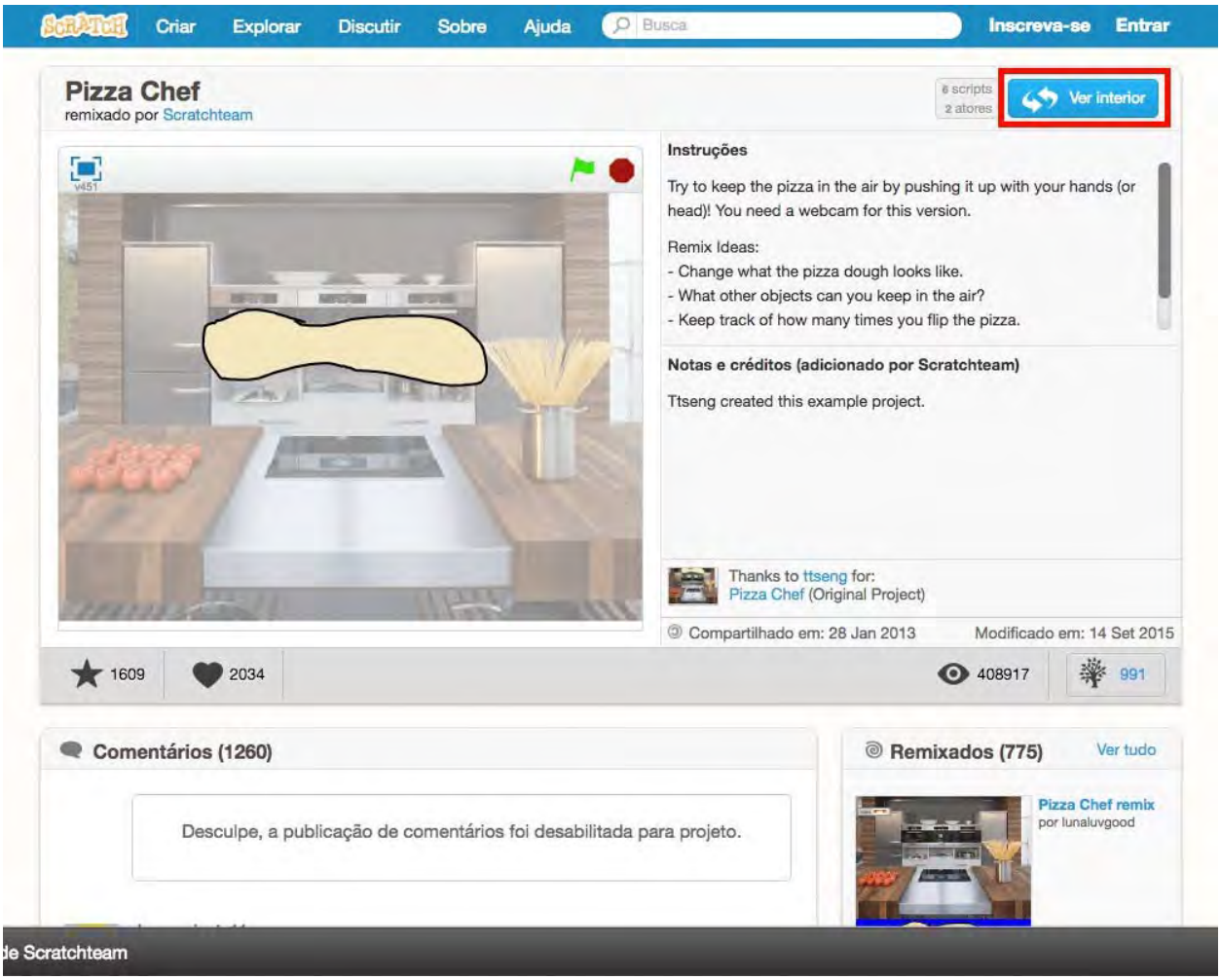
1. Acesse: http://scratch.mit.edu/projects/10128515
2. Clique em Ver interior:



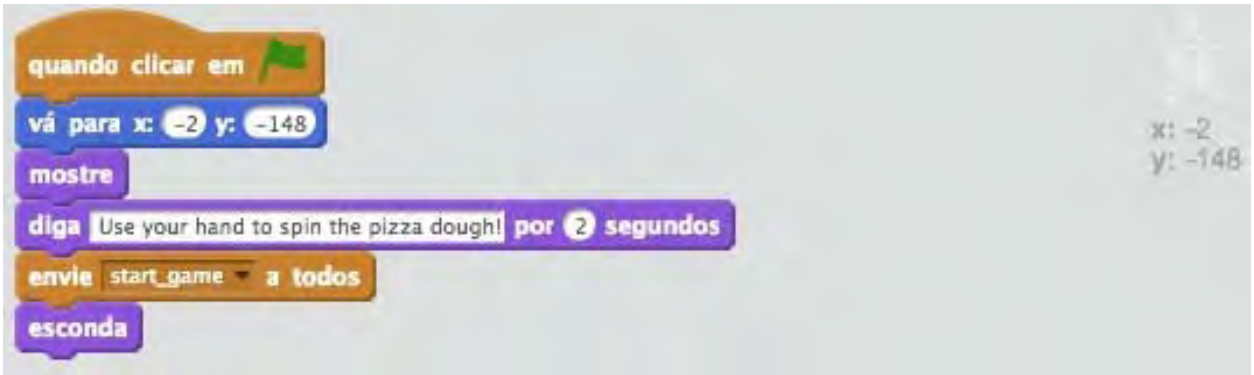
3. Brinque um pouco com o código. Altere, por exemplo, os valores dos movimentos (blocos de cor azul) e veja o que acontece.

Pizza Chef Este é um jogo interessante que utiliza a câmera do computador para movimentar a pizza como se fosse um Chef. Você precisa erguer suas mãos e girar a pizza no ar em frente à sua webcam.

1. Acesse: http://scratch.mit.edu/projects/10015802
2. Clique em Ver interior.

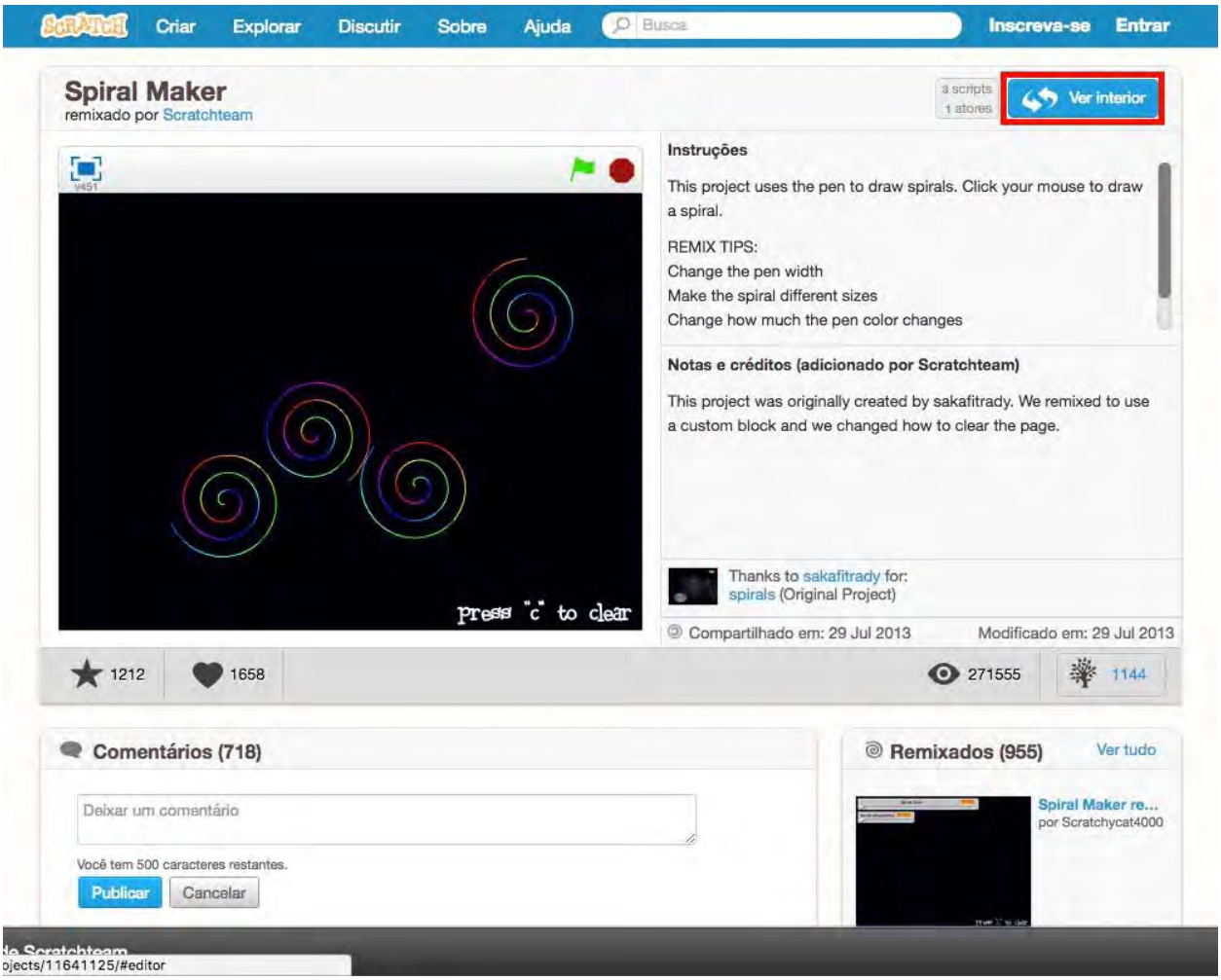


3. Altere o bloco diga e coloque a seguinte mensagem: *Use sua mão para rodar a pizza!*.



Spiral Maker Este é um exemplo de como é possível criar desenhos no Scratch.

1. Acesse: http://scratch.mit.edu/projects/11641125
2. Clique em Ver interior.



3. Veja como funciona a função Caneta, que é responsável por desenhar no Palco. Altere os blocos de cor laranja e verde para ver o que acontece.

1.5 Conclusão

- Neste primeiro capítulo, vimos um pouco da história do Scratch, de como ele surgiu e por que surgiu.
- Vimos que o Scratch possui uma comunidade online que compartilha projetos e permite ao usuário estudar qualquer projeto.
- Vimos a interface e como podemos dividi-la.
- Vimos alguns exemplos de jogos e animações feitos pela comunidade do Scratch e os códigos por trás deles.

No próximo capítulo, veremos uma ferramenta extremamente importante que é a base para qualquer programa. Estaremos aprendendo sobre a Estrutura Condicional **Se e Senão**, que nos permitirá realizar a tomada de decisões. Vamos também começar o desenvolvimento de um jogo de labirinto, no qual vamos aplicar nossos conhecimentos de cada capítulo em uma parte do jogo e que será finalizado no último capítulo.

CAPÍTULO 2

Se e Senão

Após ter lido o capítulo de introdução, você deve estar ansioso para começar a programar, então vamos começar logo com isso. O primeiro assunto que vamos tratar aqui é sobre uma estrutura muito utilizada em programação.

Aprenderemos a primeira estrutura básica, responsável por tomadas de decisão, ou seja, a estrutura que permite escolher uma opção dentre algumas possibilidades. Os comandos *Se* e *Senão* são os responsáveis por realizar a tomada de decisão com base no resultado da comparação entre dois ou mais valores, usando os operadores de comparação.

Diariamente, tomamos decisões a todo o momento, e por mais que nós não percebemos, estamos usando os comandos Se e Senão para nos ajudar a tomar a melhor decisão. Suponha que vamos comprar um computador e só temos R$ 1.000 reais. Se o computador custar mais de R$ 1.000, não comprar; senão comprar.

Percebeu o uso do operador de comparação (maior, >) e dos comandos Se e Senão? Veja a seguir os comandos escritos em pseudocódigo, que é uma linguagem mais próxima ao que utilizamos no dia a dia para descrever algoritmos, em C e na linguagem Scratch.

Pseudocódigo

```
Se o computador custar mais de 1000
 Não comprar
Senão
 Comprar

Se preço > 1000
 Não comprar
Senão
 Comprar
```

Linguagem C

```
if(preco > 1000){
 comprar();
}
else{
 naoComprar();
}
```

Scratch

2.1 Operadores de comparação

Os *operadores de comparação* são aqueles que você estudou na escola nas aulas de matemática, lembra? Eles são usados quando precisamos determinar quando um valor é maior ou menor que outro, ou precisamos saber se um número é igual ou diferente de outro. Os operadores mais básicos são sinais bem conhecidos:

- > — Maior
- < — Menor
- = — Igualdade

Todas as vezes que realizamos uma comparação, o programa nos retorna um valor sendo ele verdadeiro ou falso. Ou seja, para toda ação, existe uma reação. Seria mais ou menos isso que significa "retornar".

Para cada comparação, há um valor verdadeiro ou falso sendo retornado. Isso é o que nós chamamos de valores *Booleanos*, ou valores lógicos, que são representados como verdadeiro ou falso. Vamos conhecer os operadores de comparação do Scratch.

Operador	Exemplo
 Igual (=)	 É o **preço** igual a **1000**?
 Maior (>)	 É o **preço** maior que **1000**?
 Menor(<)	 É o **preço** menor que **1000**?

Observe que os operadores são blocos que têm uma forma hexagonal (de 6 lados) de cor verde, e isto significa que o resultado obtido desta comparação será sempre um valor booleano. Lembre-se, todo bloco em forma hexagonal retorna um valor verdadeiro ou falso. Vamos entender melhor como funcionam estas comparações utilizando letras.

Sim, é possível comparar caracteres usando os operadores. Estamos acostumados a sempre comparar número, perguntando qual número é maior ou menor. Mas no Scratch, nós conseguimos comparar inclusive caracteres.

Operação	Resultado
 **a** é igual à **A**?	Verdadeiro
 **A** é menor que **b**?	Verdadeiro
 **B** é maior que **C**?	Falso

As comparações com letras não são *case sensitive*, o que significa que não importa se a letra é maiúscula ou minúscula. Letras maiúsculas e minúsculas não são diferenciadas. Todos os operadores se encontram na aba Operadores.

2.2 Operadores lógicos

Vamos ver outro tipo de operador que são os *operadores lógicos*. Eles trabalham com valores booleanos, ou seja, **V (verdadeiro)** ou **F (falso)**. Assim como nos operadores de comparação, ele também retorna um valor verdadeiro ou falso, porém só aceita como entrada os valores verdadeiro ou falso. Ficou confuso?

Tente entender o seguinte: todas as operações realizadas com os operadores lógicos sempre vão nos retornar um valor lógico **V** ou **F**. Vamos supor que você tenha feito um vestibular e sido aprovado para um certo curso. As condições para a matrícula são: ser brasileiro e ter nota mínima de 6. Ou seja, você só poderá fazer a matrícula se for brasileiro **E** ter nota mínima 6. As duas condições têm de ser verdadeiras.

Se você é brasileiro, mas não tem nota mínima 6, não poderá fazer a matrícula. Se tiver nota mínima 6, mas não for brasileiro, também não poderá fazer a matrícula. Nesta situação, estamos utilizando o operador **E**. Existem 3 operadores lógicos: **E**, **OU** e **NÃO**. Olhando a tabela a seguir, podemos entender melhor.

	E e	**OU** ou
VV	V	V
VF	F	V
FV	F	V
FF	F	F

Veja que, quando temos duas condições verdadeiras **(VV)**, nosso operador **E** resultará no valor booleano **V**. Se temos apenas uma condição verdadeira **(VF, FV)**, ele resultará no valor booleano **F**. Se as duas condições são falsas, nosso resultado booleano será **F**.

O operador **OU** apenas exige que uma das condições seja verdadeira. Veja que, quando temos duas condições verdadeiras **(VV)**, nosso valor booleano será **V**. Se temos apenas uma condição verdadeira, nosso valor booleano será **V** também. Se temos as duas condições falsas, nosso valor será falso.

Lembre-se: operador **E**, exige que todas as condições sejam verdadeiras, e o operador **OU** que apenas uma seja verdadeira. Temos ainda o operador **NÃO**. Sua função é bem simples. Ele apenas inverte o valor resultado. Veja a tabela:

	NÃO não
V	F
F	V

2.3 Labirinto

Regras do jogo

Muito bem, caro leitor! Após ter lido todo esse conteúdo inicial, vamos agora para a parte divertida e começar logo a programar. Mas antes disso, é necessário entender as regras do jogo.

Qual o objetivo? É bem simples, o objetivo é capturar o máximo de bolinhas no tempo de 60 segundos. E as possibilidades do jogo são as seguintes:

1. **Tempo** > 0 e **Pontos** > 0
significa que ele está dentro do tempo permitido e fez pelo menos 1 ponto.
Resultado: ***GANHOU***

2. **Tempo** > 0 e **Pontos** = 0
significa que ele ainda está dentro do tempo permitido porém não fez pontos.
Resultado: ***PRECISA CAPTURAR BOLINHAS***

3. (**Tempo** = 0 e **Pontos** > 0) ou (**Tempo** = 0 e **Pontos** = 0)
significa que acabou o tempo, porém fez pontos ou não fez pontos. Em qualquer um dos casos ele não ganhou, pois acabou o tempo.
Resultado: ***PERDEU***

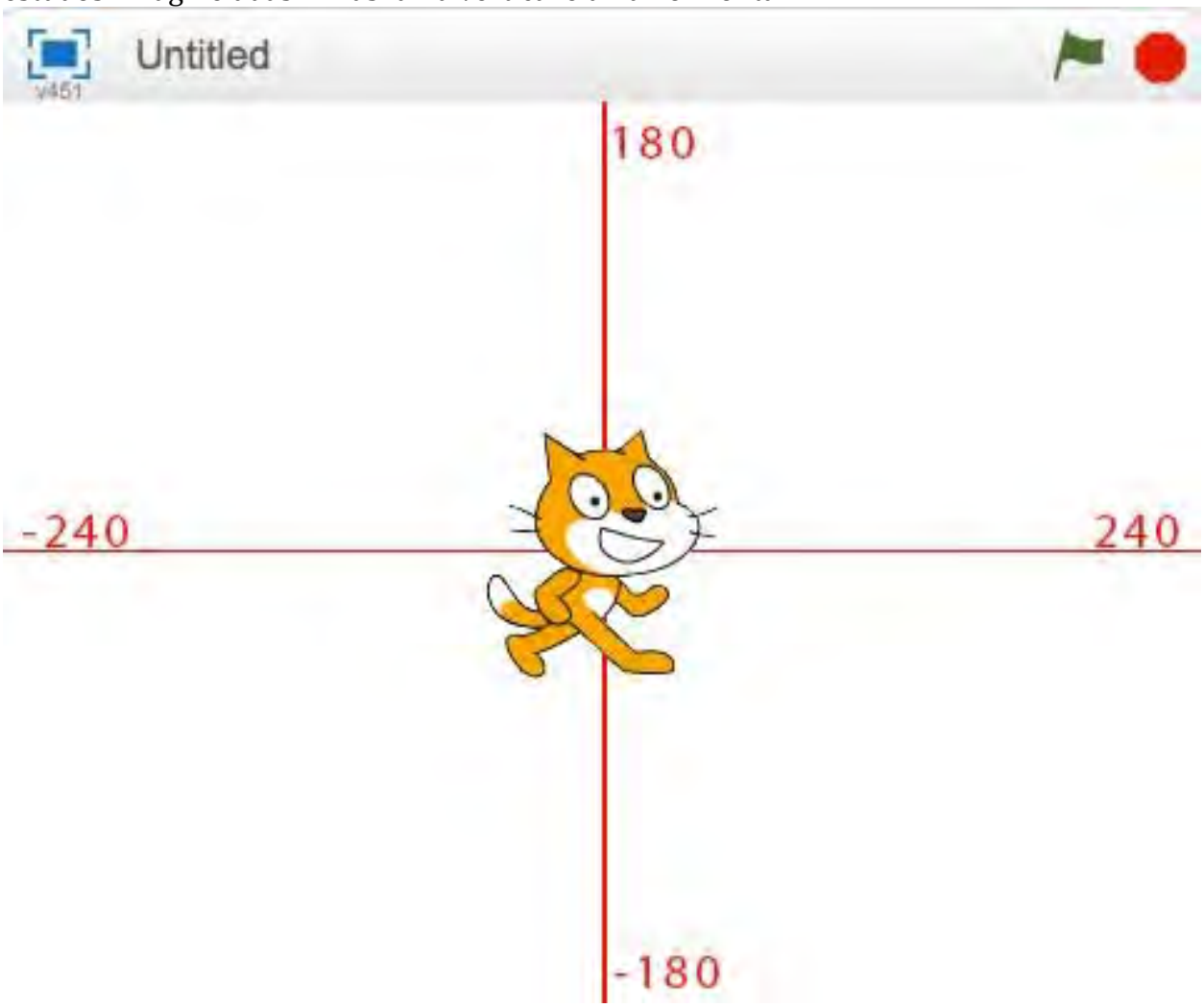
Etapas do desenvolvimento Desenvolveremos a primeira parte do jogo de Labirinto. Nesta primeira parte, precisamos adicionar:

- o mapa do jogo;
- o personagem;
- a movimentação do personagem no mapa.

Como se trata de um jogo com mapa, vamos trabalhar com coordenadas cartesianas.

Plano cartesiano Você se lembra do plano cartesiano? Aquele sistema de coordenadas que utiliza dois eixos: **X** e **Y**? Pois então, é através dele que determinaremos posições no Scratch.

Você vê como alguns assuntos aprendidos na aula de matemática estão sendo usados aqui em programação? E isto se tornará cada vez mais comum conforme você avança seus estudos. Imagine duas linhas: uma vertical e uma horizontal.

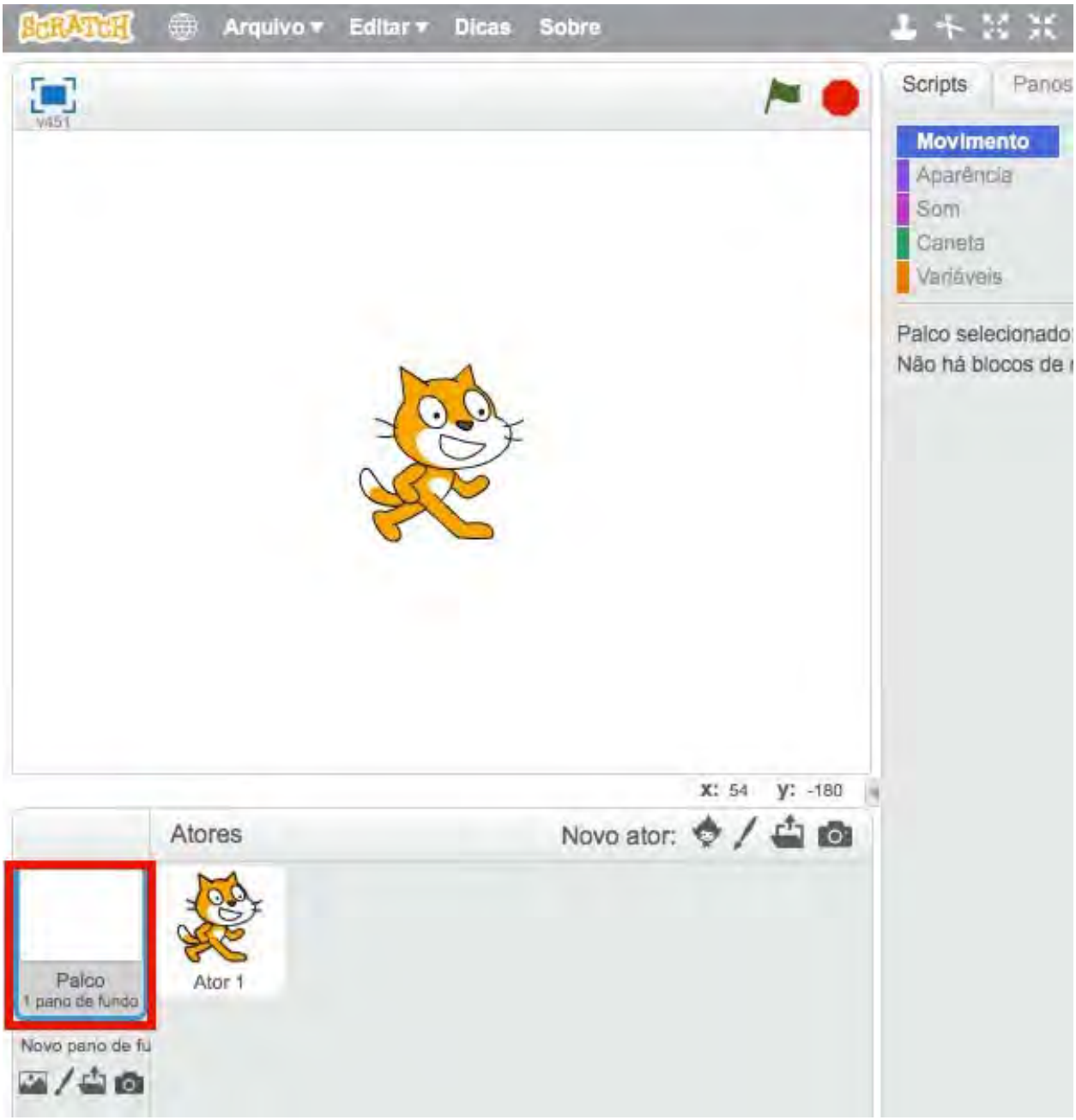


O eixo **X** (horizontal) vai de -240 a 240. O eixo **Y** (vertical) vai de 180 a -180. Independente do tamanho do seu monitor, o Palco sempre terá esse tamanho e será limitado por esses valores nos eixos X e Y.

Mão na massa O primeiro passo a realizar é baixar alguns arquivos de imagem para o jogo. Essas imagens representarão os atores, mapa do jogo, pontos *etc.* Acesse o link para baixar os arquivos: http://bit.ly/2kg0Z0o. Após baixar e descompactar, a pasta terá o nome de Arquivos Scratch.

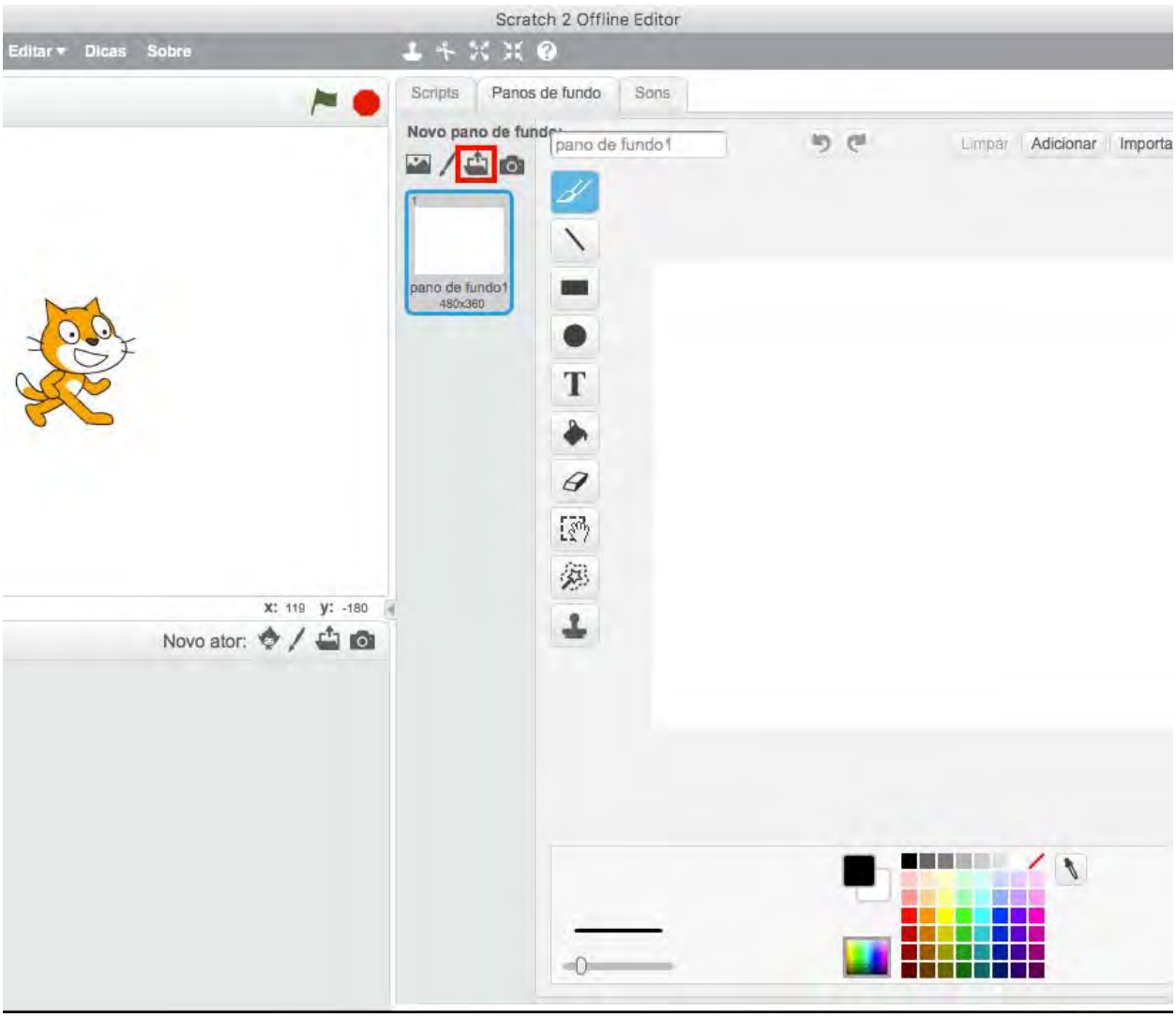
1. Abra o Scratch no seu computador ou a versão online. Se você não se lembra como, reveja o primeiro capítulo.

2. Para o nosso jogo começar a ter uma forma, vamos adicionar o mapa do labirinto. Com o Scratch aberto, clique em Palco.

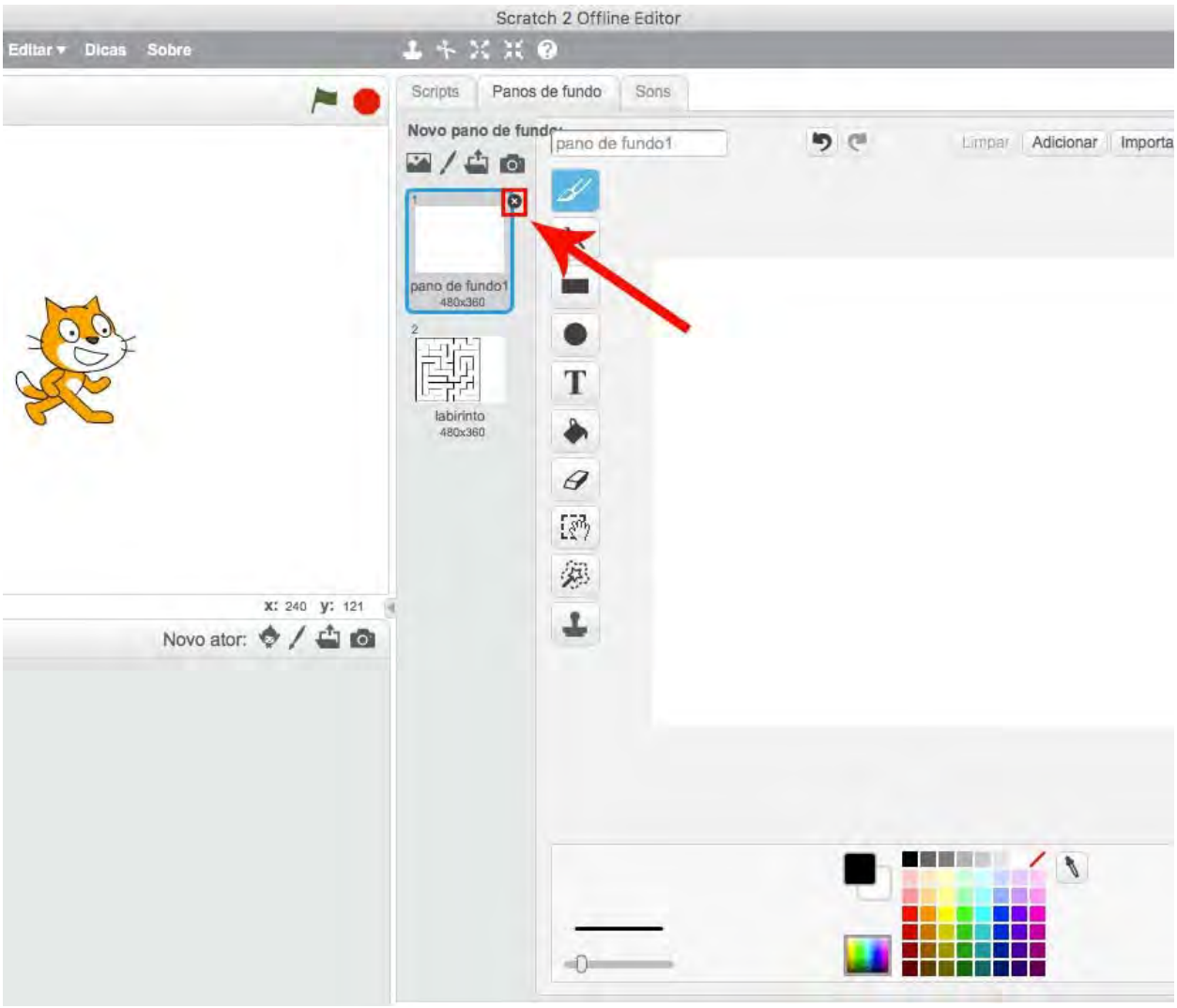


3. Após, clique em Panos de fundo. Aqui é onde ficam armazenadas todas as imagens que aparecem no Palco.

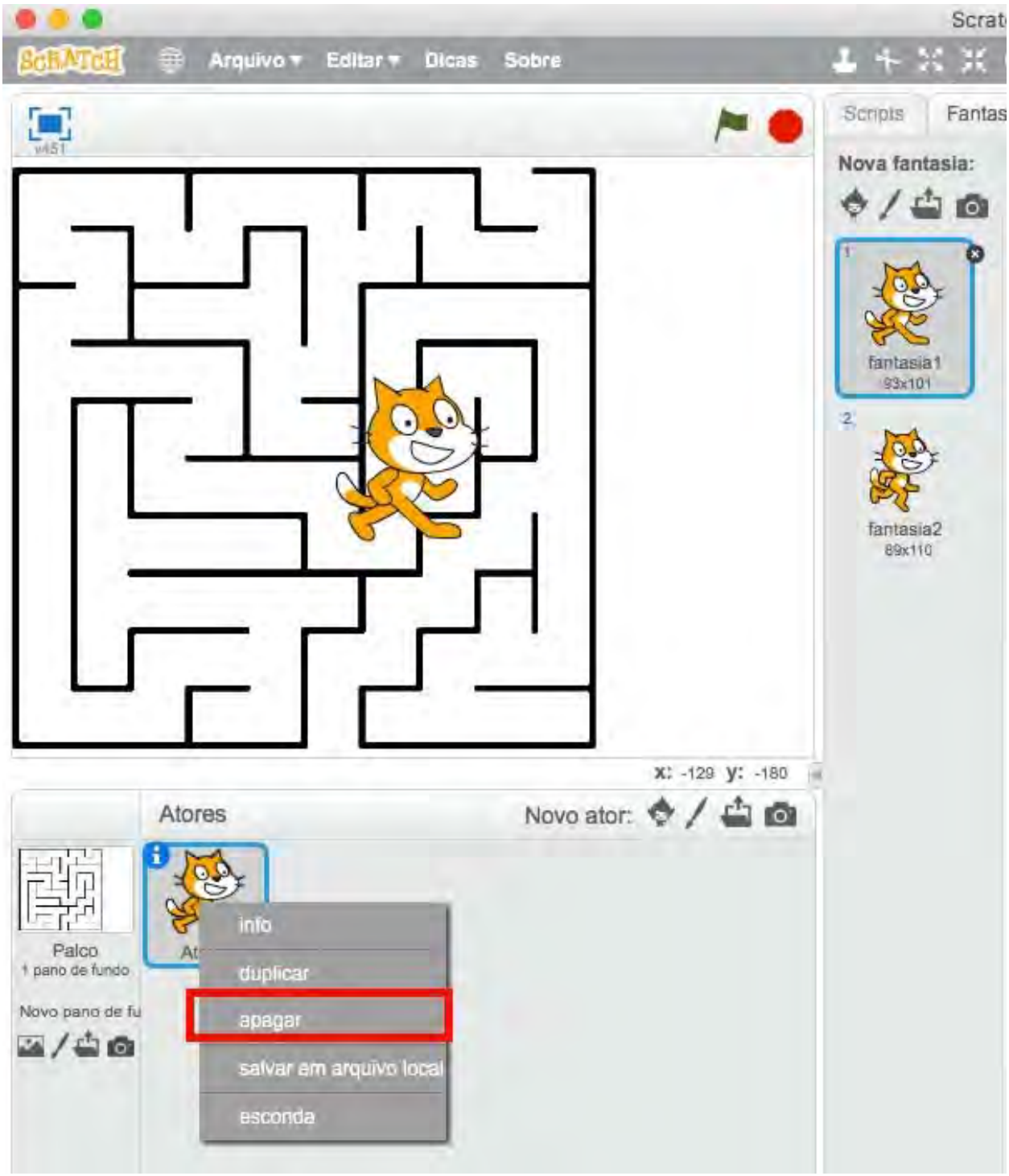
4. Vamos adicionar uma nova imagem de pano de fundo que será o labirinto. Clique em Carregar cenário a partir de arquivo e navegue até a pasta onde se encontram as imagens. Então, carregue a imagem labirinto.png.



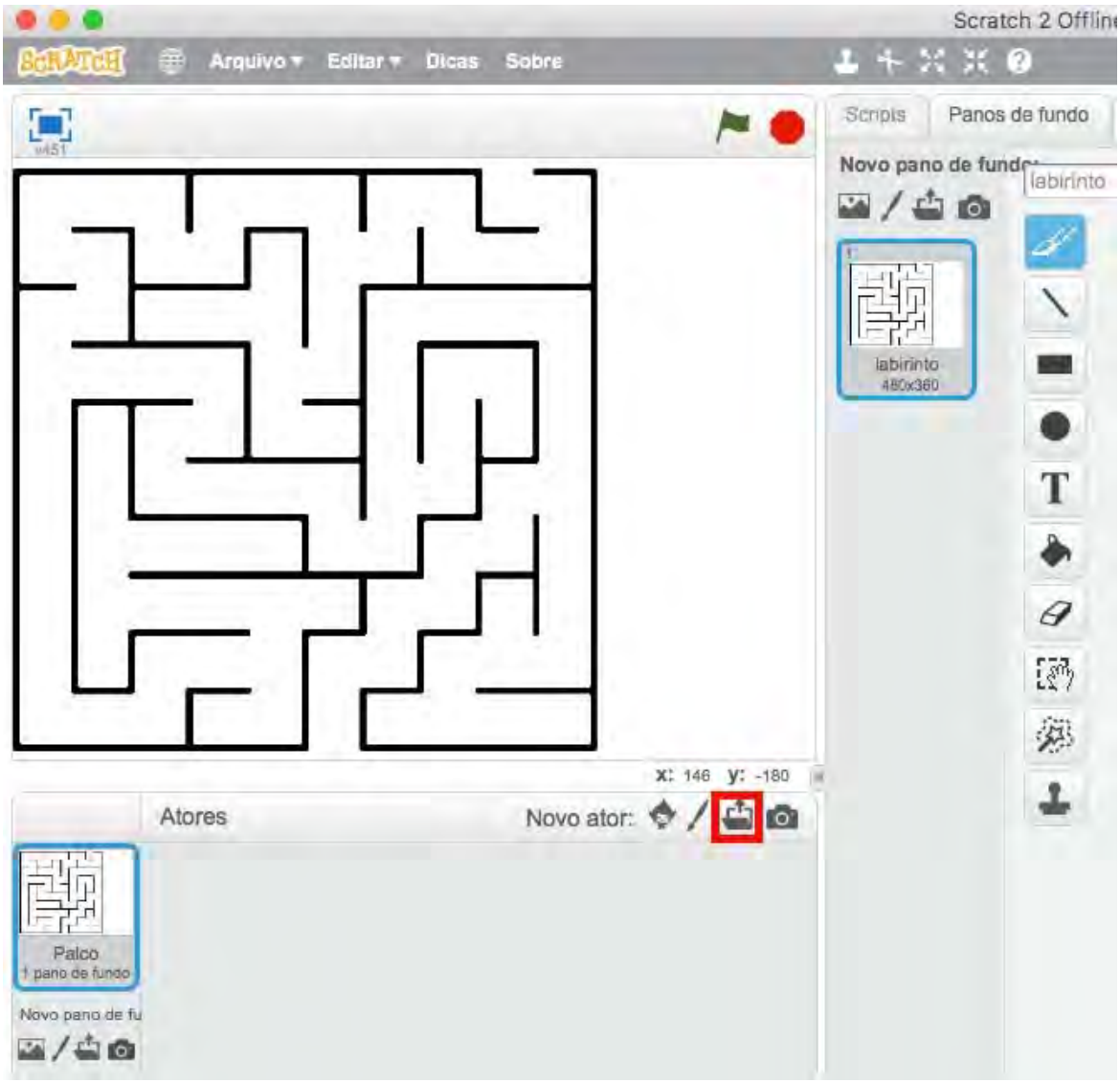
5. Veja que agora temos dois Panos de fundo: o primeiro totalmente em branco e o nosso labirinto. Precisamos apagar o primeiro; para isso, dê um clique sobre ele e clique no botão em X.



6. Vamos também apagar nosso ator padrão, pois adicionaremos um outro personagem. Para isso, dê um clique com o botão direito do mouse sobre o ator e clique em apagar.

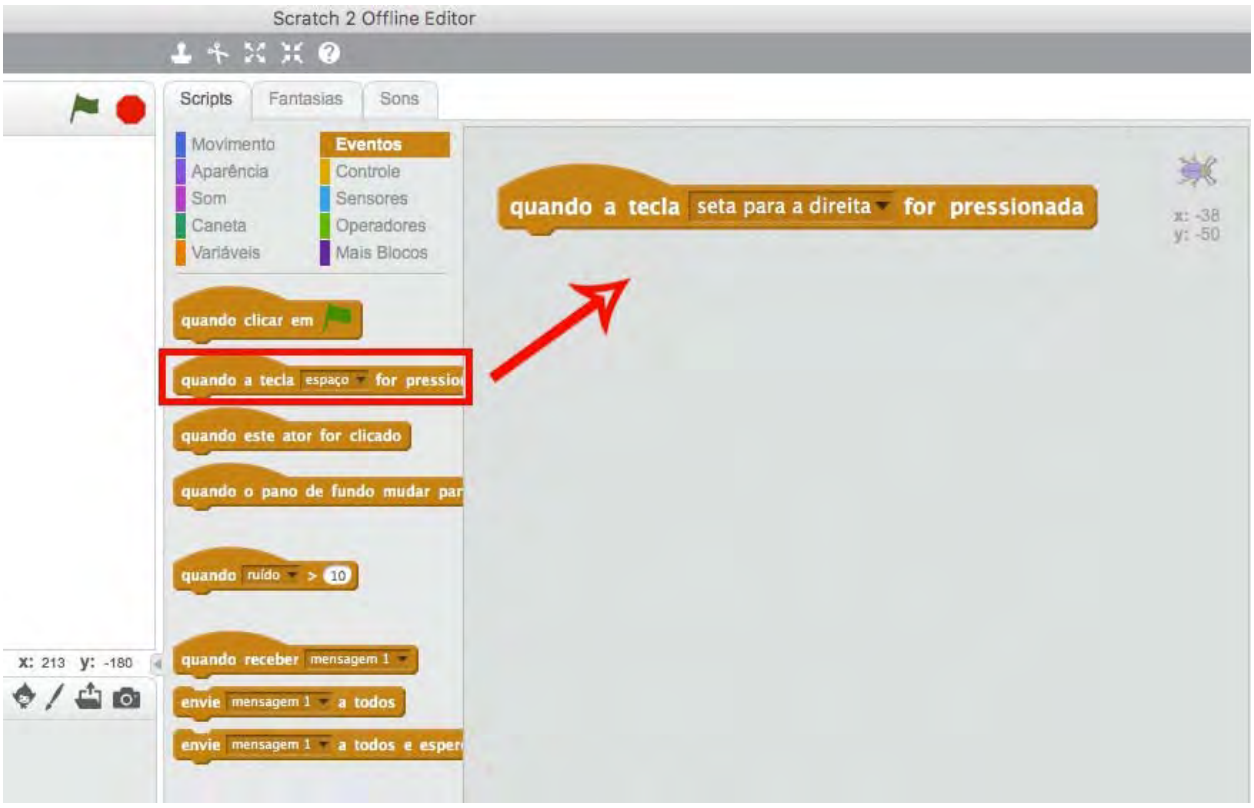


7. Gostaria que você conhecesse a baratinha. Ela será o personagem que o jogador vai comandar. Para adicionar um novo ator, clique em Carregar ator a partir de arquivo. Navegue até a pasta de imagens e carregue a imagem barataMenor.png.

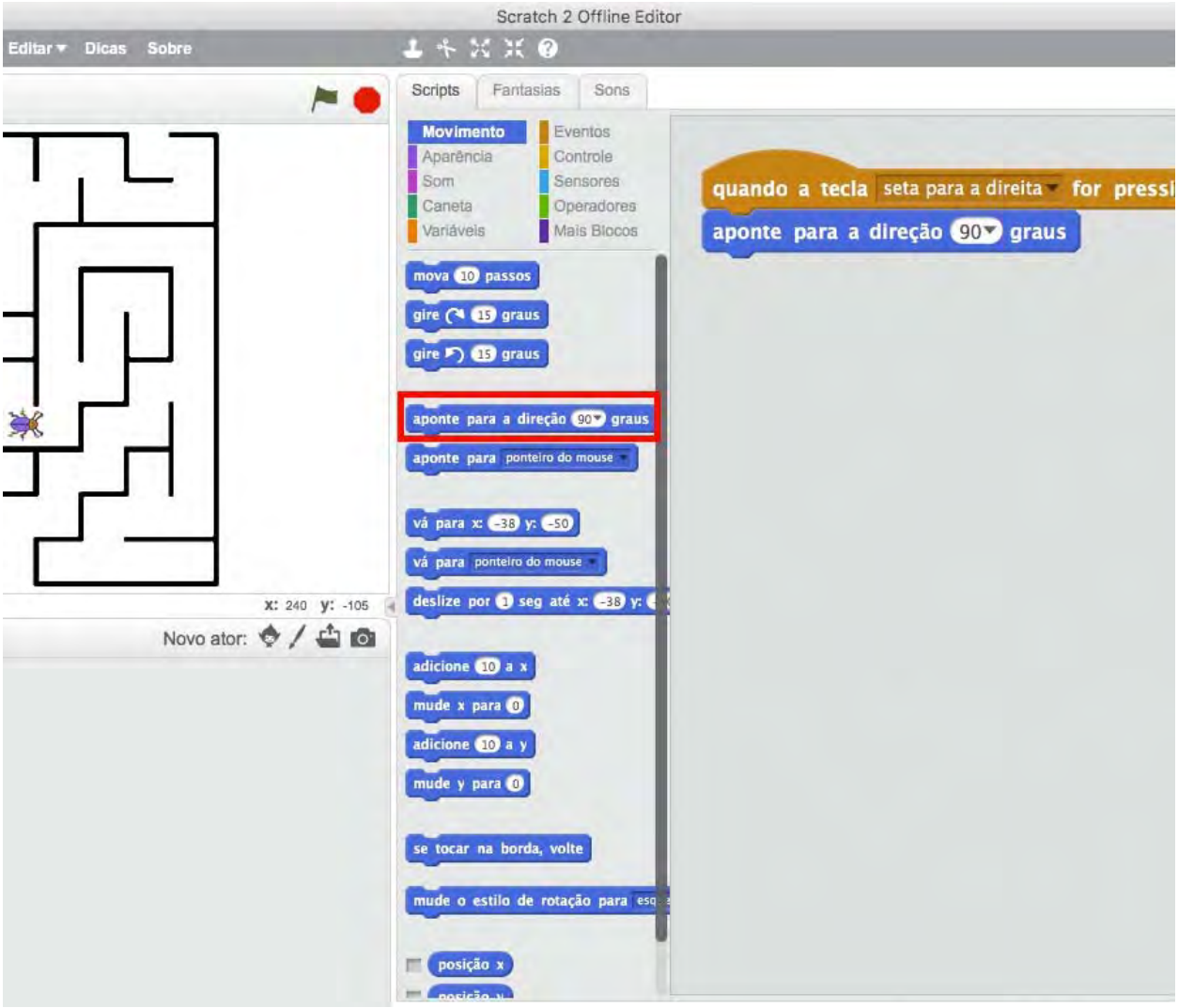


A partir de agora, vamos começar uma parte interessantíssima. Vamos programar a movimentação da barata. Como funciona? Cada vez que for pressionada uma determinada tecla, como por exemplo, as setinhas do teclado, o ator vai andar em alguma direção.

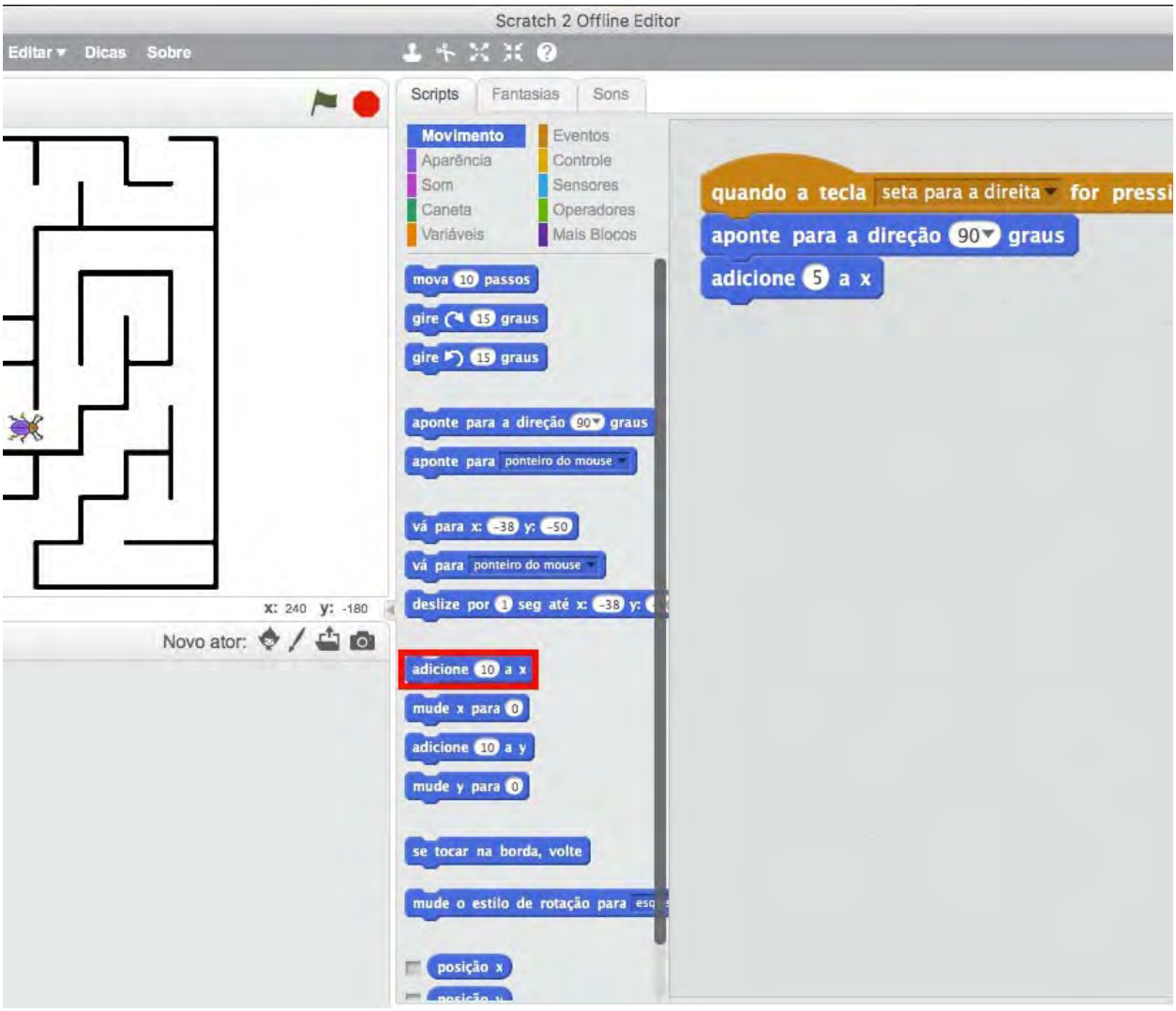
1. Clique em Eventos. Os eventos são os responsáveis por receber o estímulo para que algo aconteça. Veja que temos vários deles e utilizaremos um evento para que, ao pressionarmos as teclas de seta, o personagem se movimente. Então, o estímulo é: pressionar a seta para direita, esquerda, cima ou baixo.
2. Clique e arraste para a Área de scripts o evento: *quando a tecla espaço for pressionada*. Altere para *seta para direita*.



3. Clique em Movimento e arraste o bloco *aponte para a direção 90 graus*. Este bloco vai fazer com que cada vez que a tecla seta para a direita for pressionada, a baratinha aponta para a direita, ou seja, girando 90 graus.



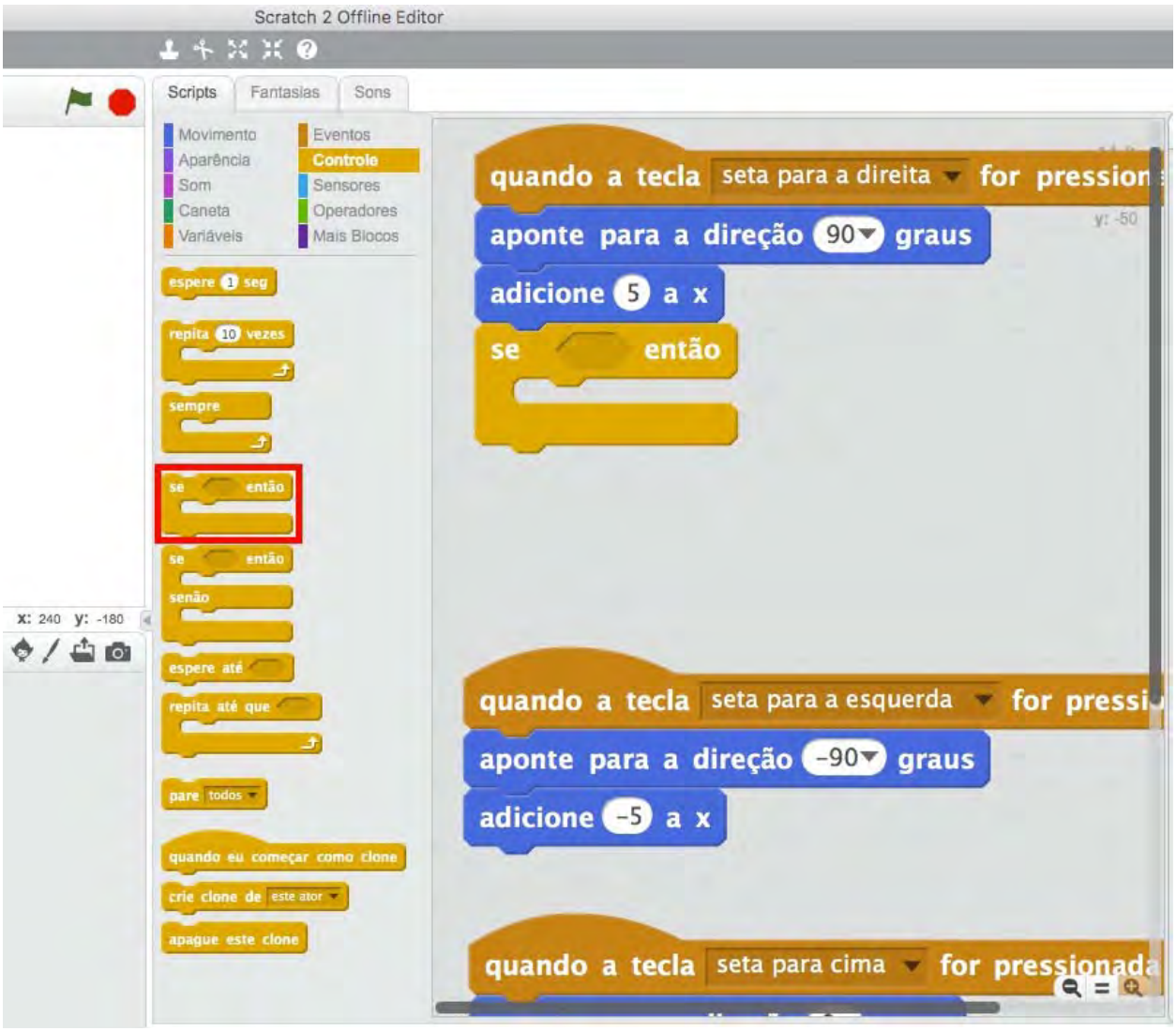
4. Arraste também o bloco Adicione 10 a x. Este bloco é responsável por fazer com que o personagem se movimente em uma certa quantidade no eixo X. Lembra do plano cartesiano? Pois então, aqui estamos dizendo que, cada vez que a tecla seta para a direita for pressionada, a baratinha vai andar 5 unidades no eixo X. Lembre-se de que o eixo X vai de -240 a 240 e o eixo Y de -180 a 180.



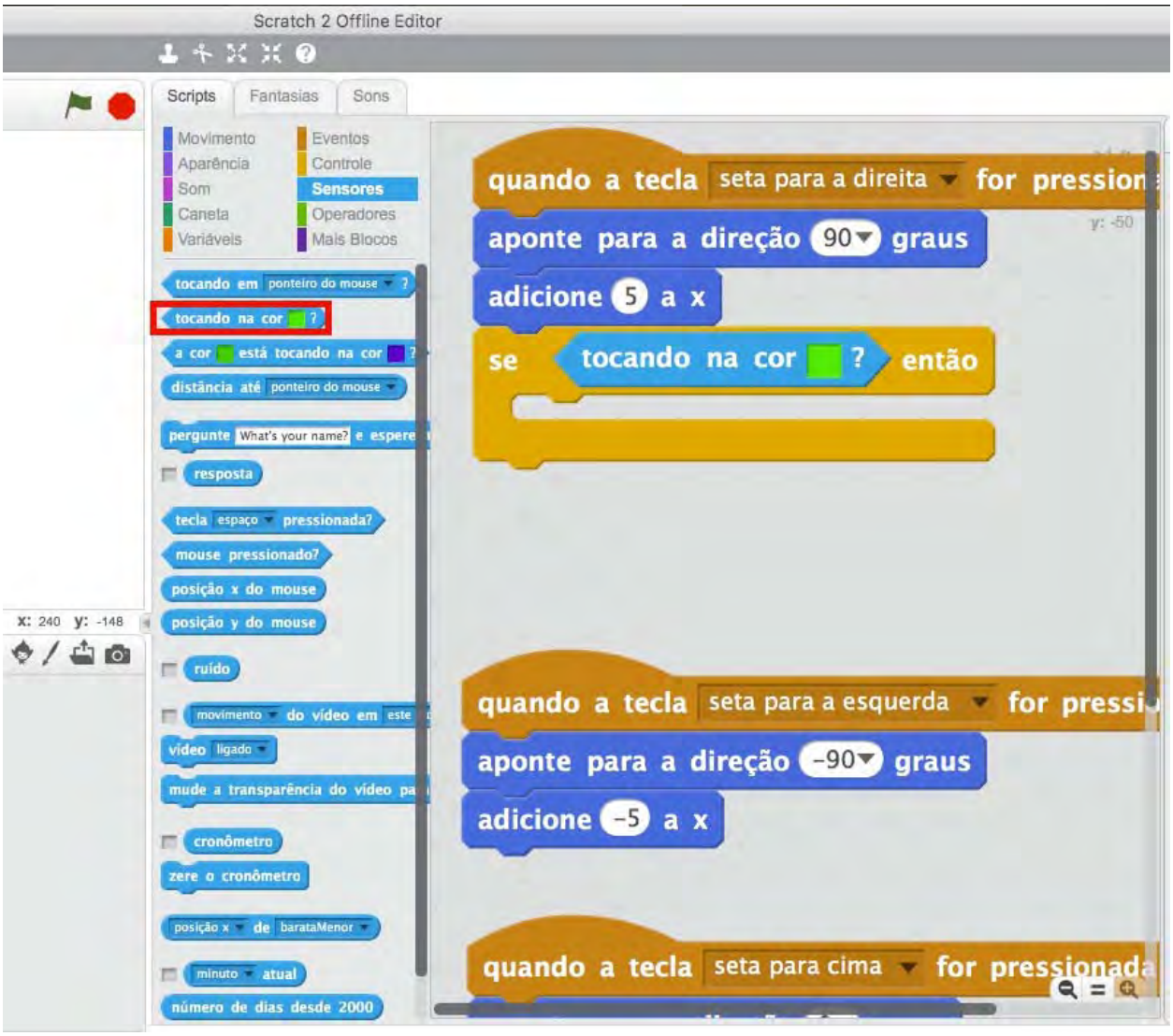
5. Altere o valor para 5. Vamos definir a movimentação para apenas 5, porque se colocarmos uma velocidade maior, nosso personagem vai travar em alguns cantos e não conseguir sair nunca mais. Isso acontece porque ele vai tentar ir além do permitido pelas paredes do labirinto, e nisso ele não vai nem conseguir sair do lugar.
6. Repita para todas as direções os passos anteriores e lembre-se de alterar os valores correspondente ao eixo X e Y.

Se você testar a movimentação, perceberá que ele está atravessando as paredes. Para que isso não ocorra, temos de verificar através de um bloco **"Se então"** se o personagem está tocando na parede.

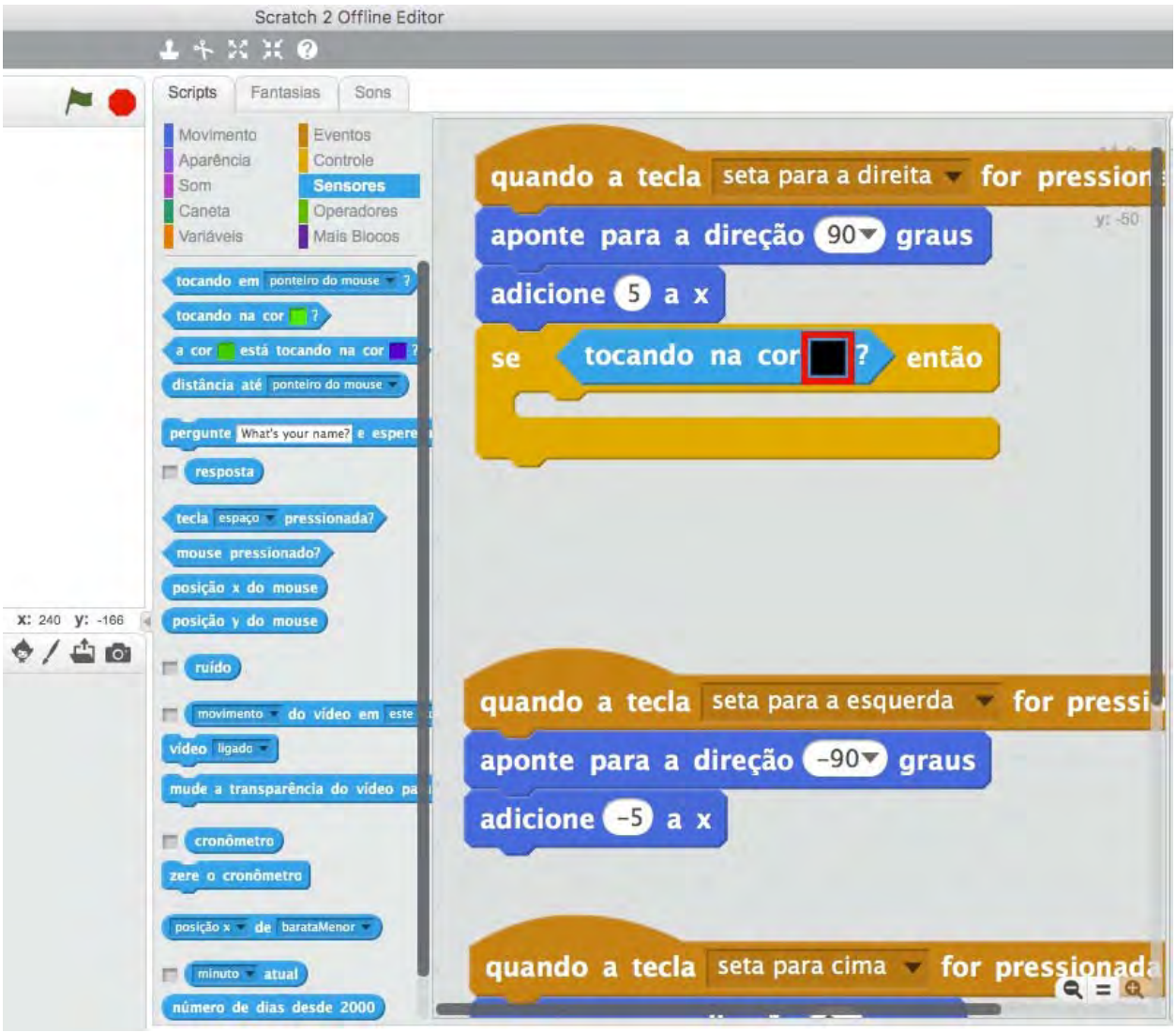
1. Clique em Controle e arraste o bloco Se então.



2. Clique em Sensores e arraste o bloco tocando na cor ? para dentro da área hexagonal do bloco Se então. Veja o formato do bloco, isso significa que ele sempre retornará um valor booleano, ou seja, verdadeiro ou falso.



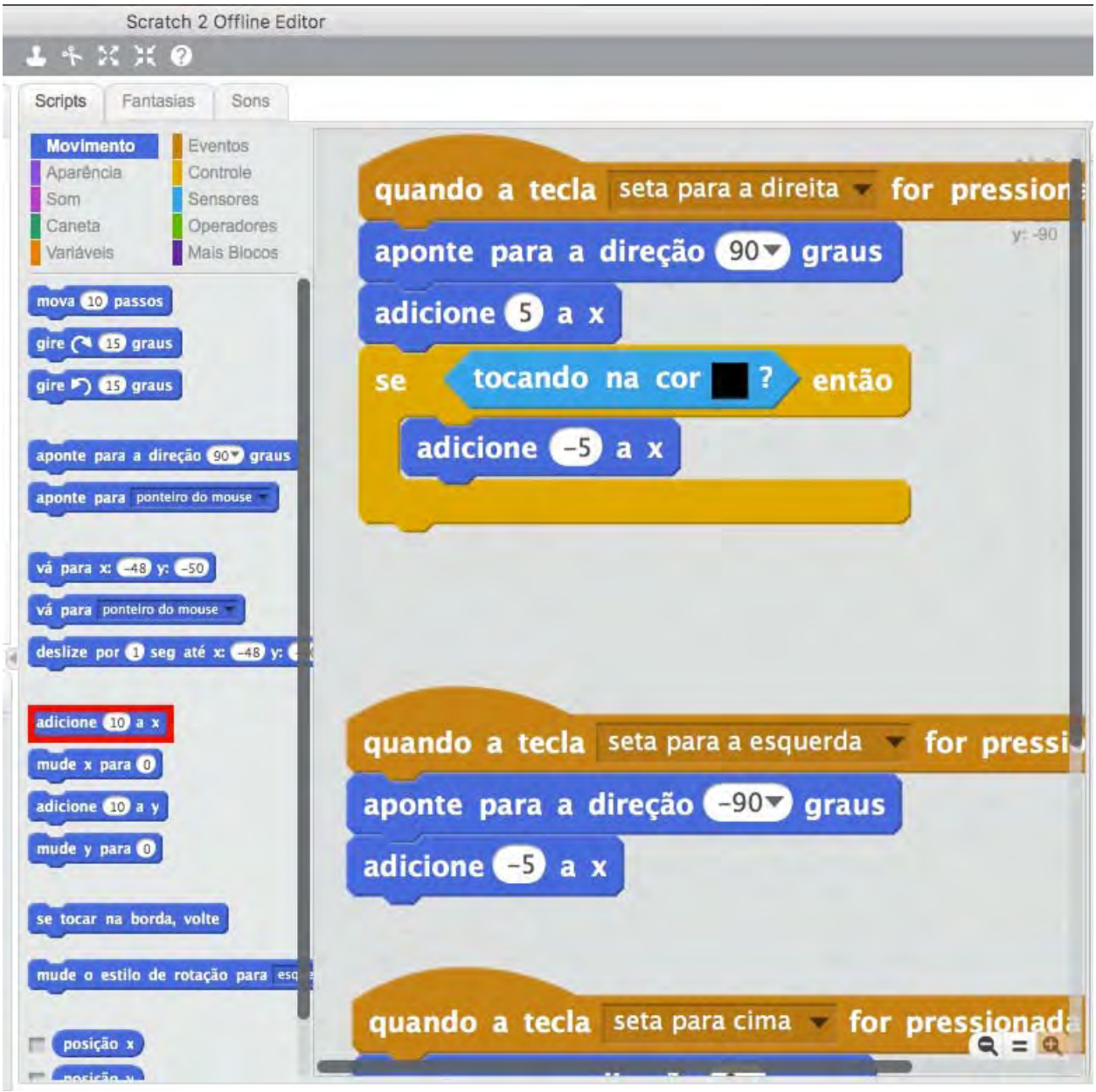
3. Clique no quadrado que contém a cor e, depois, selecione a parede do labirinto contendo a cor preta.



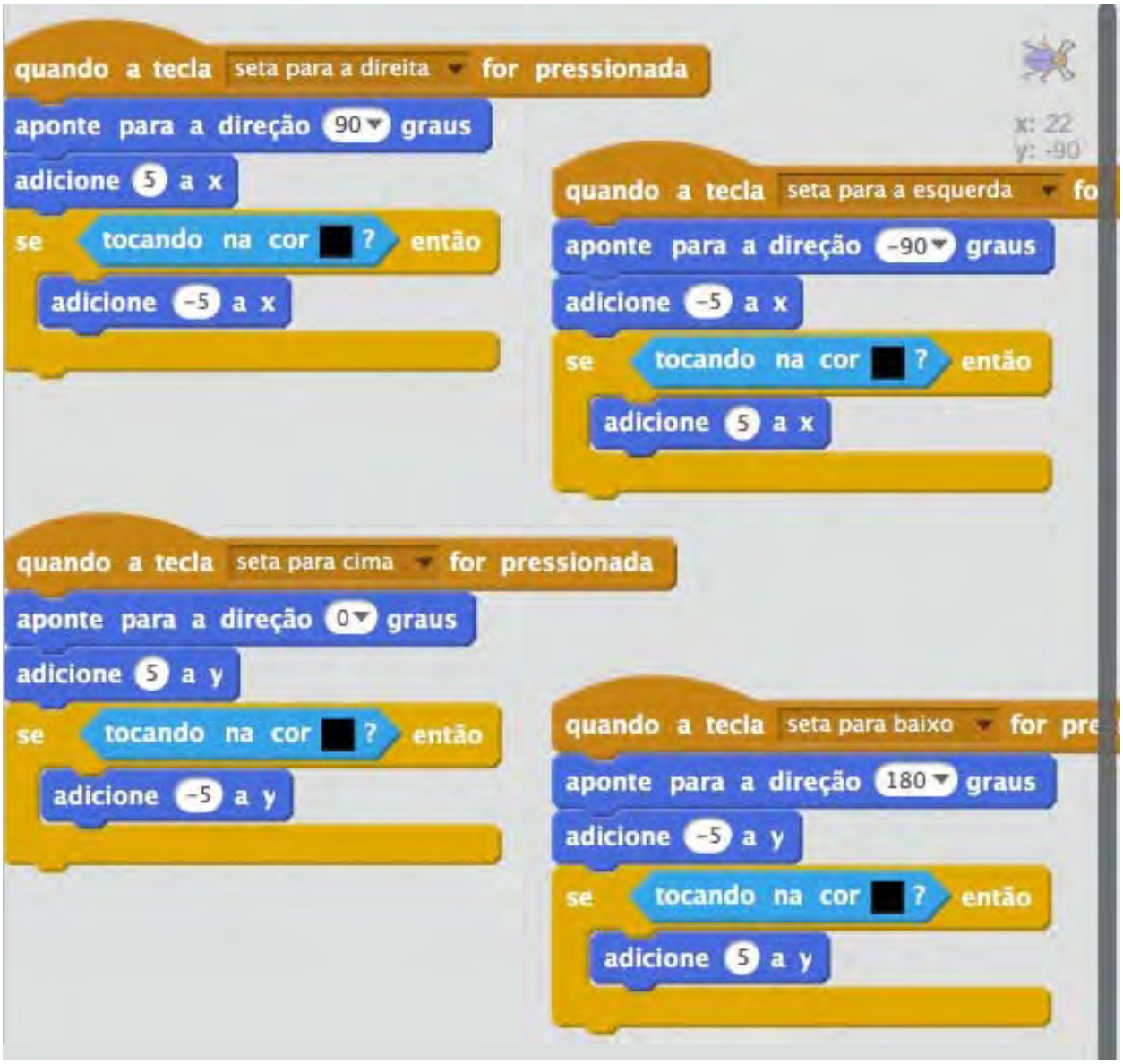
ATENÇÃO

Tente selecionar o mais preto possível, pois se a cor ficar um pouco cinza, teremos problema na verificação da cor e o personagem poderá atravessar a parede em alguns momentos.

1. Clique em Movimento e arraste o bloco adicione 10 a x para dentro do bloco Se então. Muitos blocos vêm com valores definidos, no entanto, você pode alterá-los. Por exemplo, caso deseje aumentar ou diminuir a velocidade do ator, é este bloco que você deverá modificar.



2. Altere o valor para a forma negativa da movimentação que ele está fazendo. Por exemplo, caso ele esteja andando 5 no eixo X, ao tocar na parede do labirinto, ele terá de andar -5 no eixo X, pois isso anulará o movimento dele, fazendo com que ele não atravesse.
3. Após isso, repita os passos anteriores para todas as direções e altere conforme for necessário.



Vamos tentar entender melhor o código através desta sequência:

1. Quando uma tecla for pressionada;
2. O personagem vai girar e apontar em uma direção, ou seja, na direção que ele vai andar;
3. Então, ele vai se movimentar uma certa quantidade de unidades;
4. Com o bloco Se então, ele vai verificar "se está tocando na cor preta". Caso seja verdadeira a afirmação, ele vai se movimentar no sentido contrário para anular o movimento.

Bem simples, não? Teste o seu jogo! Brinque um pouco com ele e faça algumas modificações. Após isso, salve-o, pois usaremos no próximo capítulo! :) **2.4 Conclusão**

- Neste segundo capítulo, vimos os comandos Se e Senão para tomada de decisão;
- Vimos os operadores de comparação que permitem comparar dois valores;
- Vimos os operadores lógicos que permitem comparar entre Verdadeiro ou Falso;
- Começamos a desenvolver o jogo de labirinto.

No próximo capítulo, veremos uma estrutura de controle chamada **Repetição**. Essa estrutura é fundamental para realizar repetições infinitas ou limitadas por uma condição. Também estaremos conhecendo outras ferramentas do Scratch e continuando a segunda parte do labirinto.

CAPÍTULO 3

Repetição

3.1 Introdução

Após visto o comando para tomada de decisões, veremos agora o comando de repetição que nos permite executar passos mais de uma vez, ou até que uma condição seja atendida. Esta ferramenta é tão simples quanto o Se e o Senão.

A seguir, veremos três variações do comando de repetição. Quando precisamos executar um trecho de código repetidamente, certamente usaremos algum comando de repetição. Existem três tipos de repetição: *repita x vezes*, *repita até que*, *sempre*.

Scratch	Definição
	Repete até a quantidade de vezes estipulada.
	Repete até que uma condição seja atendida.
	Repete infinitamente.

3.2 Blocos de repetição

Pseudocódigo

```
enquanto (x>10)
faça
    x = x + 1
fim_enquanto
```

Linguagem C

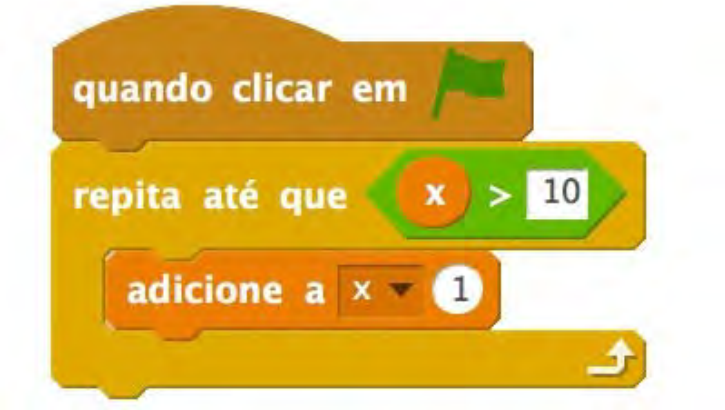
```
while(x>10){
 x++;
}
```

Ou:

```
for(i=0; i<=10; i++){
 x++;
}
```

Os três exemplos realizam a mesma operação de repetição, que é adicionar o valor 1 a uma variável. Não se preocupe agora com variáveis, estaremos estudando-as no *capítulo 4*, mas atente-se à estrutura de repetição mostrada nos exemplos.

Veja que em todos os exemplos há uma condição e, entre {}, há o trecho de código que se repetirá. Veja agora na linguagem Scratch: **Scracth**



Tudo o que estiver dentro do bloco *repita até que* será repetido a quantidade de vezes estipulada. Veja que estamos utilizando uma variável adicione a x 1, a cada repetição será adicionado o valor 1 a X, sendo incrementado até 10. Quando o X tiver um valor maior do que 10, ou seja, quando a condição for falsa, a repetição para e finaliza o script.

Você deve estar se perguntando: qual a utilidade deste comando e quando utilizá-lo. Um bom uso do comando de repetição seria, por exemplo, em uma sala de aula que tem muitos

alunos, vamos supor 40 alunos. Imagine que você queira somar as notas de todos esses alunos, e depois encontrar a média da turma.

Você precisaria repetir códigos que armazenariam as notas para cada um dos alunos, o que acabaria criando um código gigantesco e de difícil alteração. E se, em vez disso, criarmos apenas um trecho de código que pega a nota de um aluno, e depois repete para o restante. Bem mais fácil, não? Pois é isso que o comando de repetição faz.

3.3 Labirinto

Para entender melhor, vamos partir direto para a prática. A partir de agora, continuaremos o projeto labirinto e inseriremos um novo ator que representará uma bolinha que o jogador deverá capturar. Para isso, trabalharemos com alguns comandos de repetição.

Abra o Scratch e carregue o arquivo salvo do capítulo anterior.

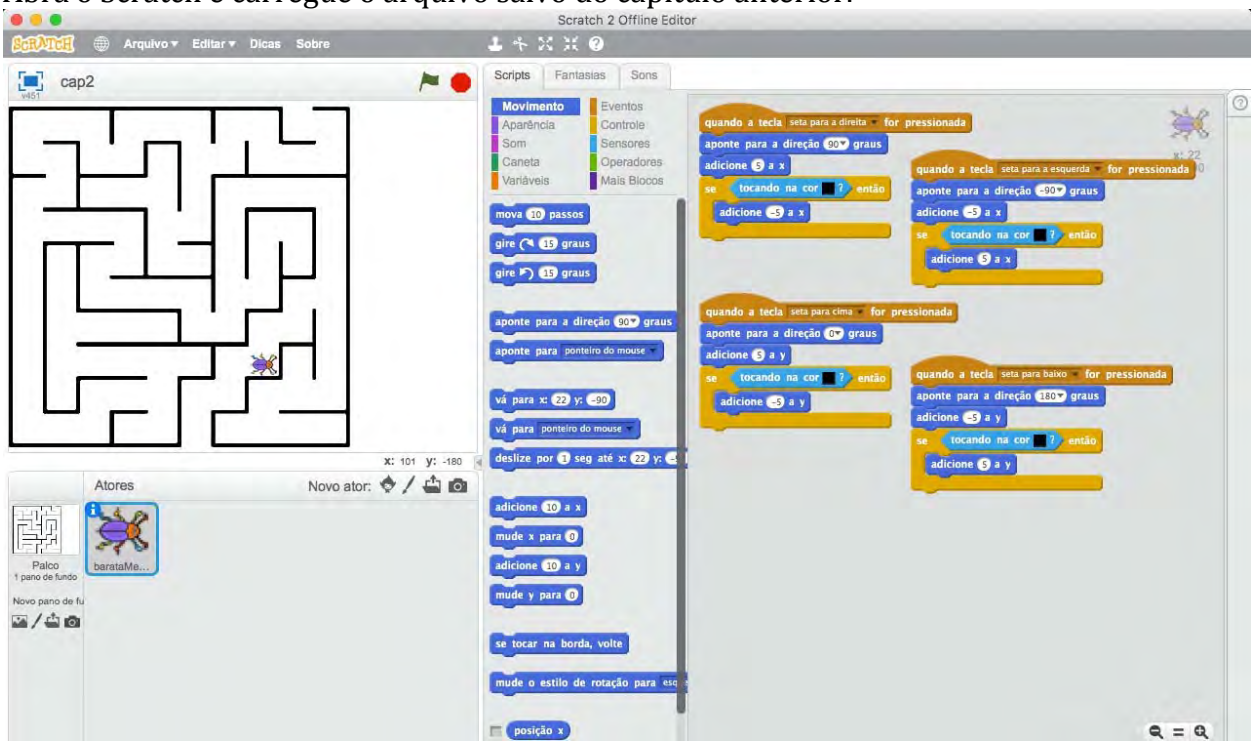
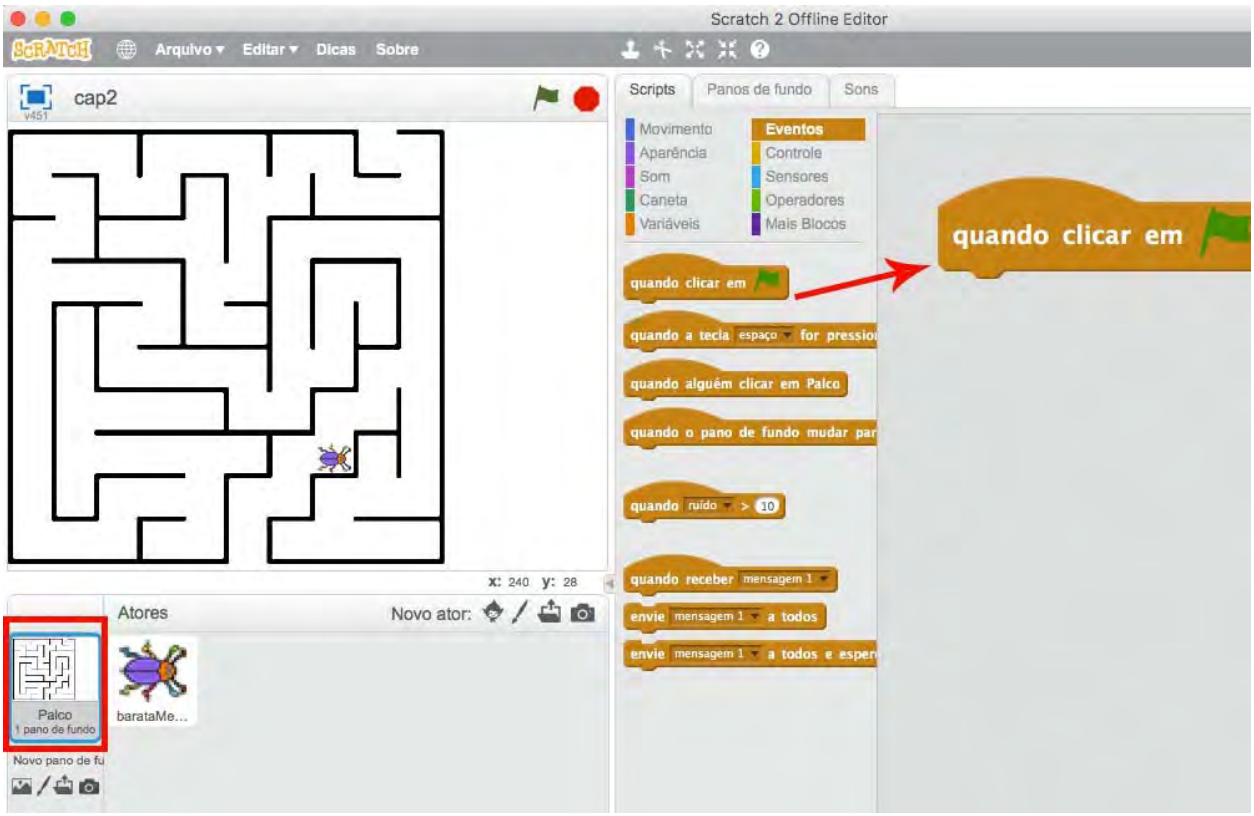


Figura 3.3: Sua interface deverá ser algo parecido com isso

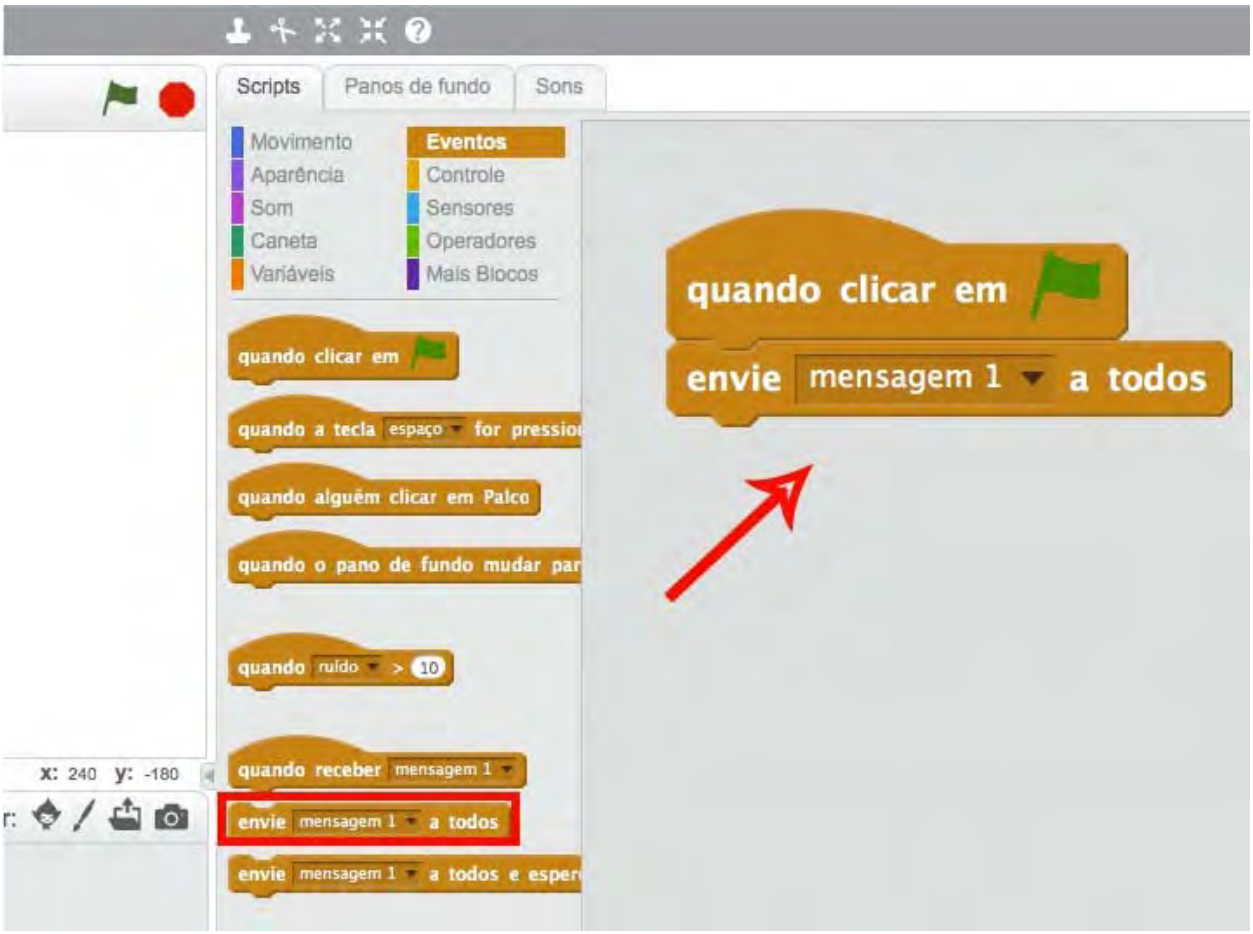
Enviando mensagens Neste momento, começaremos a trabalhar com *mensagens*. As mensagens são responsáveis por transmitir informações de um ator para outro. Dessa maneira, faremos com que, ao clicar na bandeira, seja enviada uma mensagem com os dizeres: iniciarJogo. Ou seja, estaremos enviando um recado a todos os atores informando que o jogo vai começar.

Podemos iniciar a execução de alguns scripts, como por exemplo, este que vamos implementar neste capítulo: haverá uma bolinha que o jogador terá de capturar. Esta bolinha só estará disponível para captura quando receber a mensagem iniciarJogo.

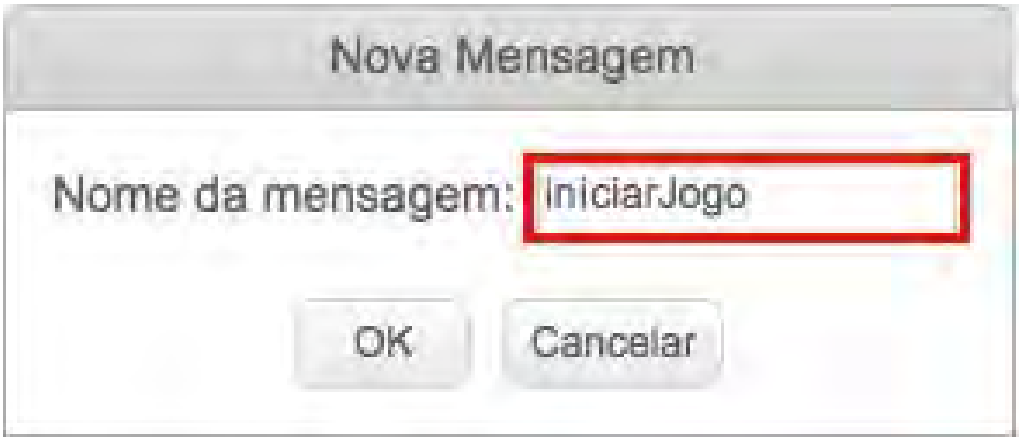
1. Clique em Palco. Para ativar o envio da mensagem, é necessário adicionar um evento. Portanto, vá em Eventos, e arraste o bloco Quando clicar em.



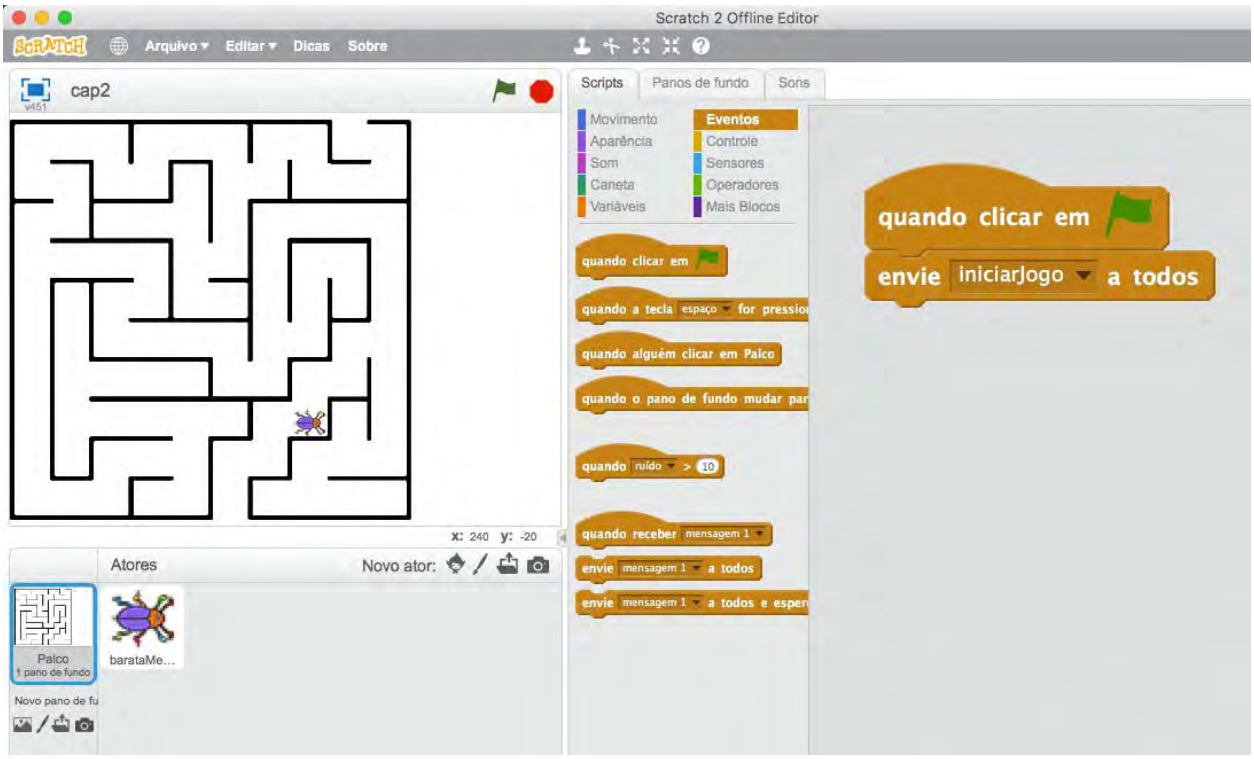
2. Em Eventos, arraste Envie mensagem 1 a todos para enviar a mensagem aos atores.



3. Para tornar o entendimento do código mais simples, vamos alterar a mensagem, pois *mensagem 1* não é um título muito intuitivo para saber o que está sendo enviado. Clique na setinha da *mensagem 1* e selecione nova mensagem para alterar a mensagem. Então, digite iniciarJogo.

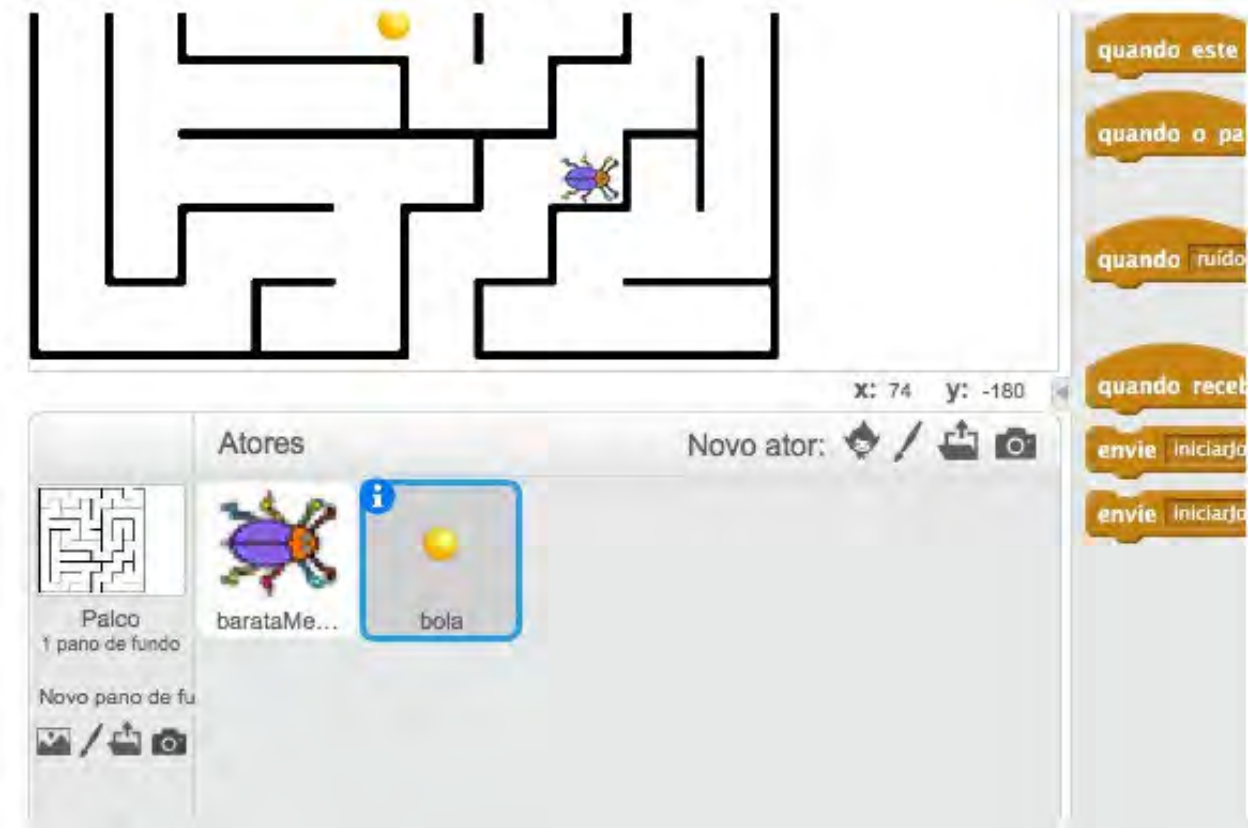


4. Veja se está igual a imagem seguinte:



5. Vamos adicionar um novo Ator que representará a bolinha que o jogador deve capturar. Clique no botão Carregar ator a partir de arquivo. Navegue até a pasta que você baixou com os arquivos do projeto e carregue a imagem bola.png.

6. Após adicionado, ficará assim:

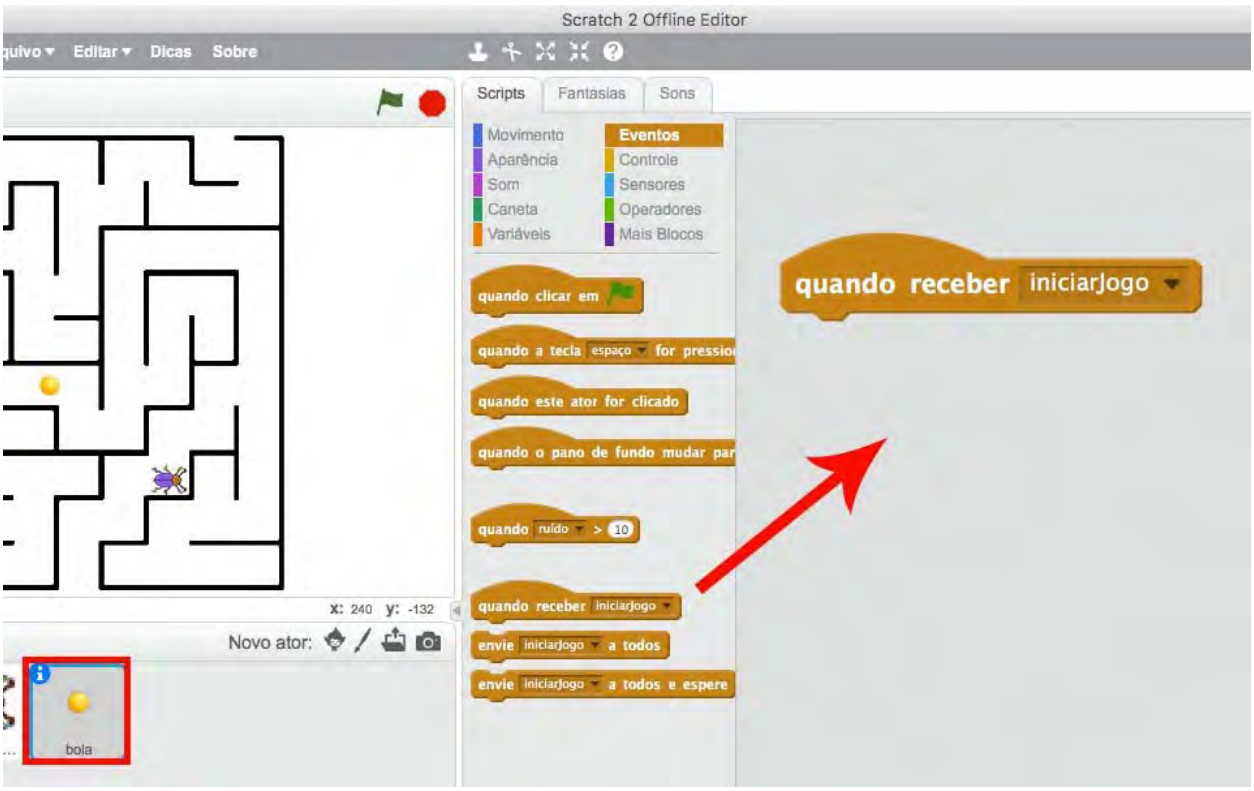


Capturando a bolinha A partir de agora, vamos implementar o sistema de captura da bolinha. Mas como funciona a lógica por trás da captura? Primeiro precisamos pensar no que deve acontecer para realizar a captura:

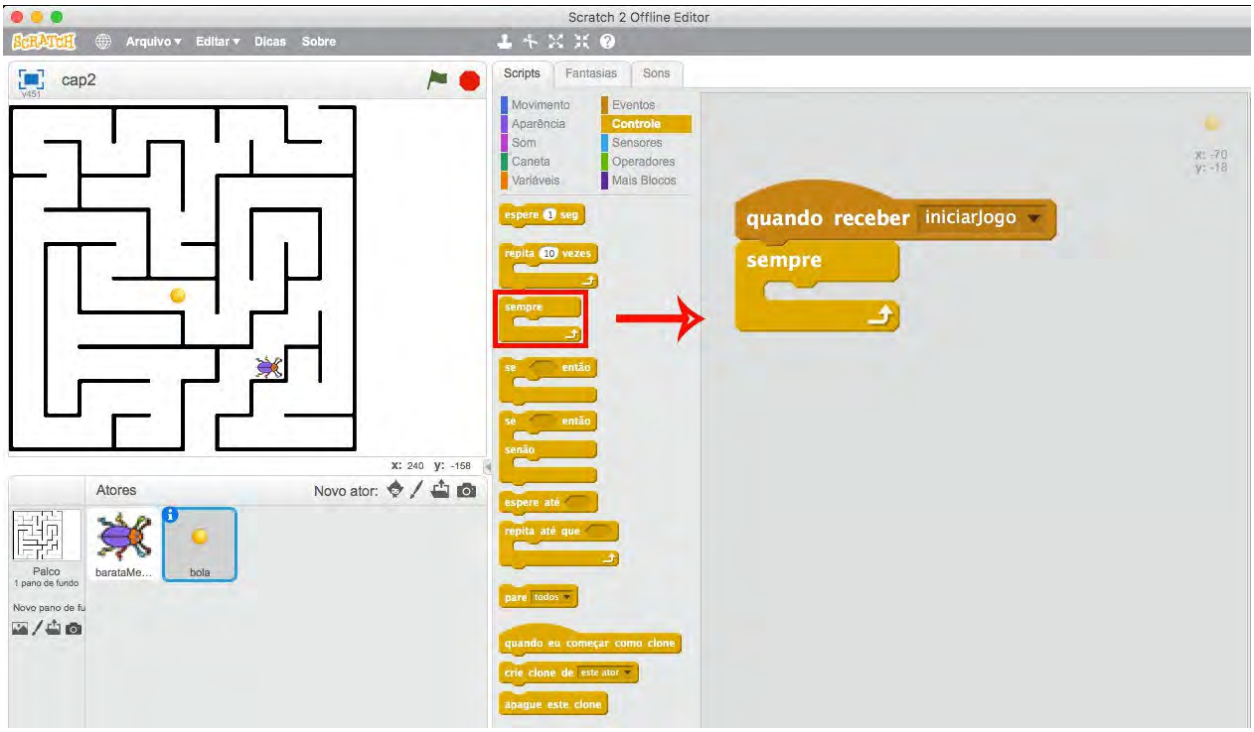
- Tocar na bola;
- Bola desaparecer;
- Aparecer em outro lugar;
- Adicionar 1 ponto para o jogador.

Veja que precisamos fazer com que o ator barataMenor encoste na bola. Em seguida, ela tem de desaparecer e aparecer em algum lugar aleatório, para depois adicionar 1 ponto ao jogador. Nesta última parte da pontuação, vamos implementar nos próximos capítulos, em que trabalharemos com algo chamado *variáveis* que é responsável por guardar dados, sejam eles números ou palavras.

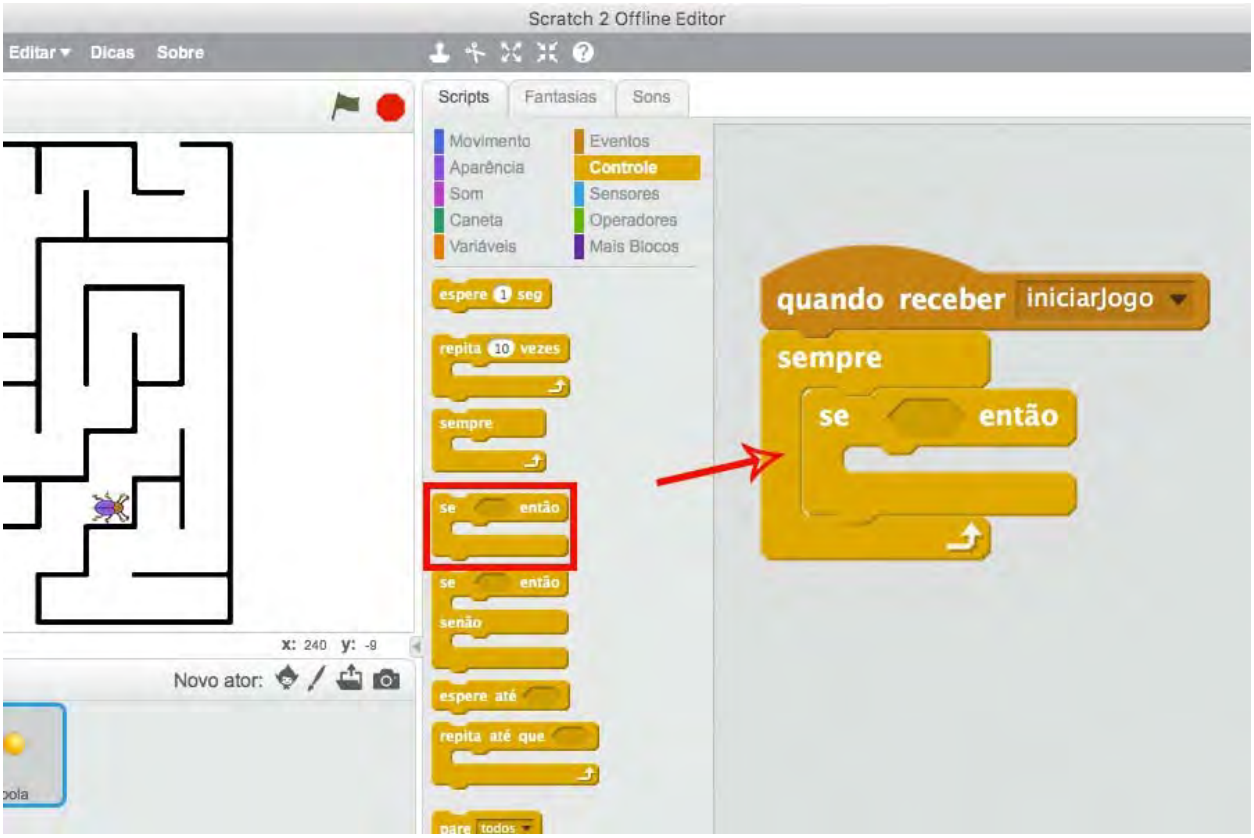
1. Com o ator bola selecionado, vá em Eventos e arraste o bloco Quando receber iniciarJogo para ficar sempre esperando um estímulo, ou seja, quando iniciar o jogo.



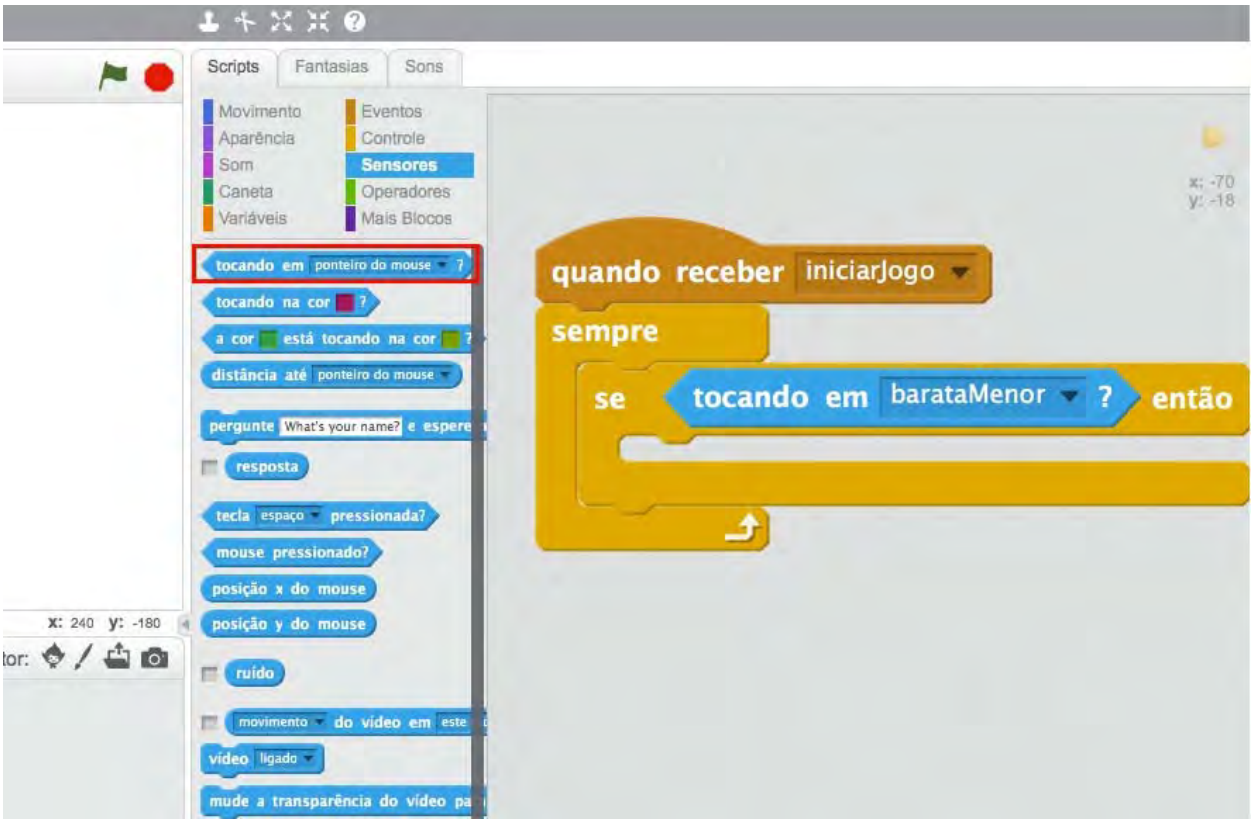
2. Agora em Controle, arraste o bloco Sempre para criar um loop infinito. Aqui estamos utilizando um bloco de repetição que é responsável por ficar infinitamente realizando uma verificação até o script ser interrompido.



3. Arraste o bloco Se então para verificar se o ator está tocando na bolinha ou não.



4. Agora em Sensores, arraste o bloco tocando em barataMenor. Veja que temos um bloco em formato hexagonal, isso significa que ele retornará um valor lógico, se estiver tocando (V) ou não (F).

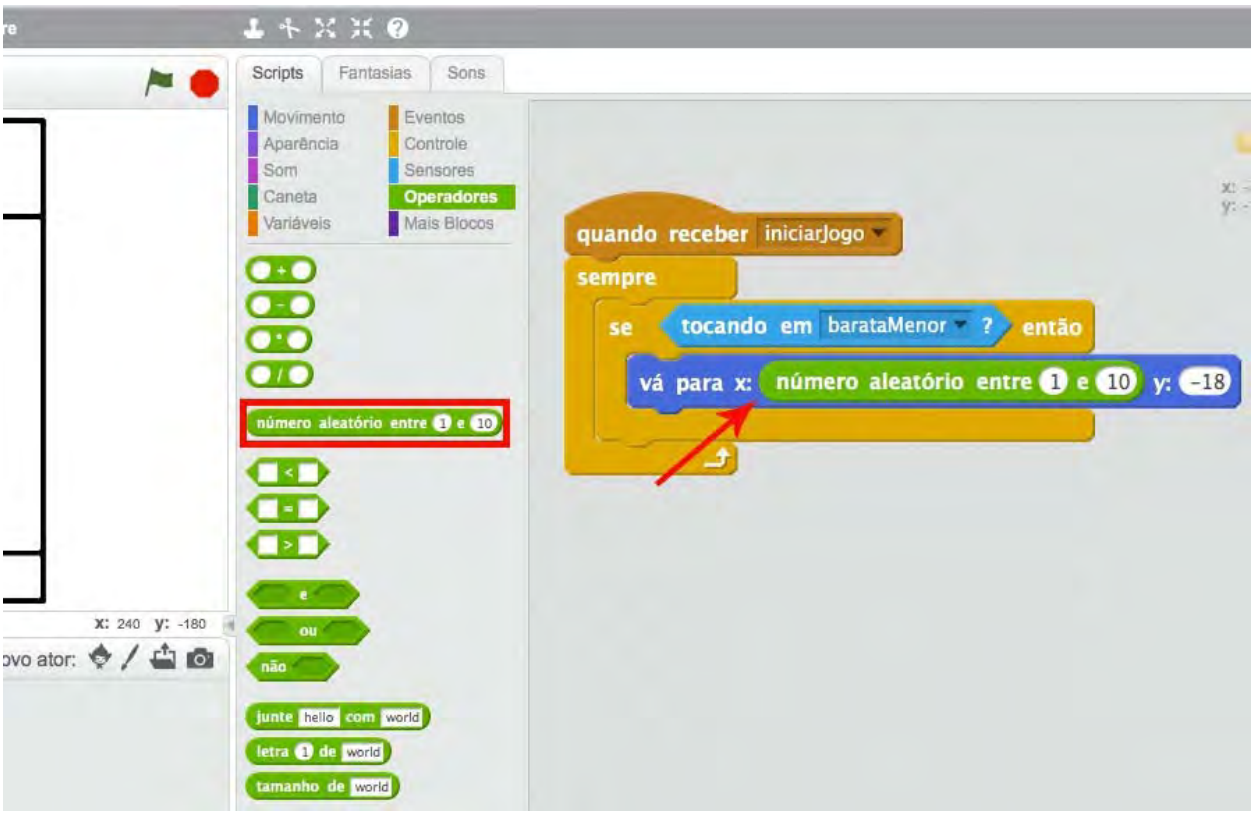


5. Em Movimento, arraste o bloco vá para x: y:, que vai definir uma posição nova para a bolinha após ser tocado pelo ator.

Este bloco é responsável por posicionar o ator em uma posição pré-definida pelos eixos X e Y. Você pode alterá-los se quiser, mas usaremos uma outra ferramenta a seguir.

Gerando números aleatórios

1. Em Operadores, arraste o bloco número aleatório entre. Este bloco gera números aleatórios nos limites definidos dentro dele e vai permitir que seja gerado uma posição nova e aleatória para a bolinha.



2. Coloque também o mesmo bloco para o eixo Y, ficando assim:

ATENÇÃO

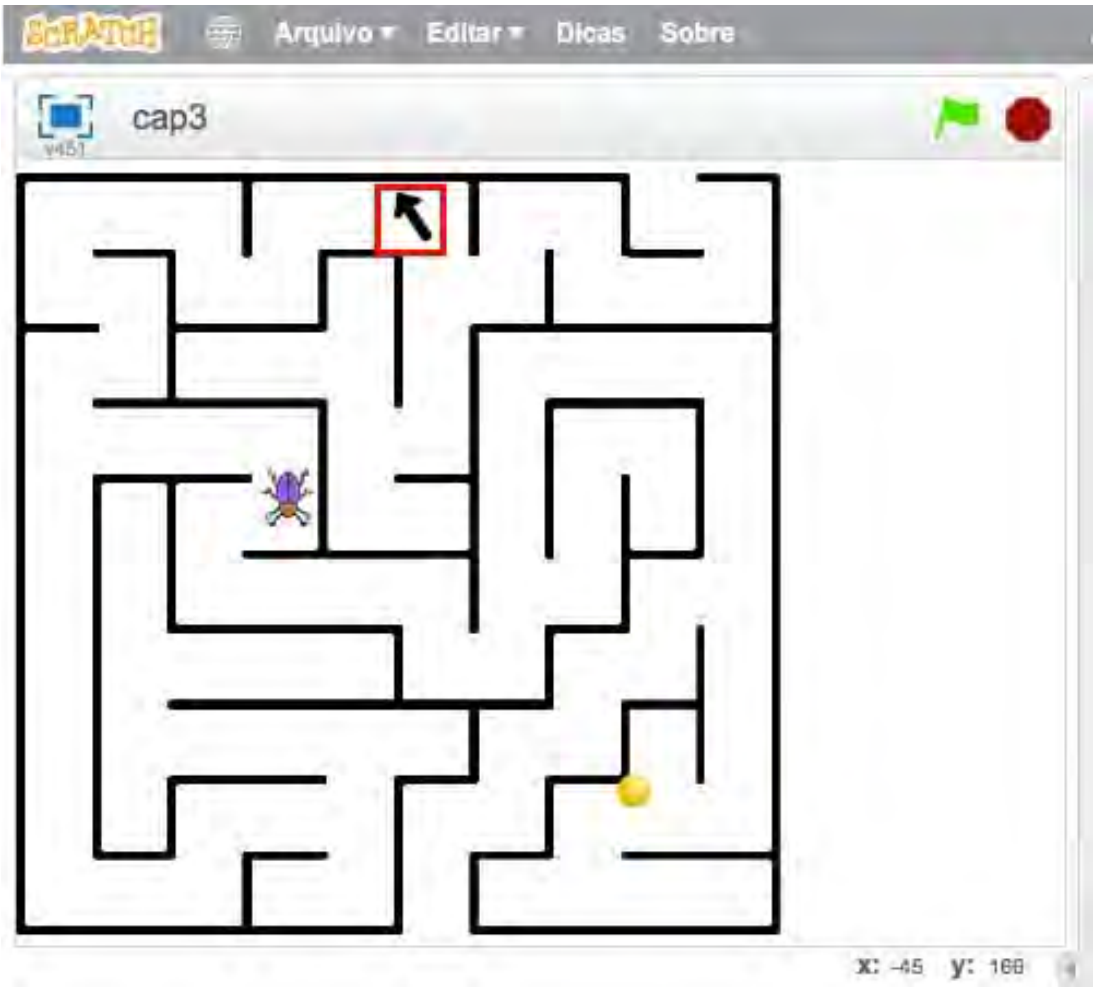
Este bloco gera um número aleatório entre os valores que nós definirmos. Vamos entender o que está acontecendo neste script.

Quando o ator receber a mensagem de que o jogo começou:

- Fique a todo o momento verificando (Se então) se o ator está tocando em barataMenor.

- Caso esteja tocando, vá para uma posição aleatória em X e Y.
- Esta posição aleatória é definida pelos limites do labirinto em X e Y. Veja que é definida pelo limite do labirinto e não do Palco.

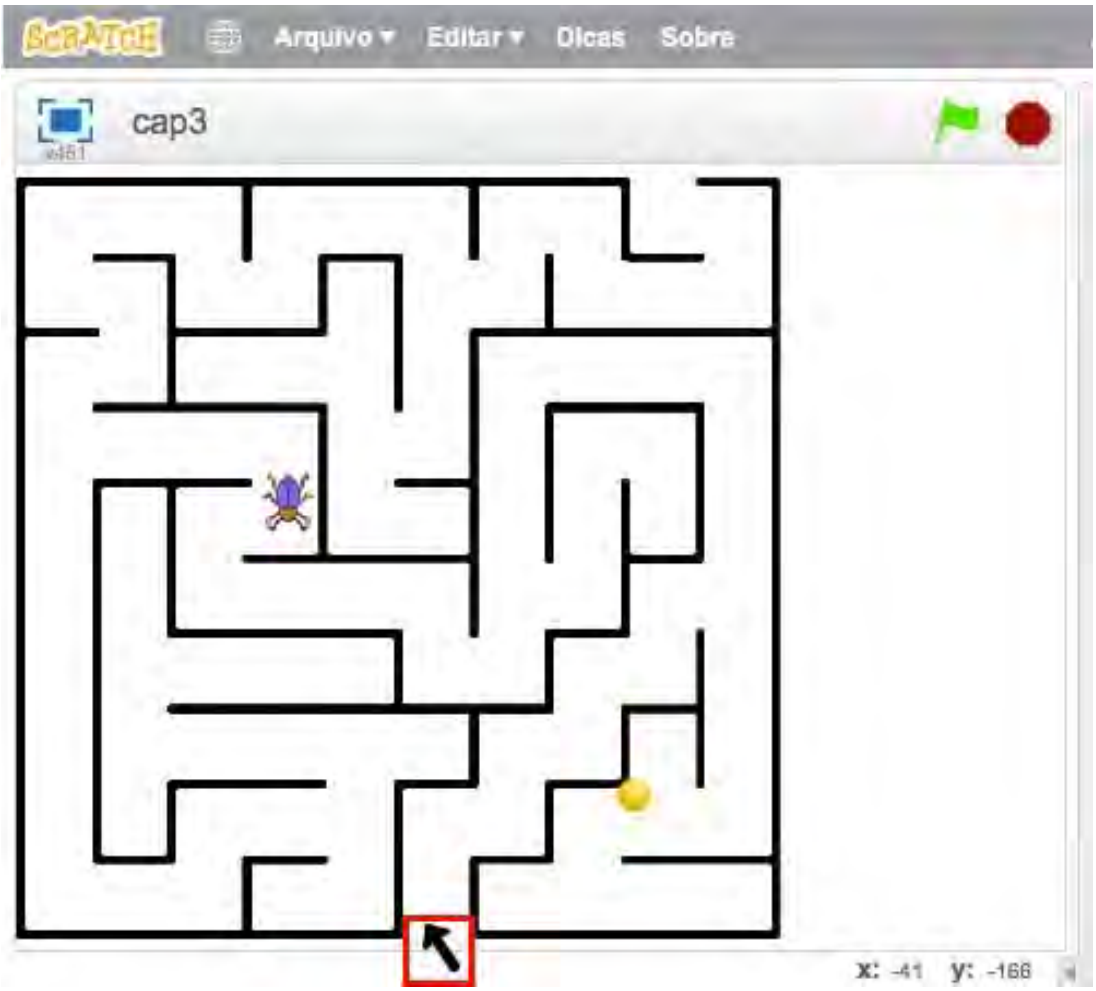
Para descobrir os limites, faça o seguinte:



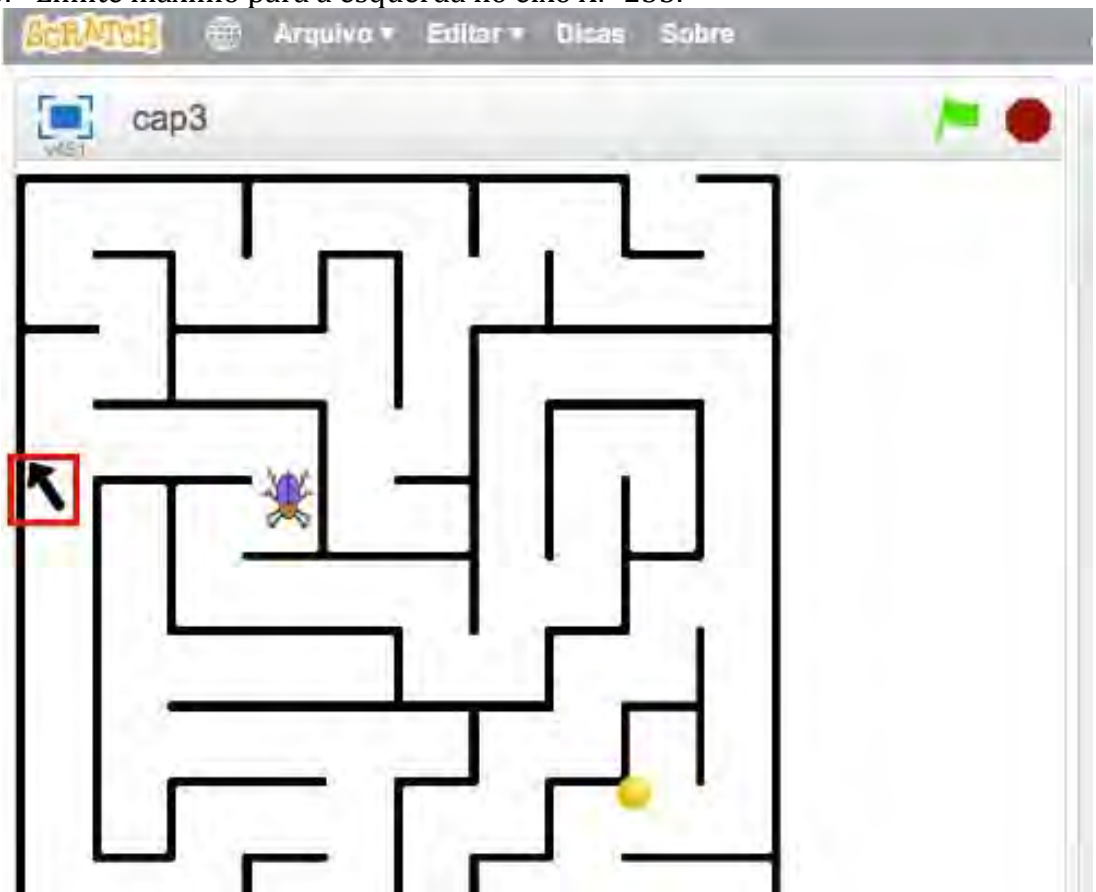
1. Posicione a setinha do mouse na parte superior do labirinto.

Observe na parte debaixo do Palco os valores X: -45 e Y: 166. Como estamos medindo o limite máximo para cima, ou seja, eixo Y, o que nos interessa é somente o valor 166. Fazemos o mesmo para as outras direções:

2. Limite inferior para o eixo Y, valor: -166.



3. Limite máximo para a esquerda no eixo X: -235.



4. Limite máximo para a direita no eixo X: 105.

Estes 

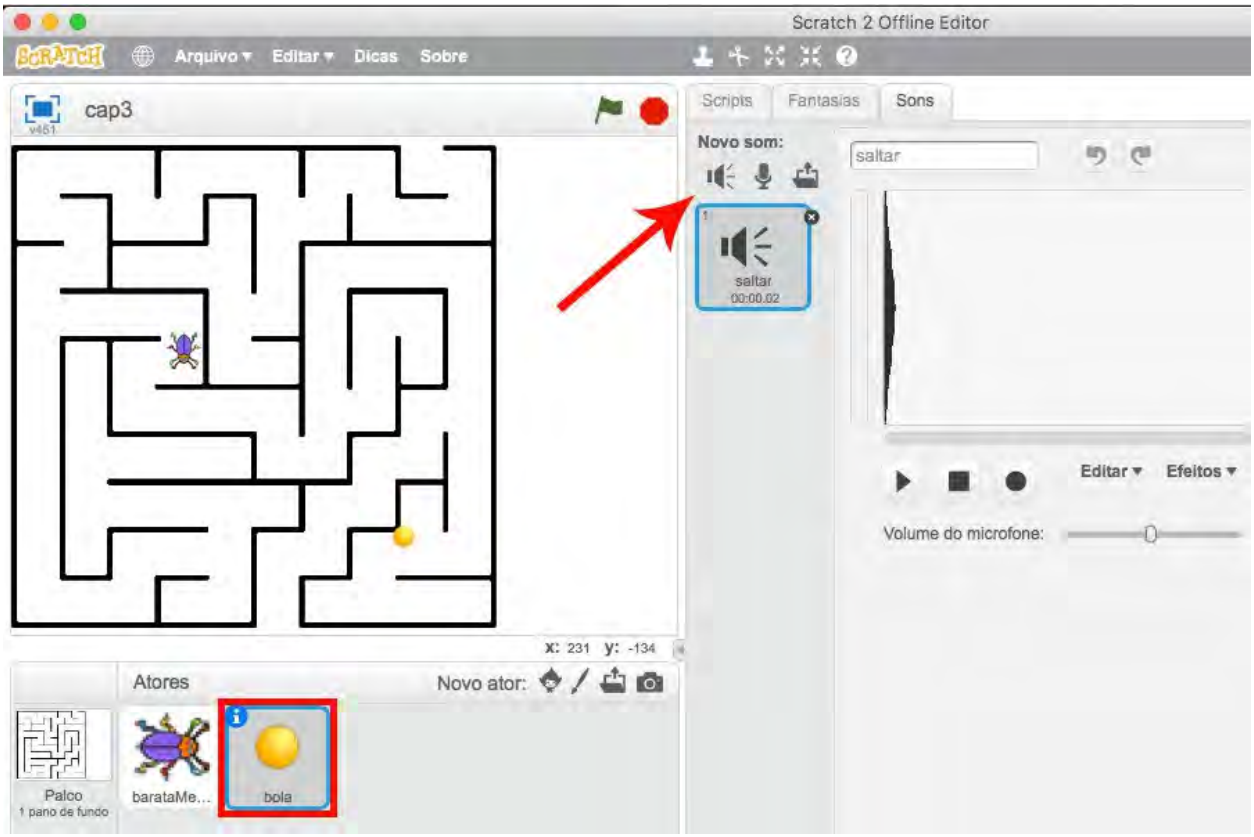
não são valores exatos, dependendo da forma que você posicionar o labirinto, os valores podem mudar, ou até mesmo dependendo de onde você posicionar a seta do mouse. Não fique preso nestes valores, faça a sua própria medição e altere no bloco número aleatório entre. Após isso, teste seu jogo e veja se já está funcionando o bloqueio das paredes.

Sons Nossa primeira parte do jogo está pronta. Nos próximos capítulos, trabalharemos com a parte lógica do jogo, ou seja, pontuação, leitura e armazenamento de dados. Mas agora, para dar um toque final, vamos adicionar alguns sons para a movimentação do personagem, um som ao capturar a bolinha e um fundo musical. Vamos usar diferentes tipos de repetição para sons.

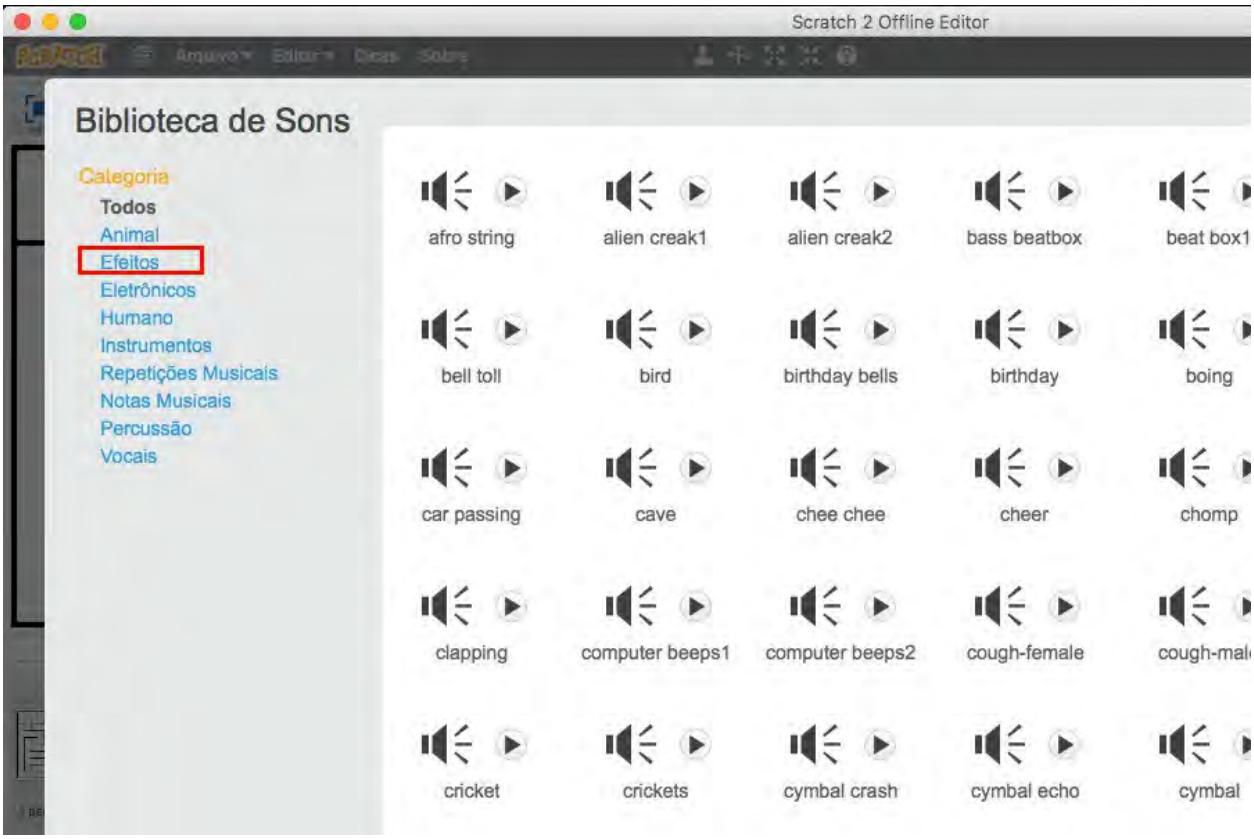
Por exemplo, no fundo musical, precisaremos que o som seja executado infinitamente, sendo assim, vamos utilizar o bloco sempre. Ao capturar a bolinha, precisaremos que o som seja executado caso o ator esteja encostando na bolinha, então precisaremos usar uma condição. Dessa forma, realizaremos a execução dos sons.

Em um primeiro momento, vamos adicionar um som ao capturar a bolinha. No próximo capítulo, vamos implementar um fundo musical com a utilização de variáveis.

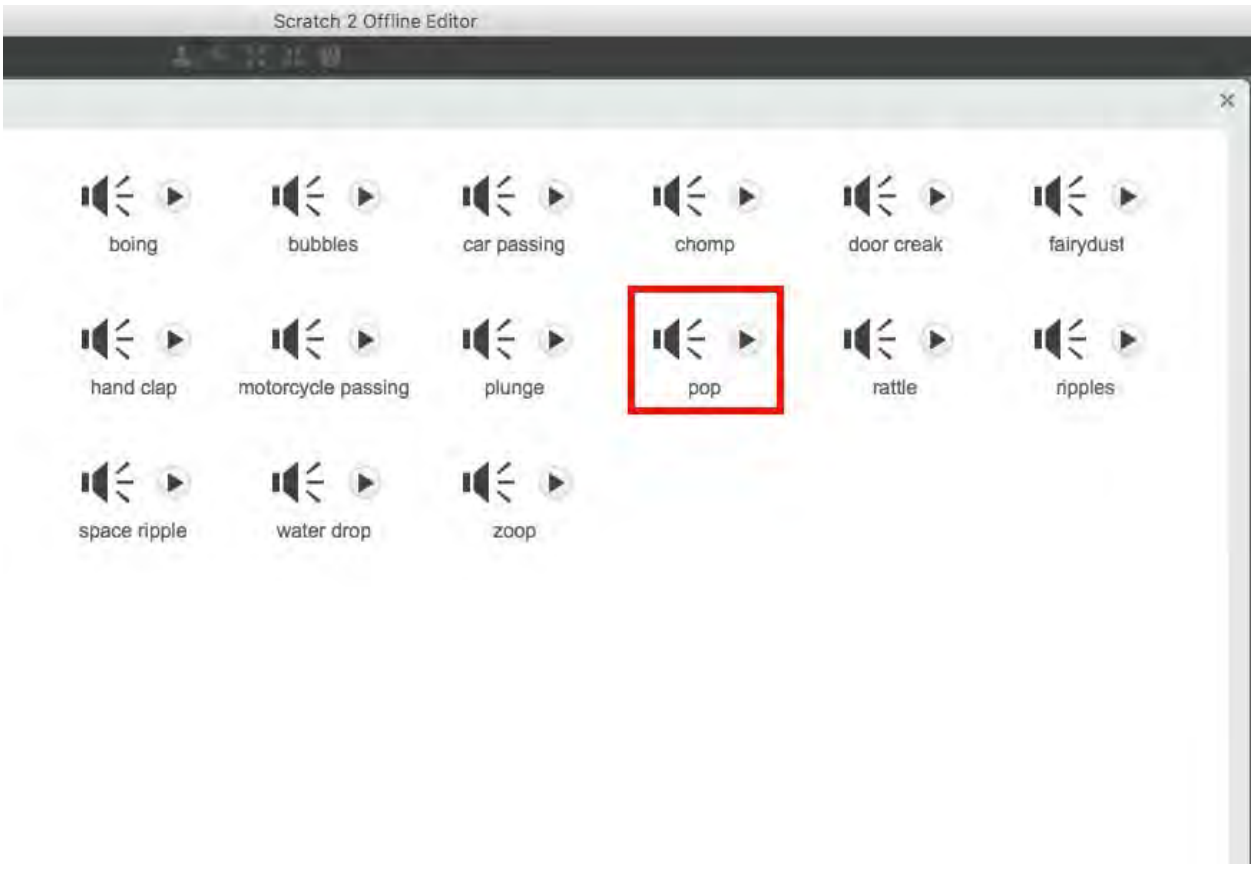
1. Primeiro som a ser adicionado é o de quando o jogador capturar a bolinha. Para isso, selecione o ator bola e, na aba Sons, clique em Escolher som da biblioteca.



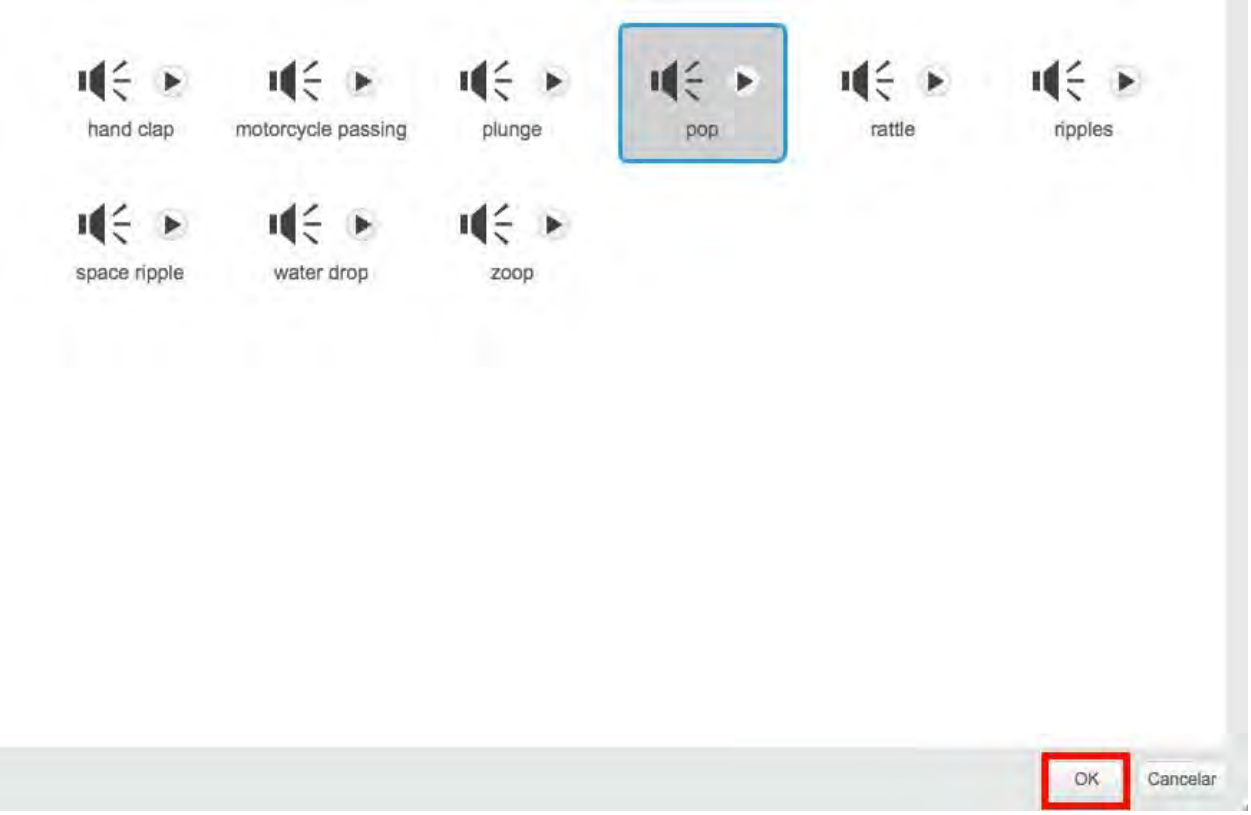
2. Clique em Efeitos.



3. Selecione o som Pop.

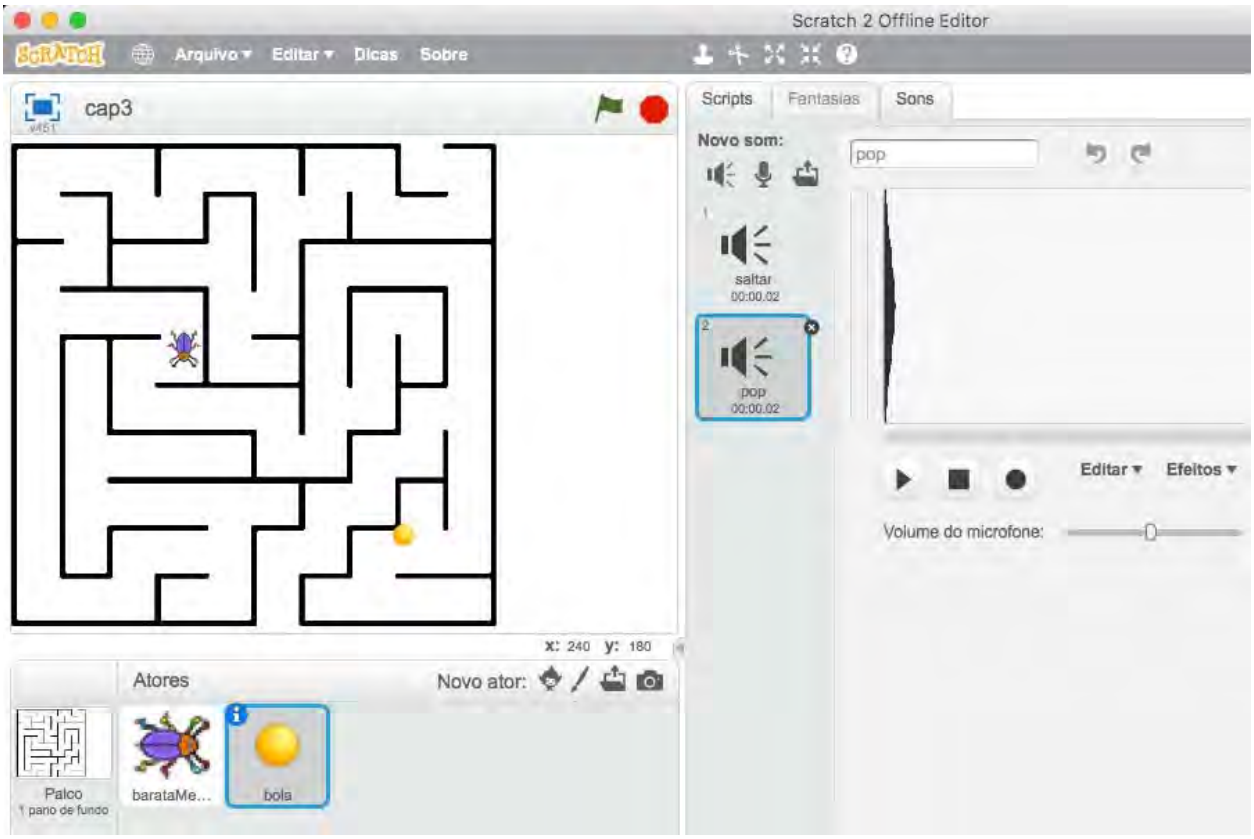


4. Clique em Ok.

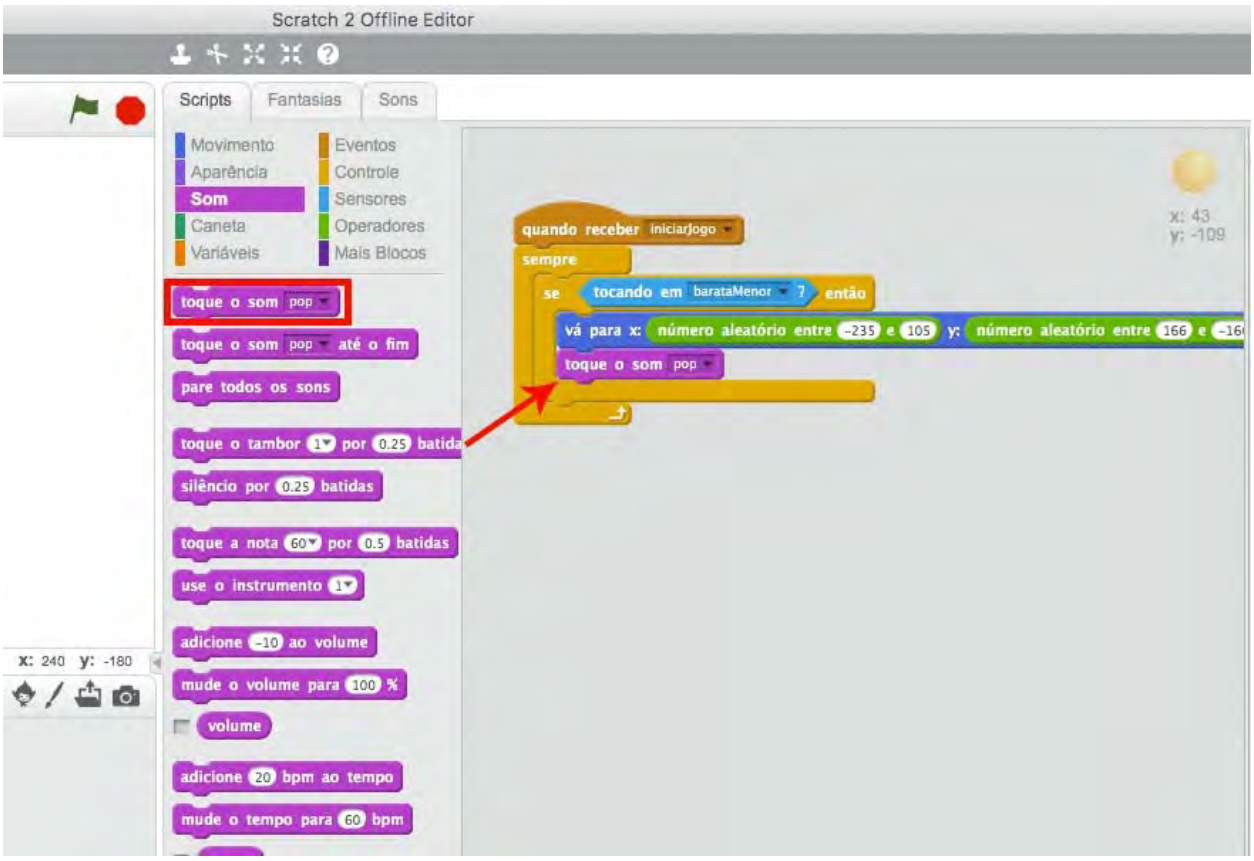


Não é necessário que você escolha exatamente este som, você pode ouvir os outros disponíveis e escolher um que você prefira. Faremos com esse pois foi o som que melhor se encaixou com a proposta do jogo.

5. Veja que o som foi adicionado abaixo do som saltar.

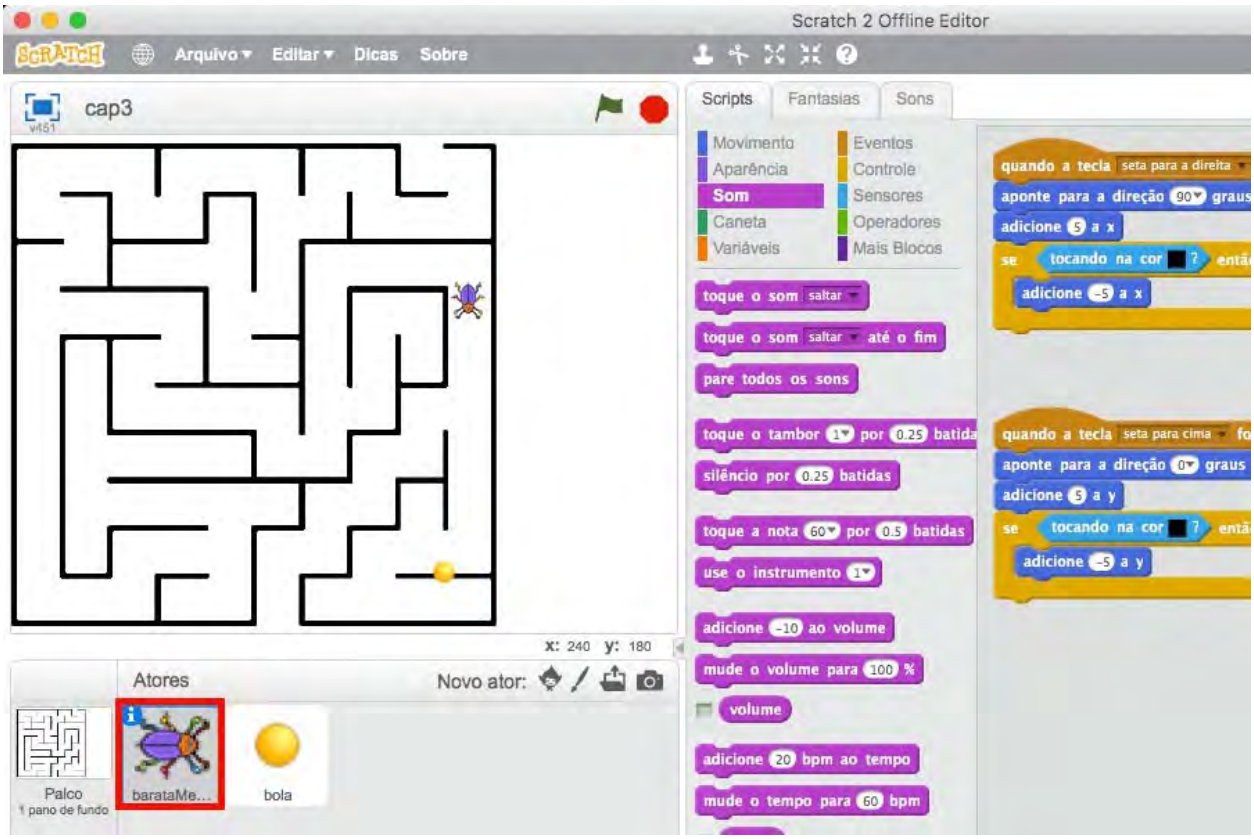


6. Agora clique na aba Scripts e em Som, arraste o bloco toque o som. Se for necessário, altere o som para pop.

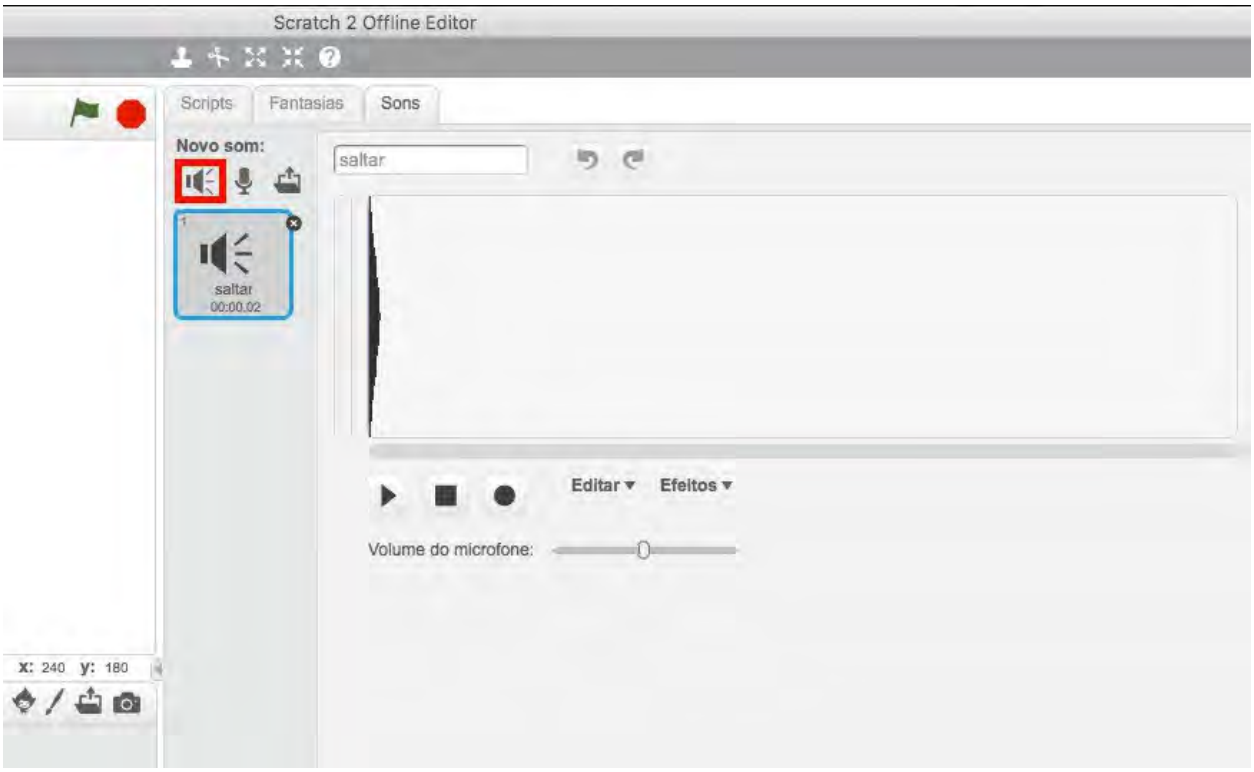


Agora, vamos adicionar um som para o movimento do ator.

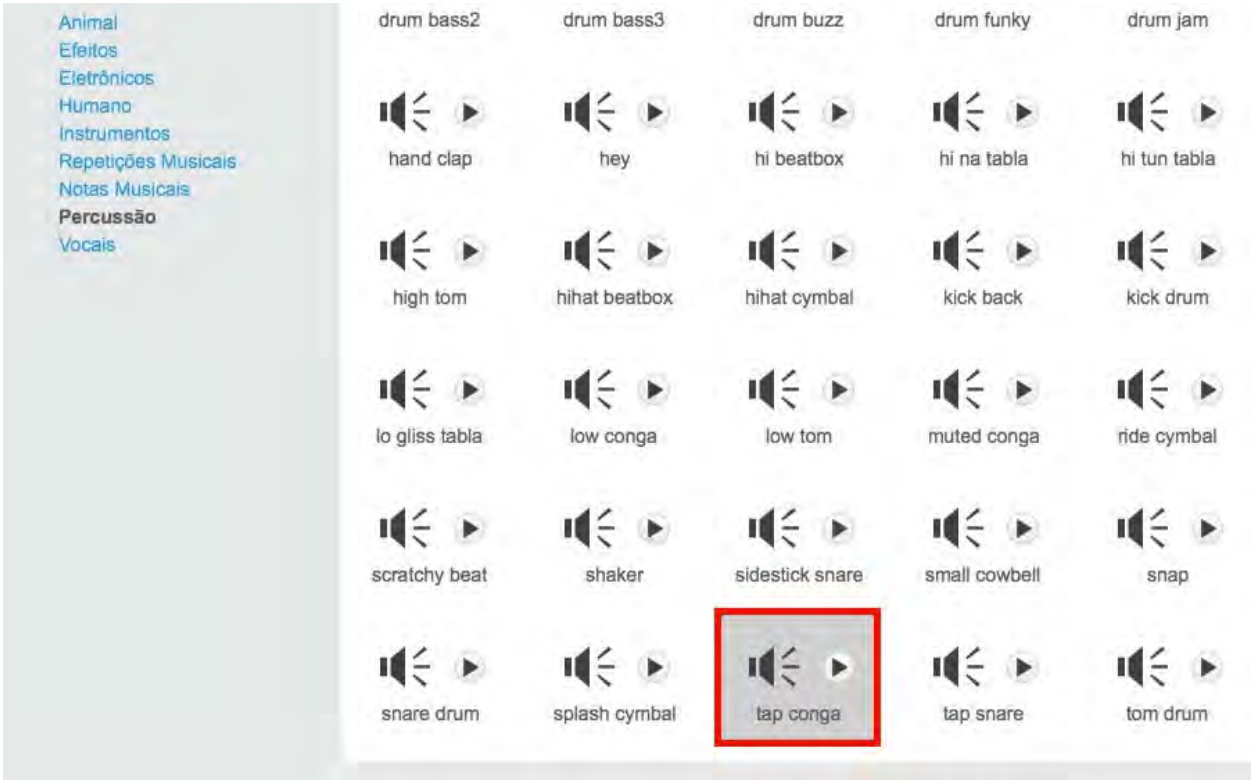
1. Selecione o ator barataMenor.



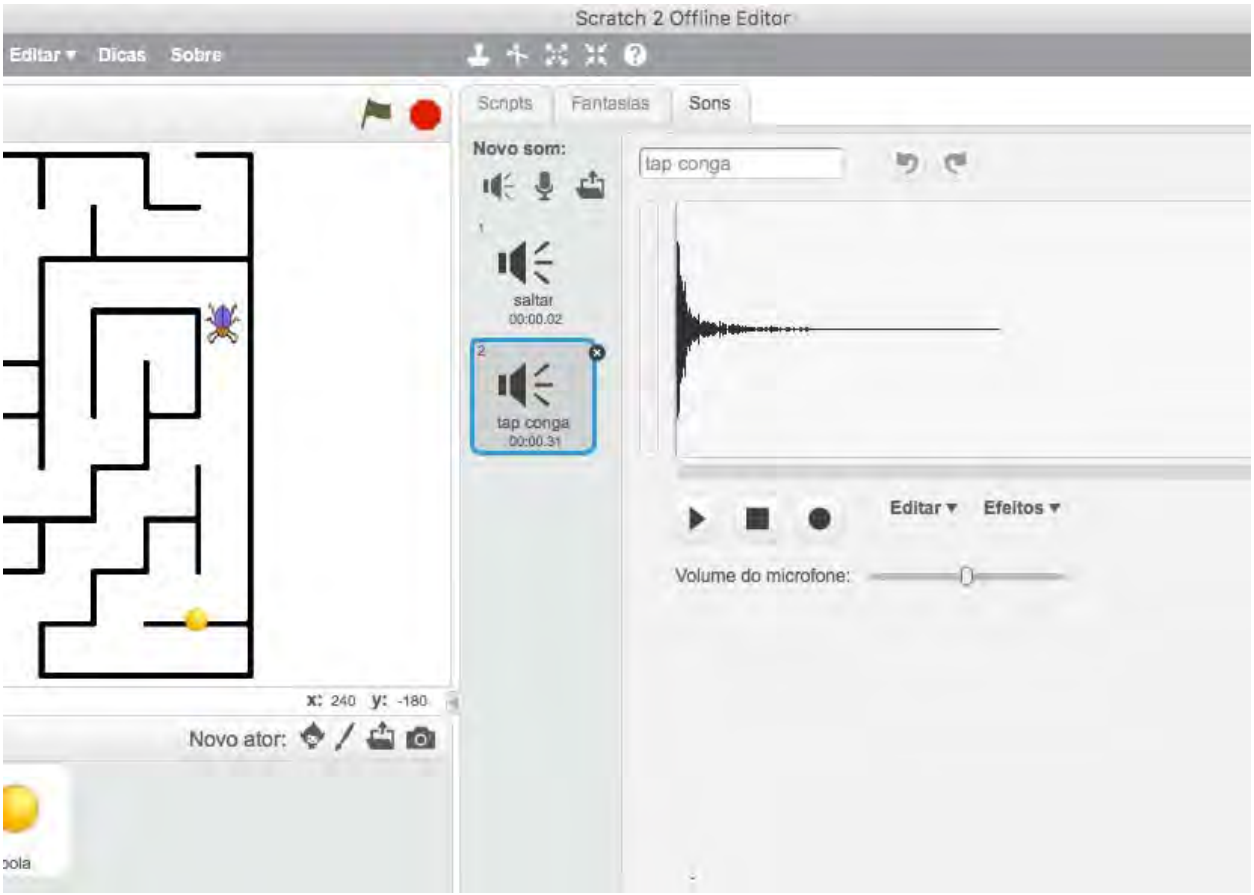
2. Na aba Sons, clique em Escolher som da biblioteca.



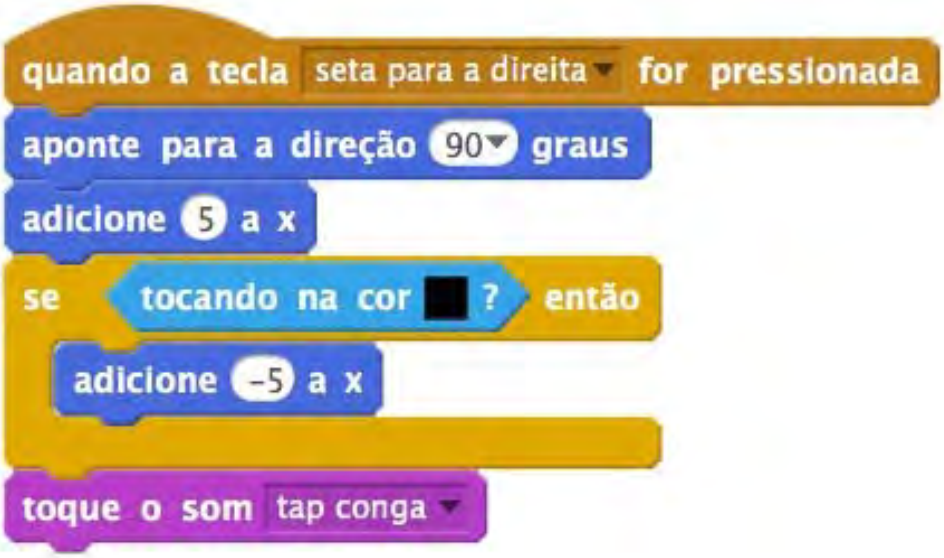
3. Em Percussão, selecione o som tap conga.



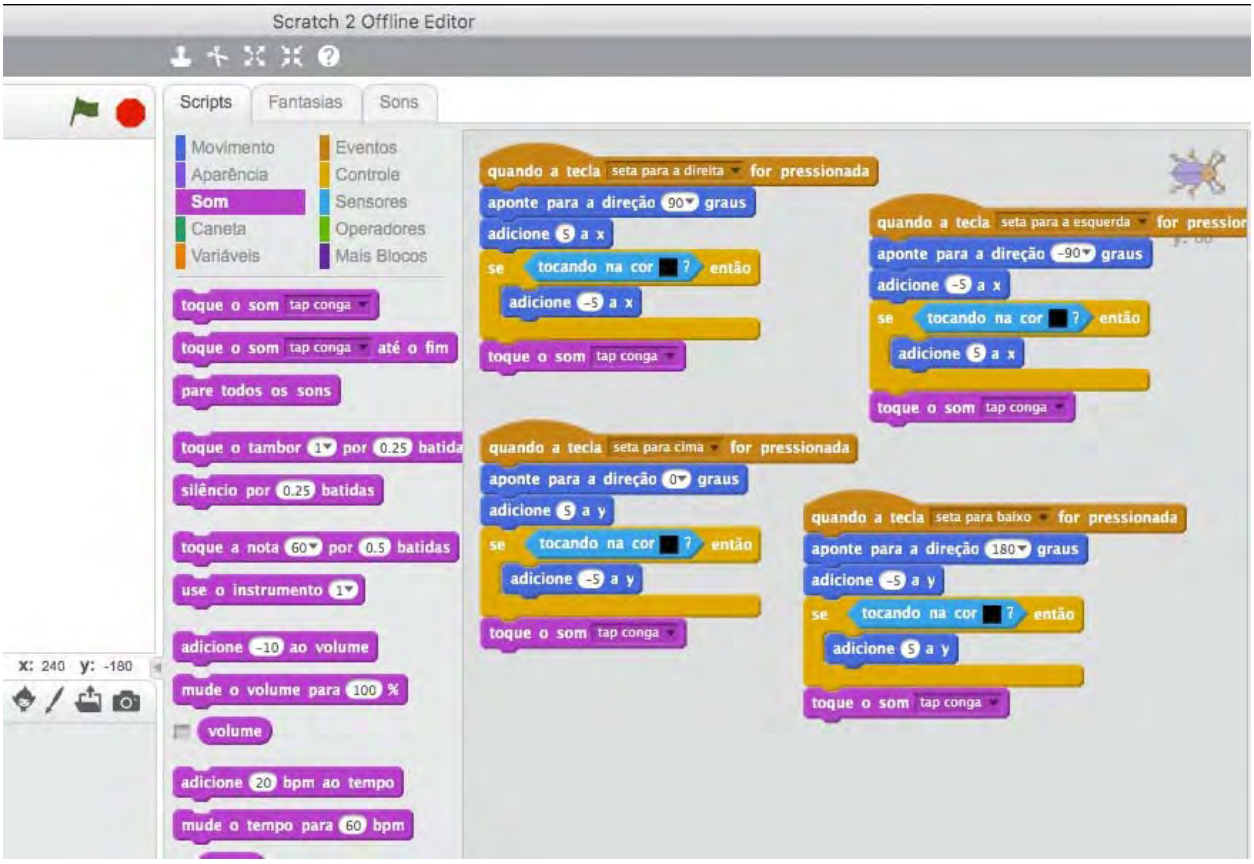
4. Veja que foi adicionado logo abaixo do som saltar.



5. Vá na aba Scripts e em Som, arraste o bloco toque o som para baixo do bloco Se então. Se for necessário, altere para tap conga.



6. Ficará assim:



Novamente, você poderá escolher outro som que deseje. Fique à vontade e construa, pois o jogo é seu! :) Teste e brinque um pouco com o seu jogo, e veja se tudo está funcionando como deveria! Se tudo estiver correto, você já estará pronto para o próximo capítulo e para seguir com a sua carreira de programador.

3.4 Conclusão

- Neste terceiro capítulo, vimos os comandos de repetição que permitem executar trechos de código repetidamente.
- Demos continuidade ao jogo de labirinto.
- Adicionamos alguns sons para dar mais dinâmica ao jogo.

No próximo capítulo, estudaremos variáveis, que nada mais são do que espaços de memória do computador que permitem guardar informações como nome, pontuação e tempo. Daremos continuidade ao jogo e implementaremos um sistema de pontuação e contador de tempo. Isso vai permitir que o jogo seja incrementado de muitas funções novas.

CAPÍTULO 4

Variáveis

4.1 Introdução

Existem situações nas quais precisaremos guardar dados e/ou informações para serem usados posteriormente em nossos programas. Se estivermos desenvolvendo um jogo, precisaremos guardar a pontuação do jogador. Caso estejamos desenvolvendo um programa de gerenciamento, precisaremos guardar dados como nome, data de nascimento e CPF. Essas informações são armazenadas utilizando **variáveis**.

Variáveis são espaços na memória do computador que nos permitem armazenar dados. Porém, não podemos guardar qualquer tipo de dado. Precisamos especificar ao computador o que estaremos armazenando. Existem dezenas de tipos de dados, mas vamos trabalhar especificamente com três tipos no Scratch, são eles: Numeral, Caractere e Booleano.

Se fôssemos declarar uma variável em uma linguagem de programação, como por exemplo, o JavaScript, faríamos o seguinte: `var exemplo = 'Exemplo de uma variável';`

Esta é uma variável que armazena uma string, ou seja, um texto. Será sempre necessário darmos um nome a uma variável, e é bom que esse nome seja bastante sugestivo para podermos identificá-la melhor.

Poderíamos declarar variáveis que armazenam números também, mas o Scratch, para fim de aprendizado, não faz essa distinção. Em uma mesma variável, é possível armazenar qualquer coisa, seja um número, caractere ou booleano.

Numeral	Armazena números	0, 1, 2, 3, 4... 8392 -20 0.12
Caracter	Armazena letras e símbolos	a, b, c, d, e, f... Escola, scratch, blabla... @!*&
Booleano	Armazena valores lógicos	0, 1 V, F

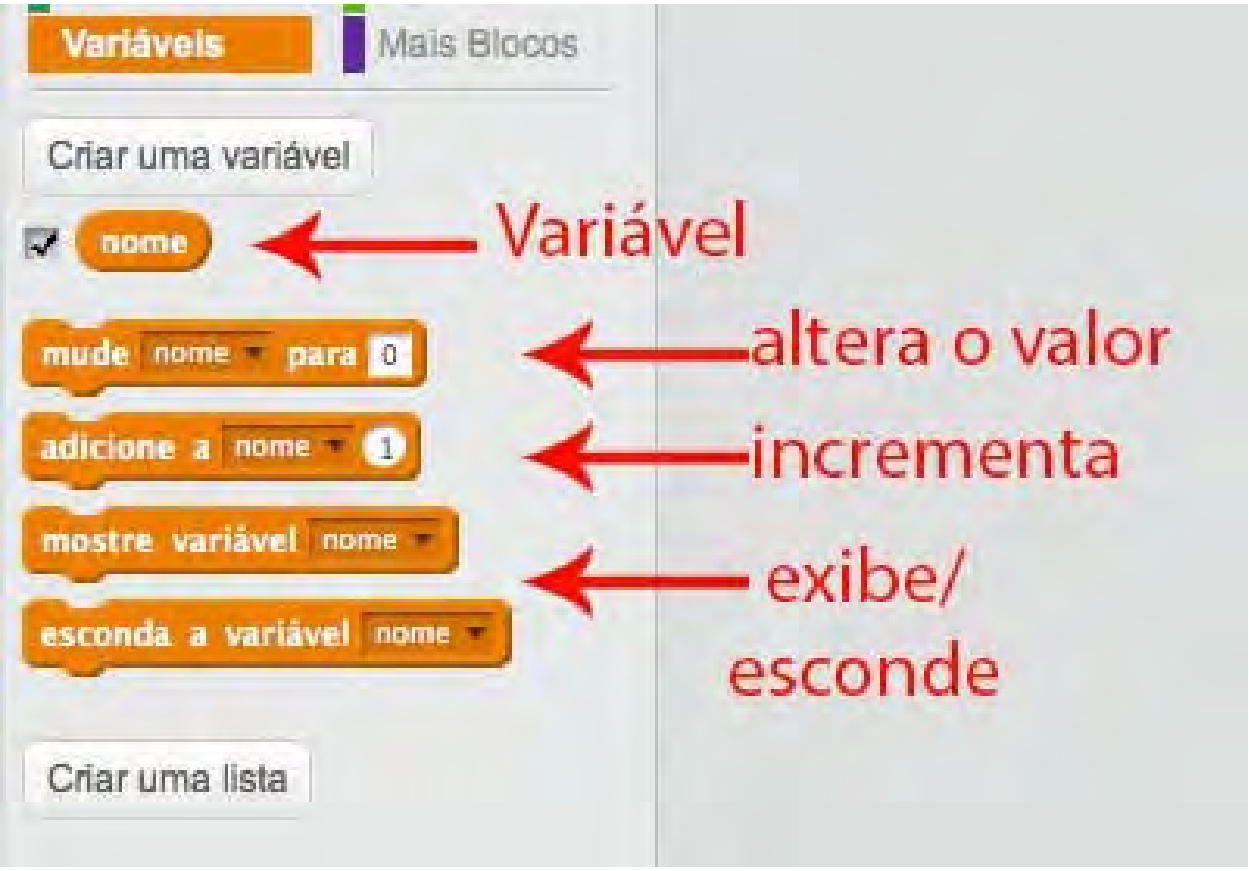
Veja que já conhecemos um tipo de dado que é o valor booleano. Lembre-se de que o tipo armazenado em uma variável do tipo booleano sempre será um valor lógico, ou seja, 0 ou 1

(V ou F). O tipo de dado numeral nos permite armazenar números, e o tipo de dado caractere nos permite armazenar letras e símbolos.

4.2 Declarando variáveis no Scratch

Antes de começarmos a criar as variáveis, vamos entender como podemos manipular os dados. Podemos realizar 3 operações básicas:

- Alterar o valor;
- Incrementar (caso seja um número);



- Exibir/esconder.

Dentro da seção de variáveis, existem alguns blocos que permitem manipular os dados de uma variável. Por exemplo, se quisermos que, ao reiniciar o jogo labirinto, os dados de uma variável sejam zerados, precisamos alterar o valor delas para 0. Qualquer coisa que definirmos dentro da caixa do bloco mude vai ser alterado. Porém quando implementarmos o sistema de pontuação, será necessário incrementar valores.

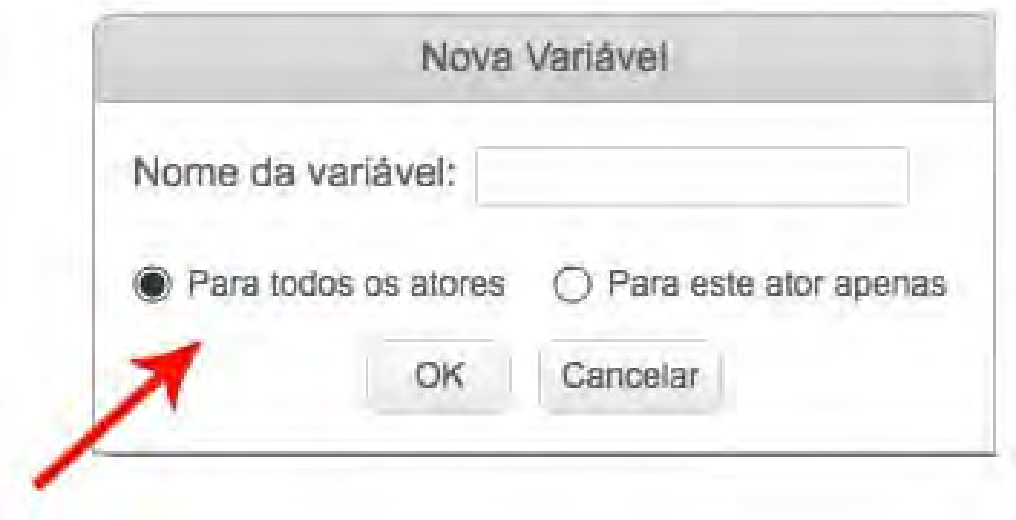
Também é possível utilizar o bloco mude para realizar esta tarefa, mas é preciso um pouco mais de código para isso. O bloco adicione nos poupa tempo e código, porque é necessário somente que você defina o quanto será adicionado ao ser executado. Se há algumas informações que você não quer que o usuário veja, é possível ocultar as variáveis através do bloco esconda; ou se quiser voltar a exibir em alguns momentos, utilize o bloco mostre. Por padrão, as variáveis estão sempre à mostra no Scratch.

4.3 Variáveis globais e locais

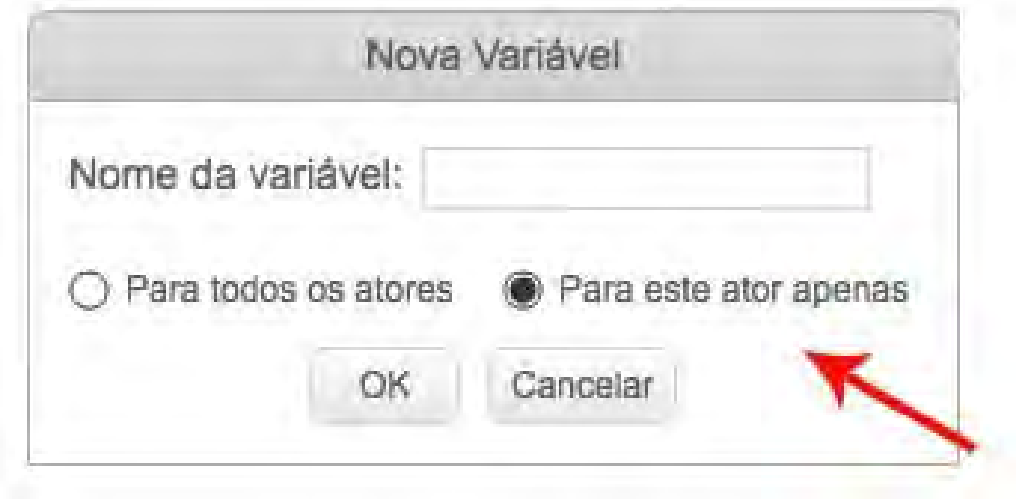
Algumas vezes, precisaremos criar variáveis que têm acesso limitado, ou seja, apenas uma pequena parte do nosso programa conseguirá ter acesso. Raramente você precisará usar variáveis globais, somente em casos em que você precise que uma informação seja compartilhada entre os atores, tornando desse modo o dado mais "geral".

Variáveis locais só estão disponíveis quando certo trecho de código é executado, e logo após a execução ela é apagada da memória do computador. Evite o uso indiscriminado de variáveis globais. Use somente se necessário, pois ocupam espaço na memória do computador e tornam o código menos legível.

Cria uma variável global:



Crie uma variável local:

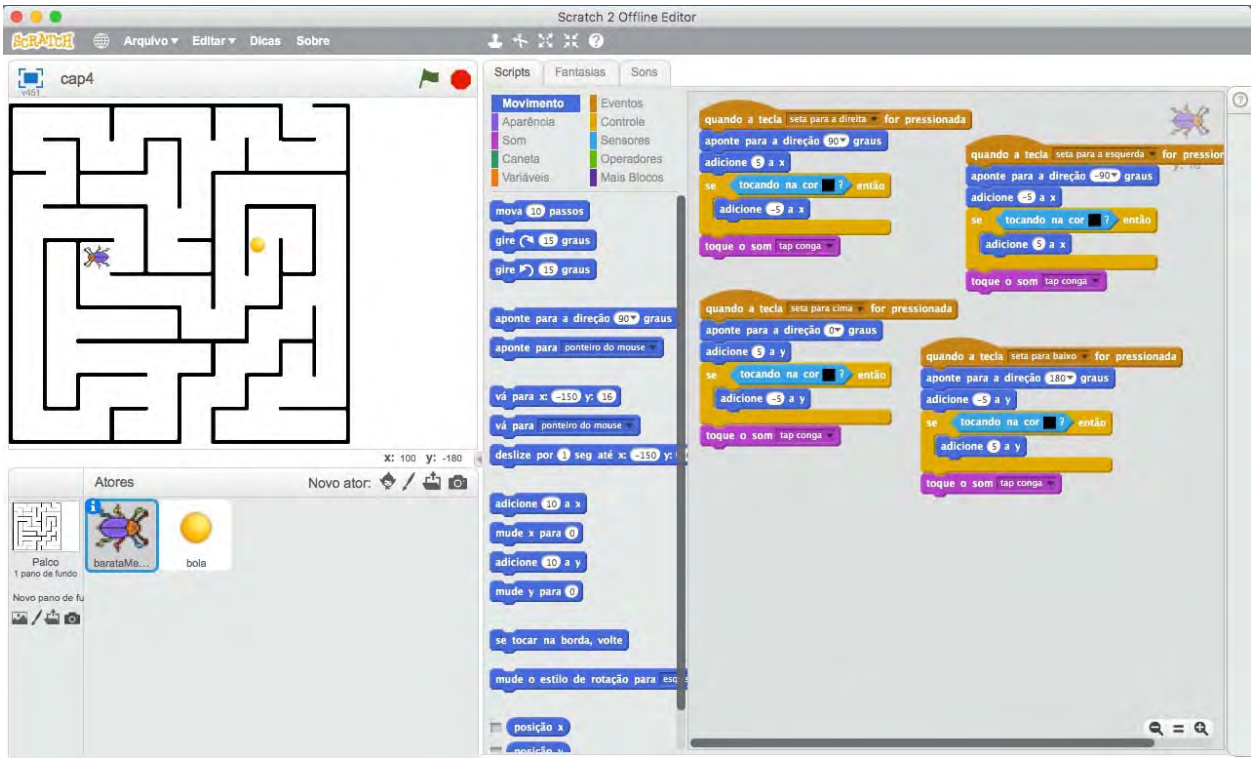


Após todo esse apanhado de informações sobre variáveis, nada melhor do que praticar para aprender.

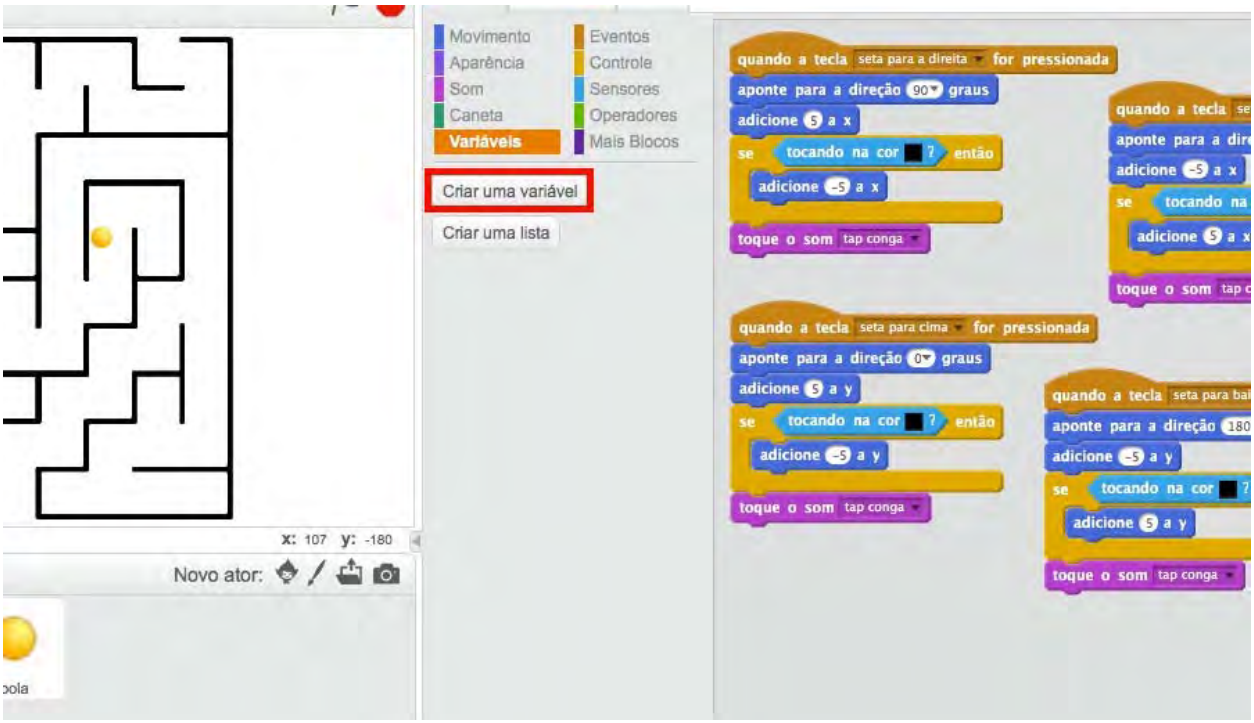
4.4 Labirinto

Vamos dar continuidade ao projeto, e neste capítulo implementaremos a pontuação e o tempo de jogo usando variáveis. Vamos começar criando o sistema de pontuação. O primeiro passo para isso é criar a variável responsável por armazenar os pontos.

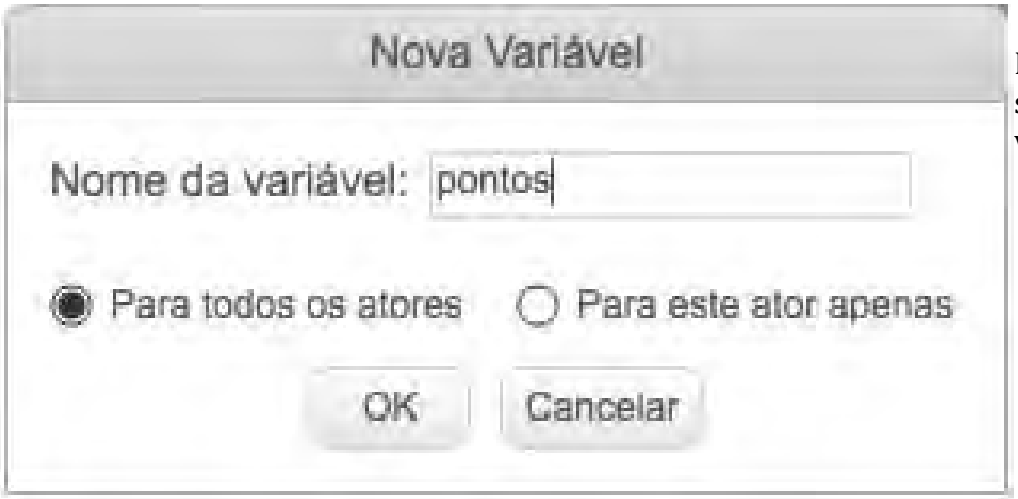
1. Abra o arquivo do capítulo anterior. Sua tela deverá estar parecida com essa, toda colorida:



2. Clique em Variáveis e depois em Criar uma variável. A partir de agora, vamos criar a primeira variável do nosso jogo e ela armazenará a pontuação.

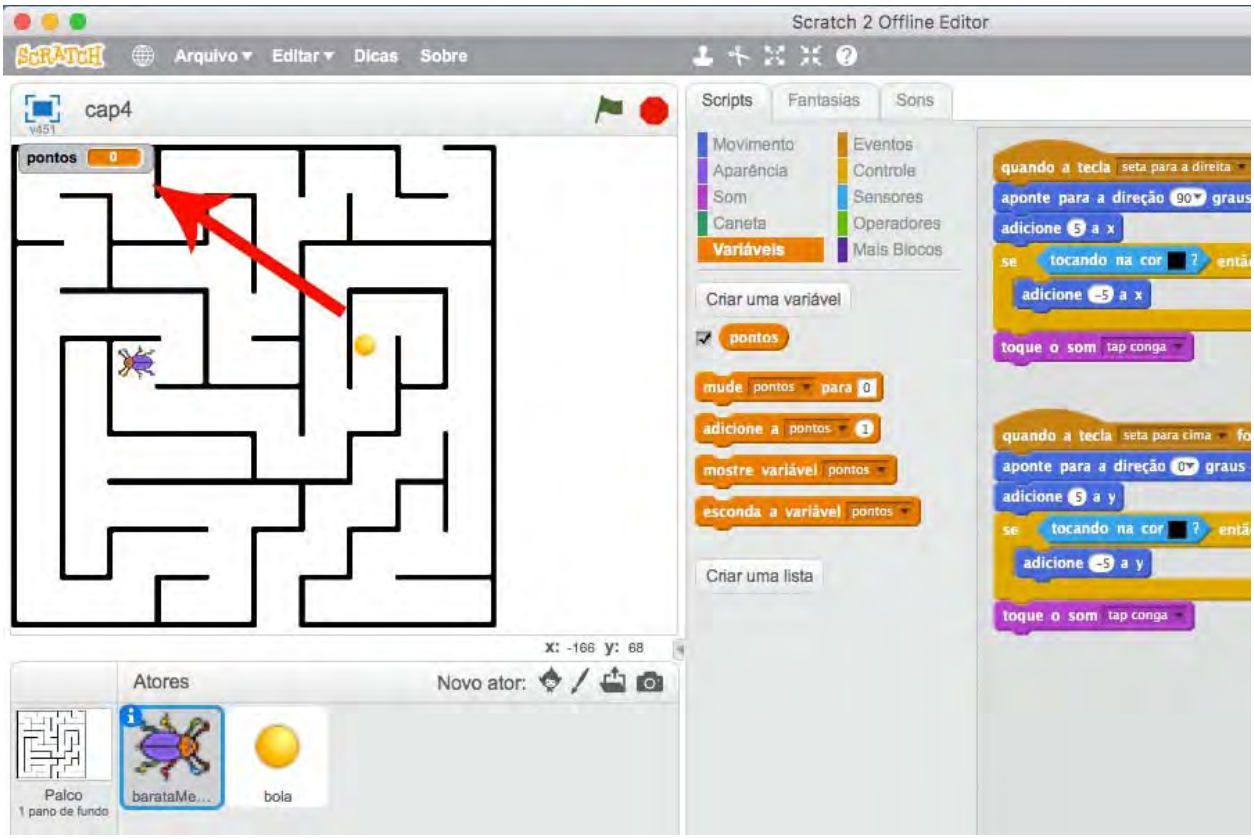


3. Após isso, digite pontos para dar um nome e selecione para todos os atores (variável global). Lembre-se de dar um nome sugestivo, ou seja, de simples memorização e que seja fácil saber do que a variável está tratando.



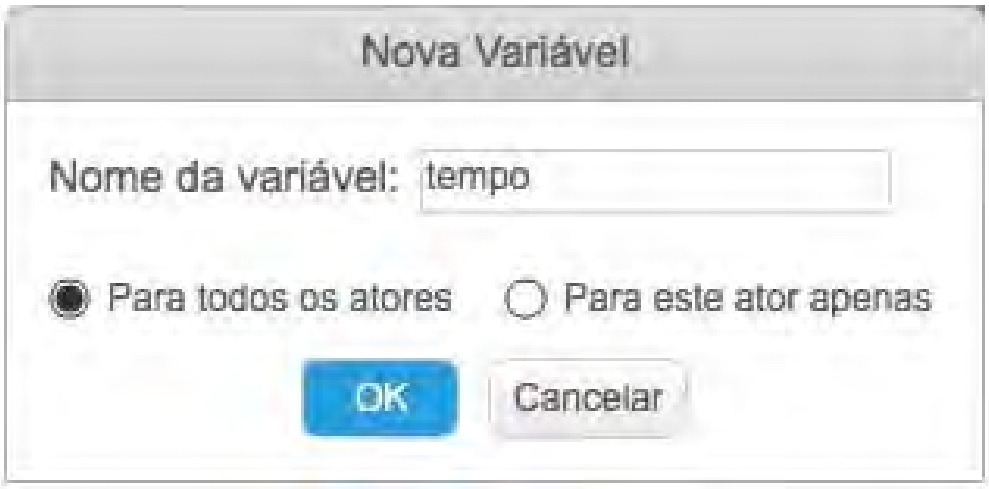
Esta será a variável responsável por armazenar a quantidade de pontos do jogador. Veja que estamos usando uma variável global. Mas você pode estar se perguntando: por que vamos usar variável global se foi dito que é bom a evitarmos? Pois bem, aqui estamos armazenando a pontuação do jogador, ou seja, uma informação geral que pode ser usada por qualquer um dos atores. Se um certo dado for de uso geral no programa, vários atores utilizarem não há nenhum problema. Dessa forma, nós podemos sim usar uma variável global. Lembre-se de que, se sua informação for compartilhada com outros atores, você poderá sim utilizar variável global.

4. Veja que apareceu do lado esquerdo na parte superior do palco a variável e o valor que ela contém.

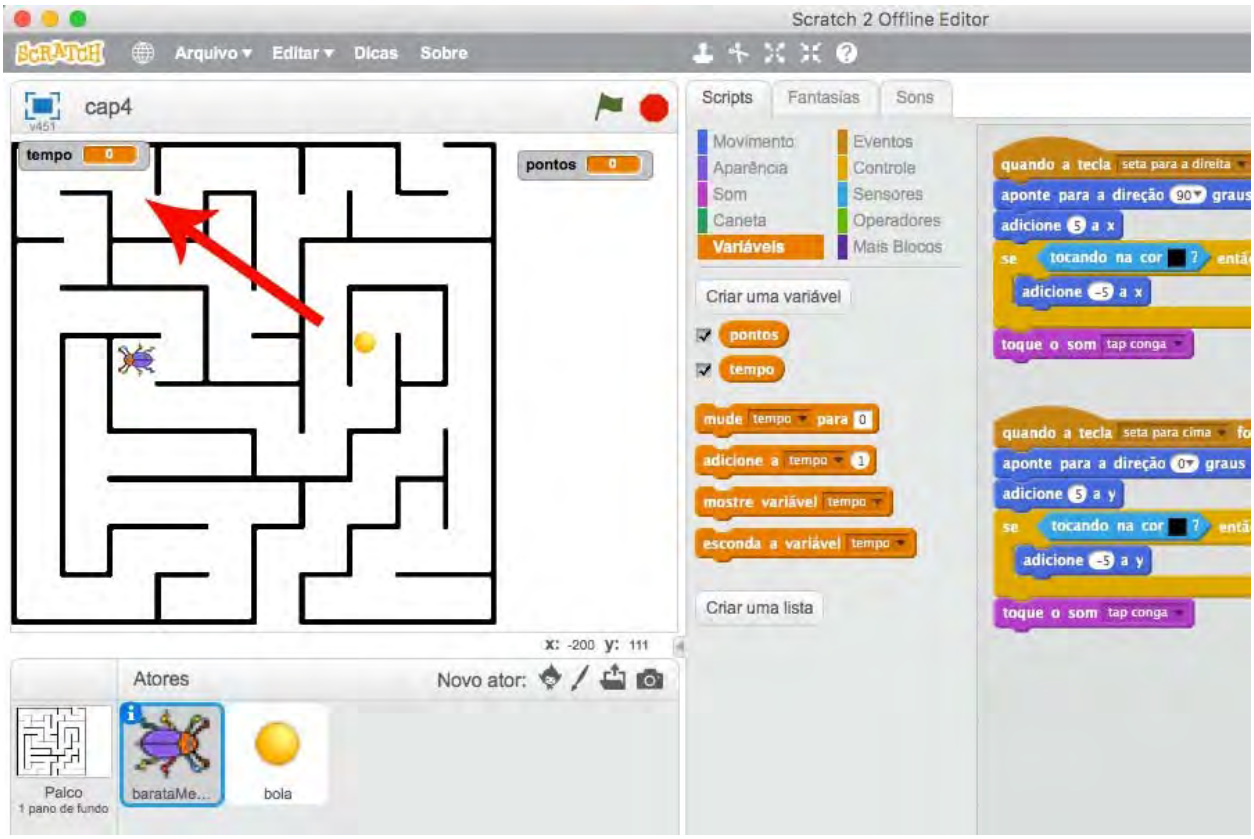


5. Para melhor organização da área e para a pontuação não aparecer "dentro" do labirinto, vamos arrastar para o lado direito.

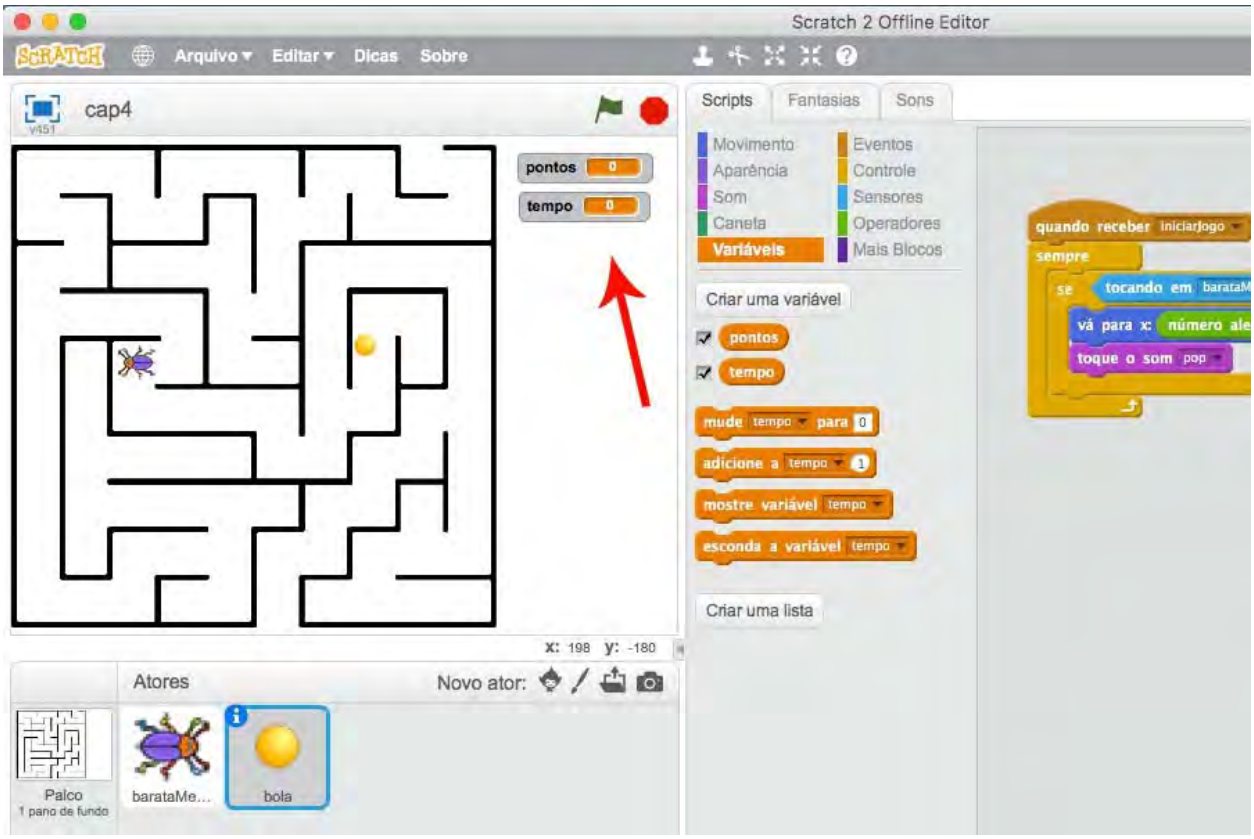
6. Novamente, vamos criar outra variável. Dessa vez, vamos chamar de tempo. Esta será responsável por armazenar a quantidade de tempo que ainda resta para o jogador, que será calculada através de um contador. Viu como está ficando interessante nosso jogo? As possibilidades de trabalhar com variáveis são muitas. Se você quiser, use sua criatividade e crie mais algumas variáveis que armazenarão outros dados do jogo.



7. Veja que ela também apareceu no lado esquerdo na parte superior do Palco. Sendo assim, por uma questão de organização, vamos arrastá-la para o lado direito. Somos programadores, mas às vezes também fazemos o trabalho de um designer.

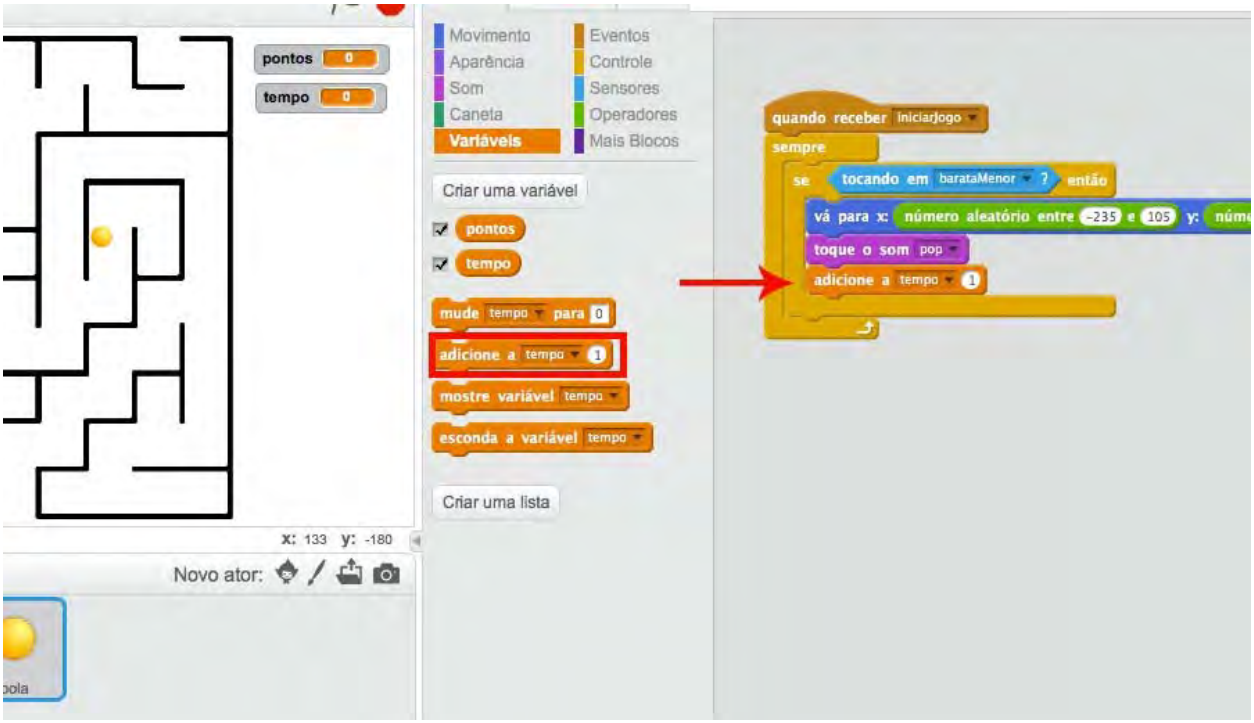


8. Arraste-a para o lado direito.



Adicionando pontos Agora que temos as variáveis criadas, vamos criar a lógica do sistema de pontuação. O funcionamento é bem simples: toda vez que o ator barata encostar na bola, é adicionado o valor 1 à variável pontos.

Nesse projeto do livro, a quantidade de pontos é de 1 em 1. Mas você pode personalizar para a quantidade que desejar, de 10 em 10, 5 em 5 ou 100 em 100.

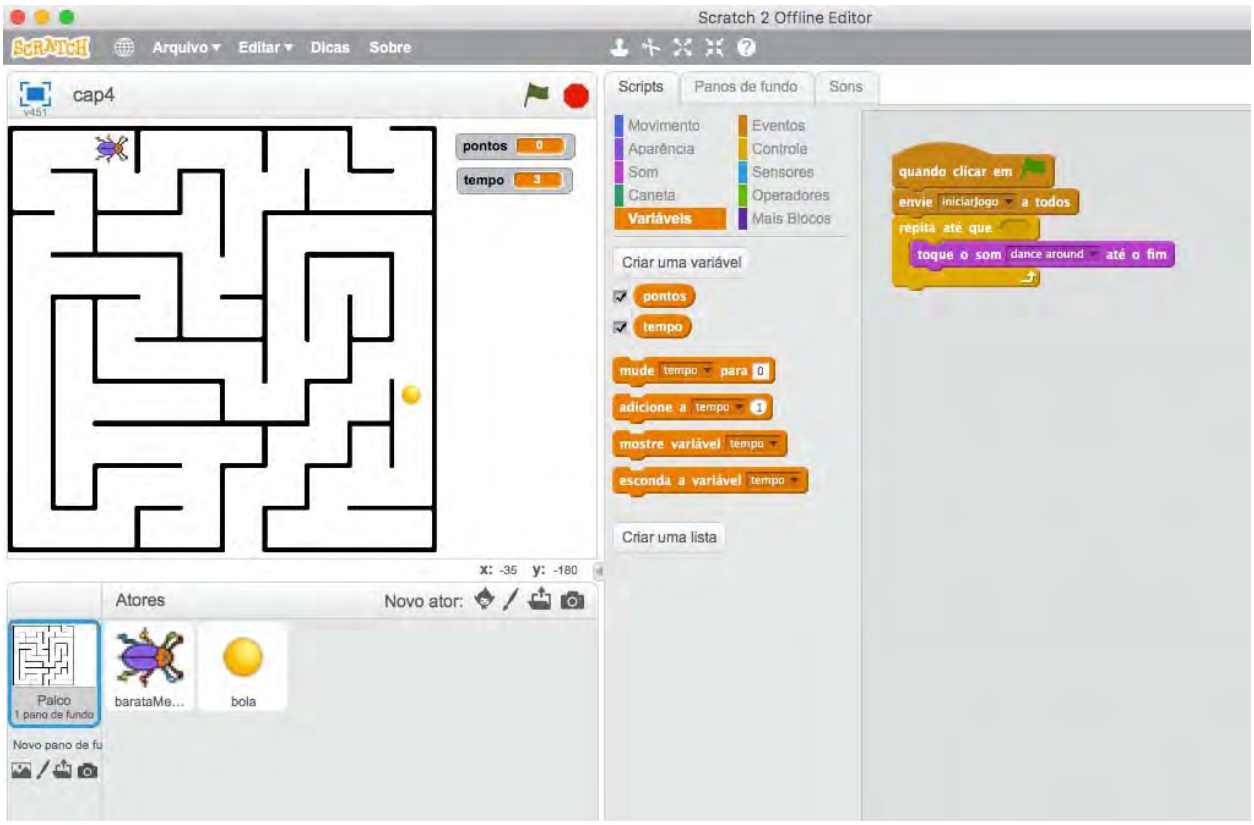


Criando um contador de tempo O objetivo do jogo é conseguir capturar a maior quantidade de bolinhas em menor tempo possível. O jogador terá o máximo de 1 minuto para capturar o máximo de bolinhas que conseguir. Para isso, teremos de usar em conjunto as variáveis tempo e pontos.

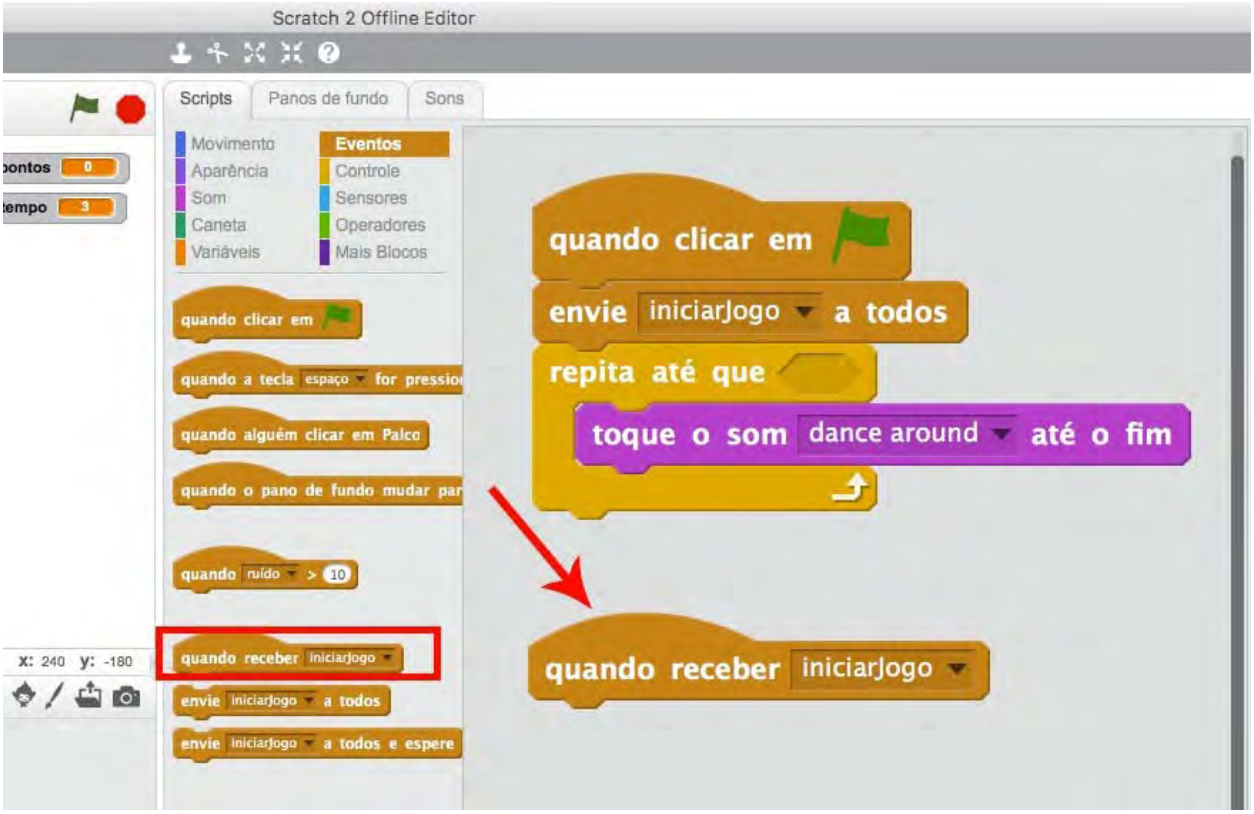
De que forma faremos isso? Simples. Primeiro usaremos uma variável global. Mas por quê? Queremos que a informação de quanto tempo resta seja compartilhada entre todos os atores, e também porque queremos que o jogo inicie com um tempo padrão.

Então, a cada 1 segundo, será decrementado da variável tempo o valor 1. Inicia-se com 60, e cada segundo que passa diminui o valor em 1. Quando a variável tempo chegar em 0, param todos os scripts que estão sendo executados. Confuso? Vamos programar para entender melhor.

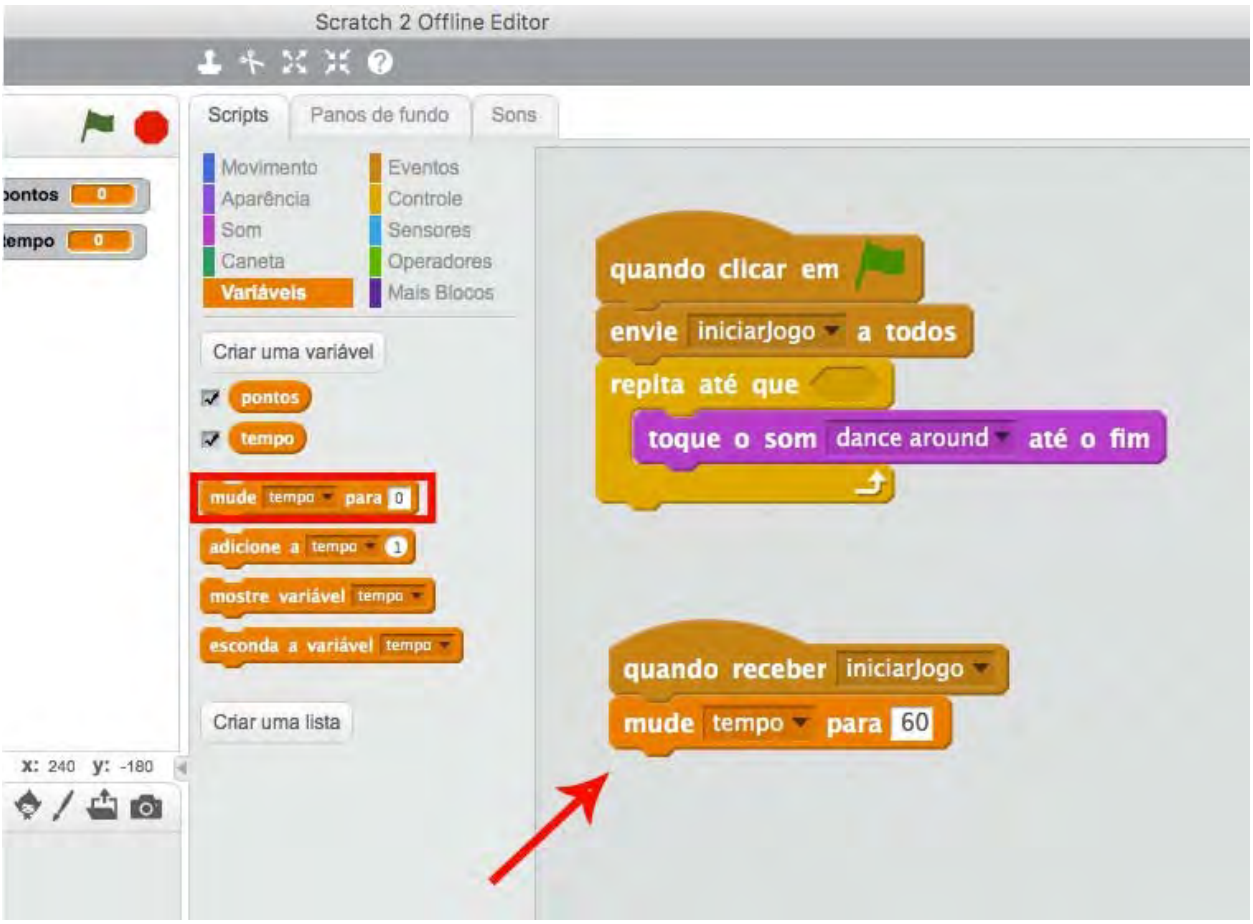
1. Clique em Palco.



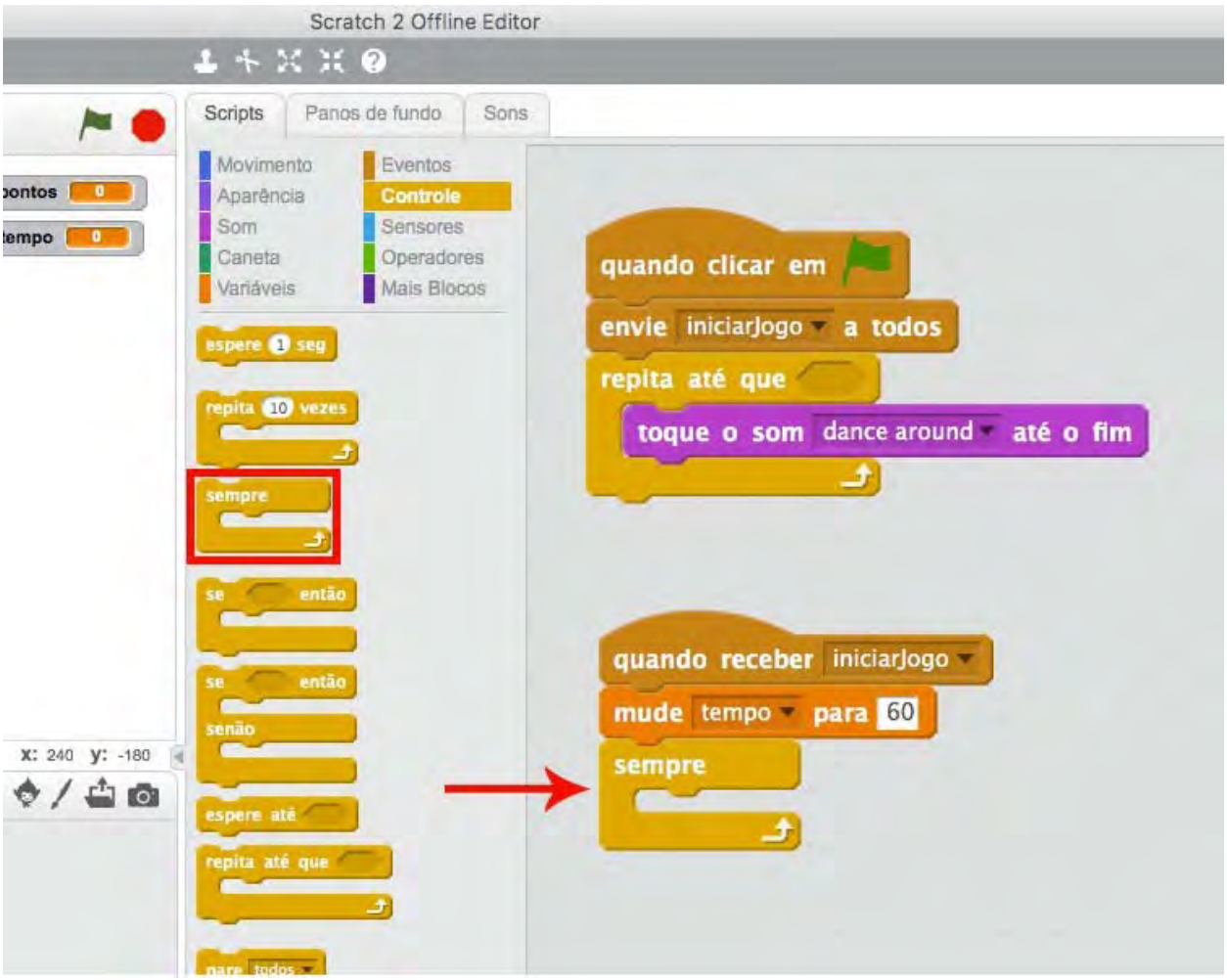
2. Em Eventos, arraste o bloco Quando receber iniciarJogo.



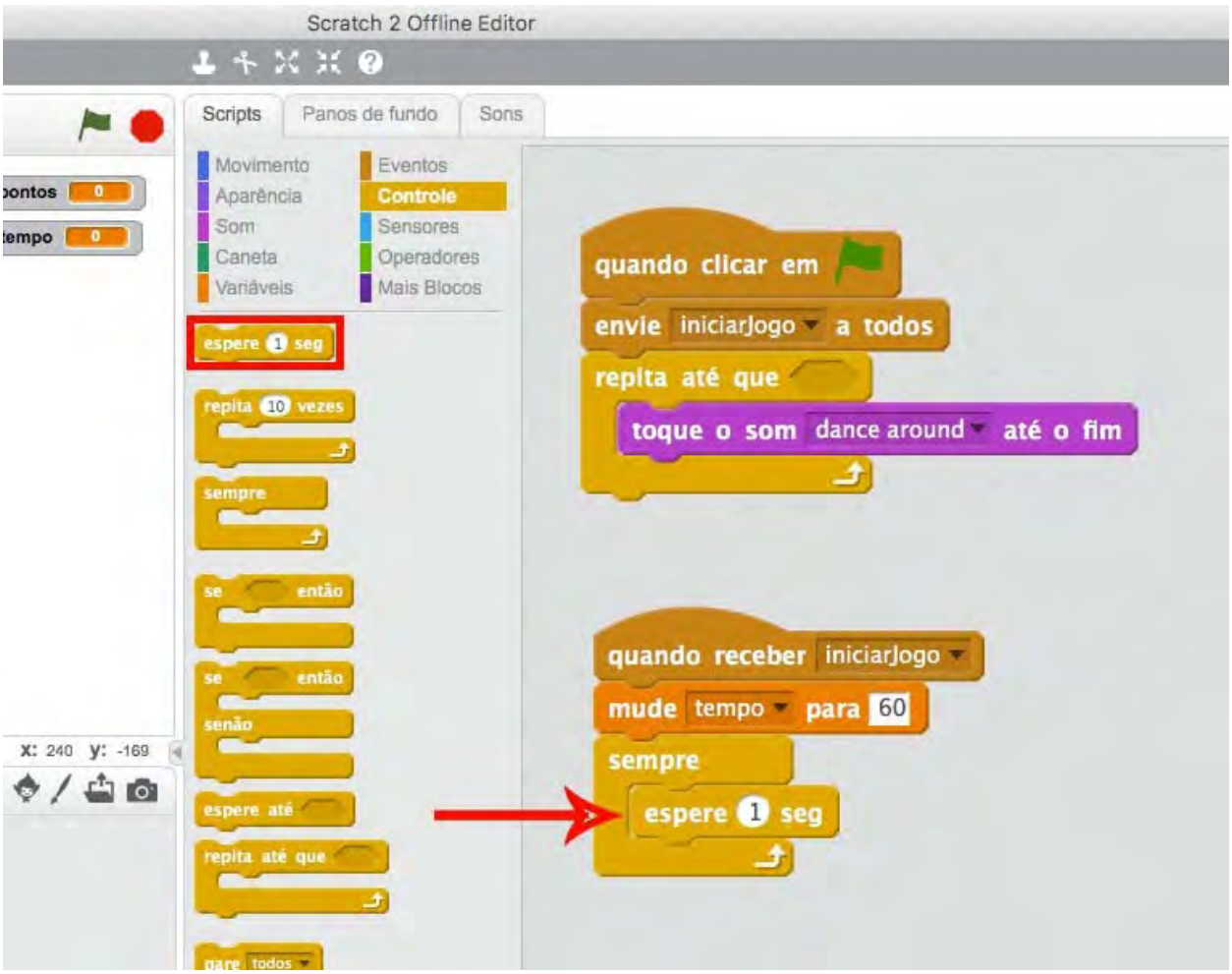
3. Agora em Variáveis, arraste o bloco mude para. Aqui vamos definir para o tempo começar com 60 segundos. Variável global, já iniciando com valor padrão.



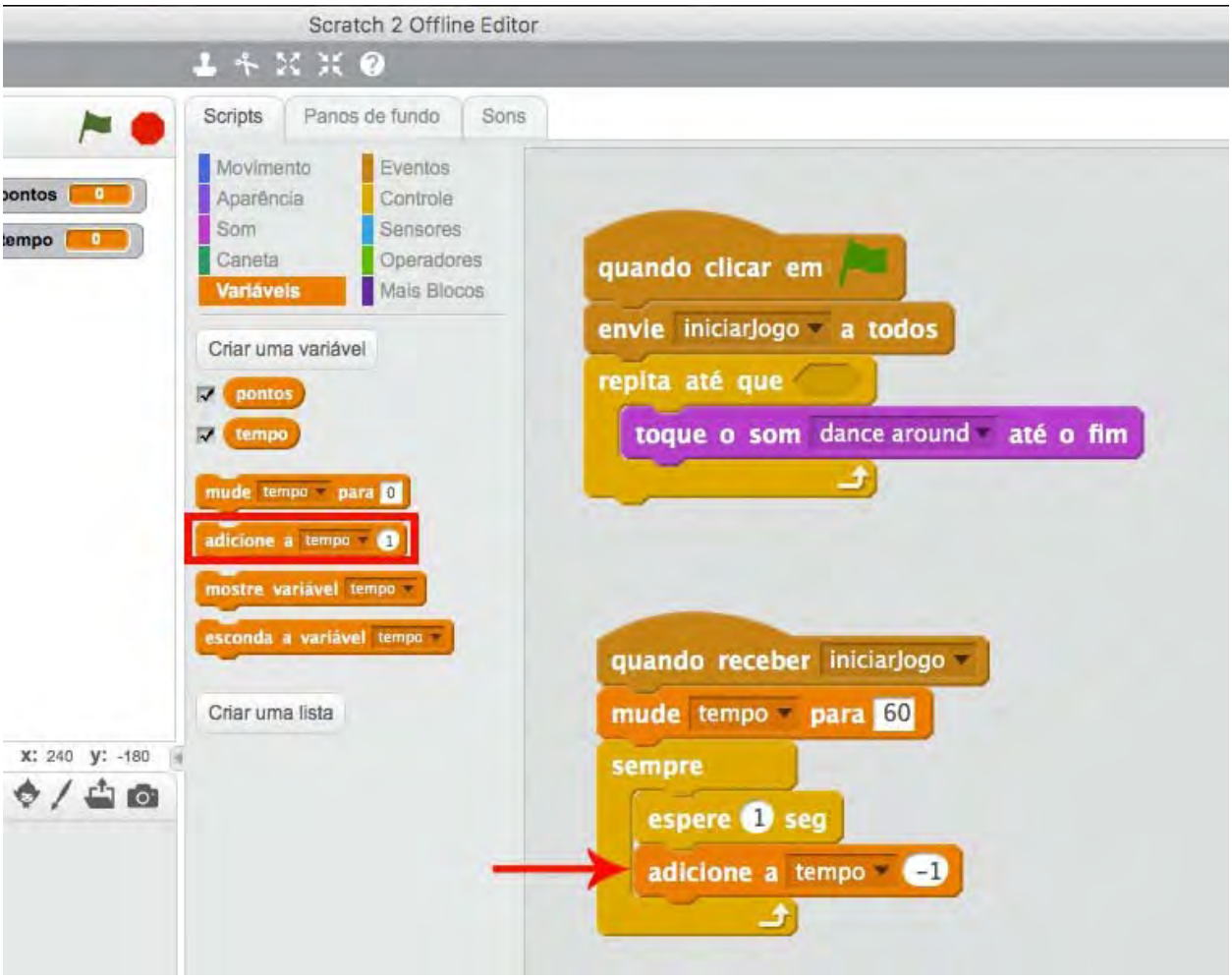
4. Em Controle, arraste o bloco Sempre para criar um loop infinito.



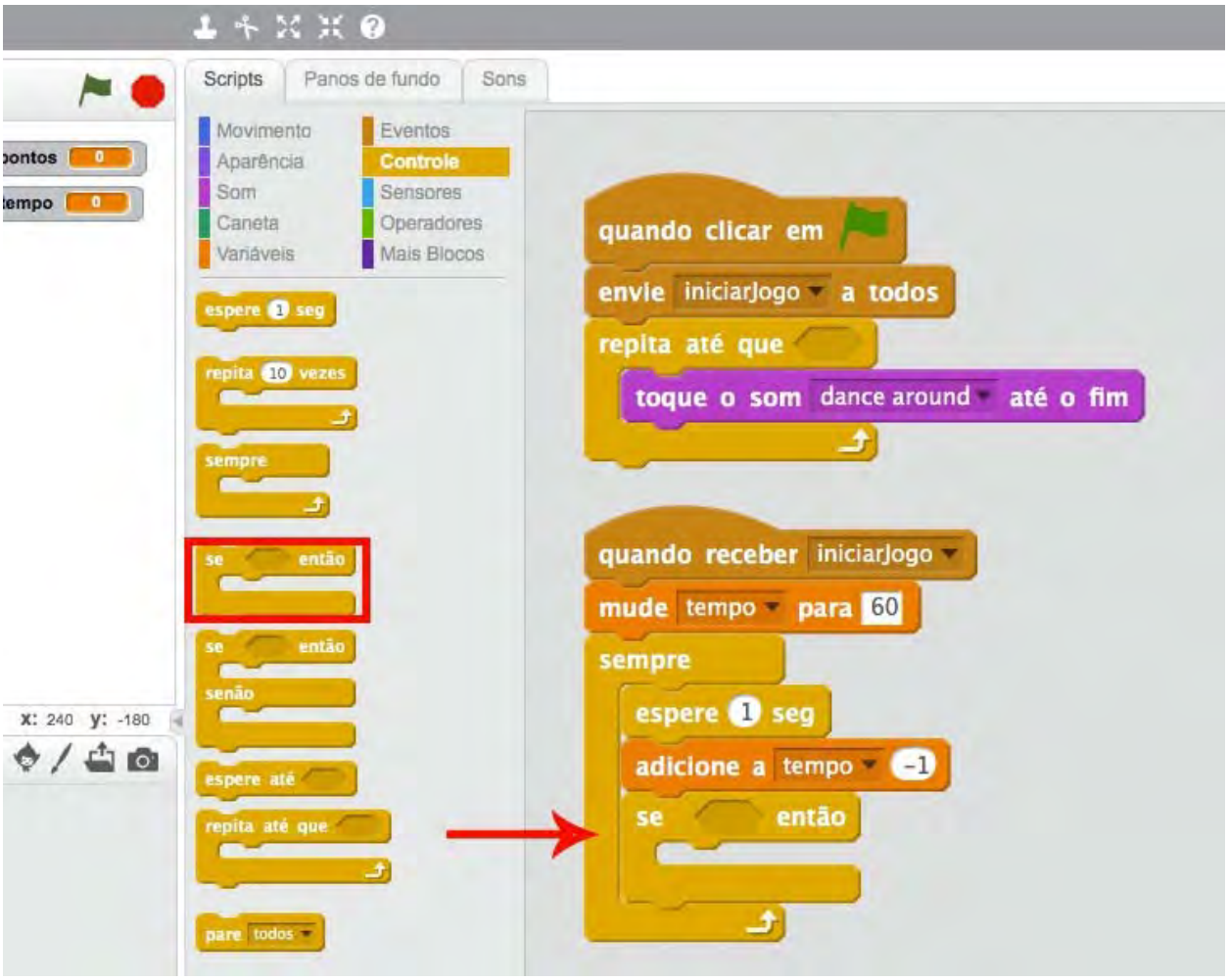
5. Em Controle, arraste o bloco espere 1 seg. Não altere este valor, pois é necessário que haja uma pausa a cada 1 segundo para o contador funcionar, que vai de 60 até 0.



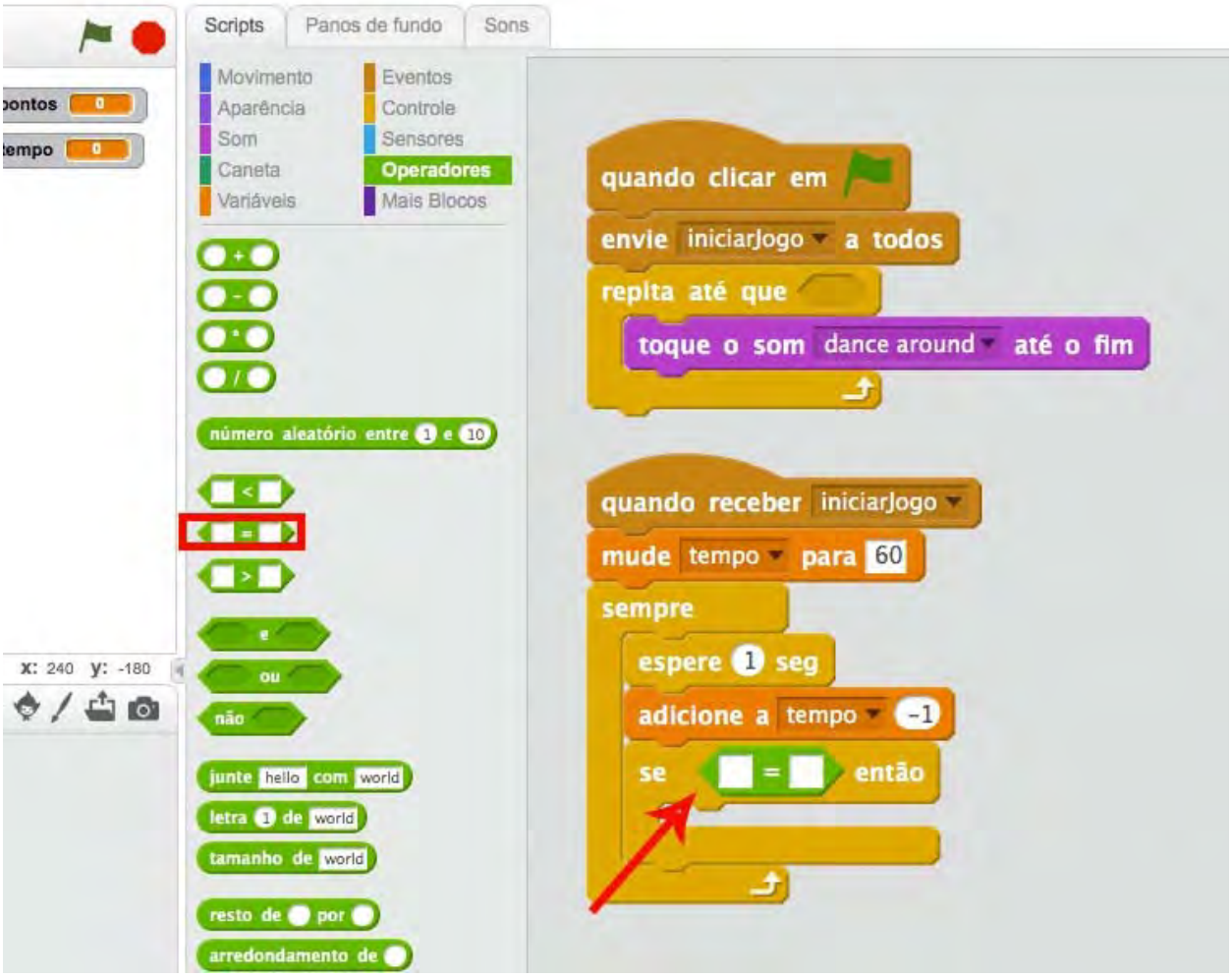
6. Em Variáveis, arraste o bloco adicione a tempo 1. Vamos alterar o valor para -1, pois a cada segundo que ele esperar, acrescenta-se -1 na variável tempo.



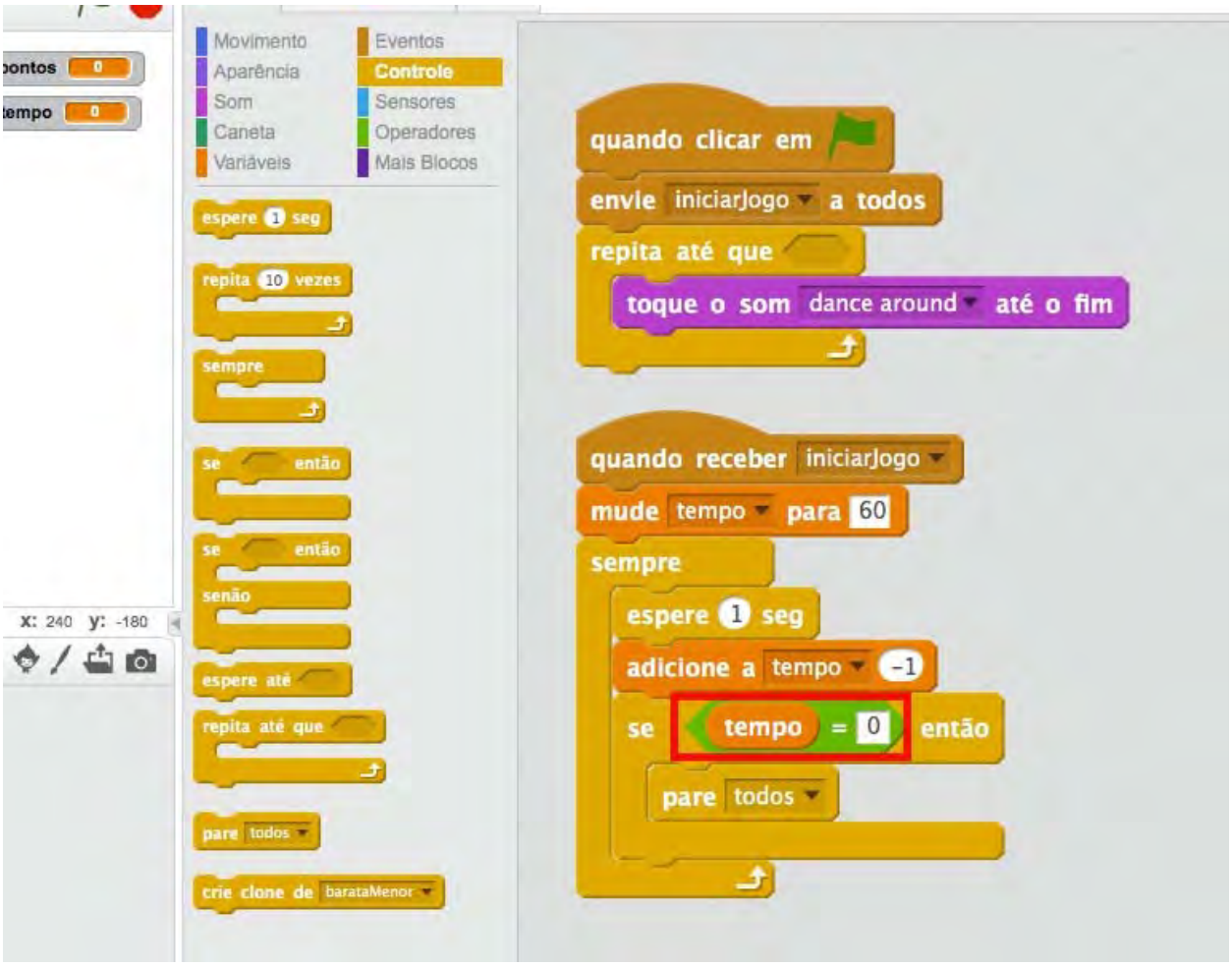
7. Em Controle novamente, arraste o bloco Se então para verificar se o tempo acabou ou não.



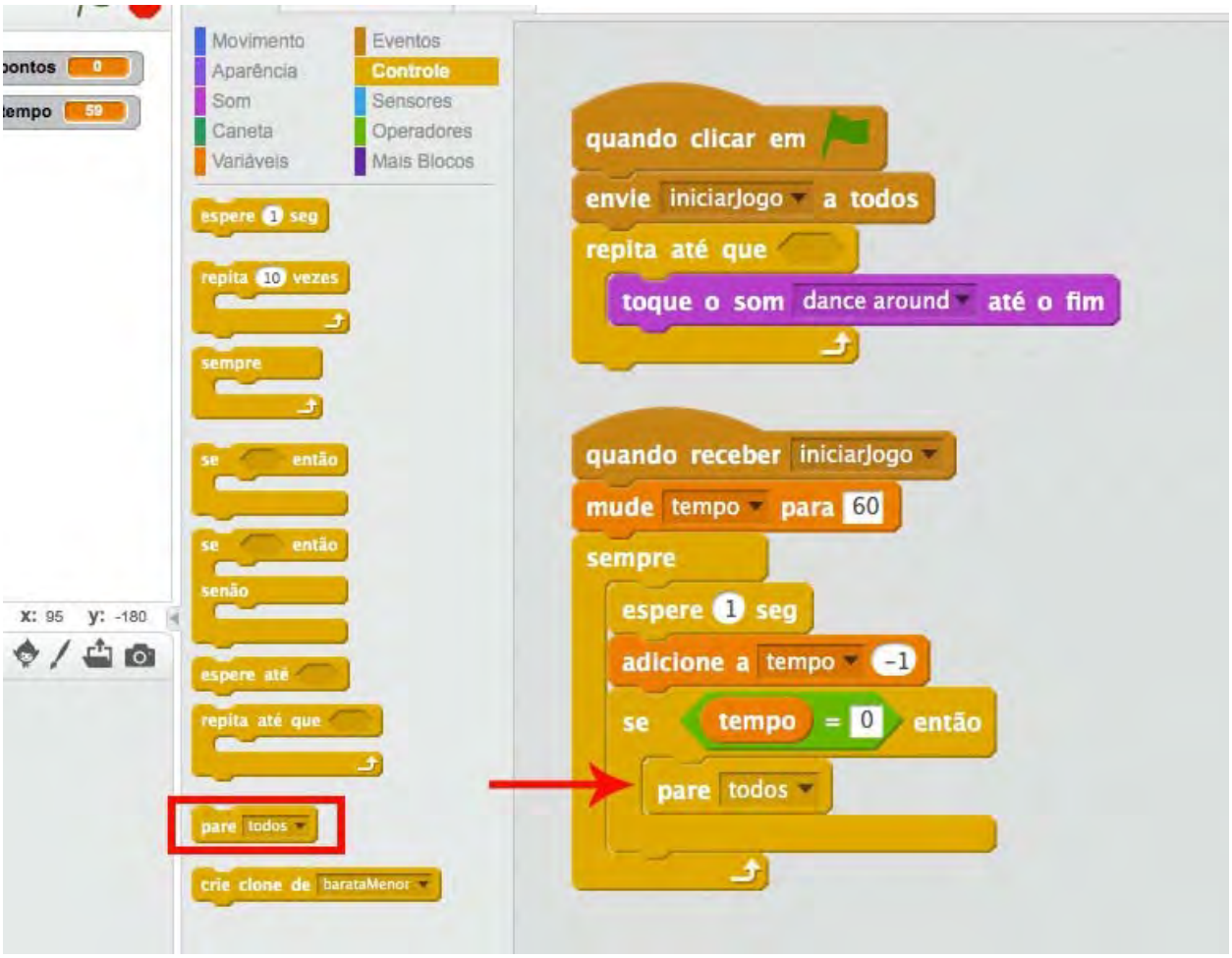
8. Em Operadores, arraste o bloco do operador de igualdade (=) para dentro da área hexagonal do bloco Se.



9. Preencha os espaços do bloco de igualdade com a variável tempo e valor 0. Dessa forma, comparamos o tempo que está na variável com o valor 0. Caso seja 0, então o script para. Veja:



10. Em Controle, arraste também o bloco pare todos. Caso o tempo seja 0, significa que o jogo encerrou para todos os scripts.



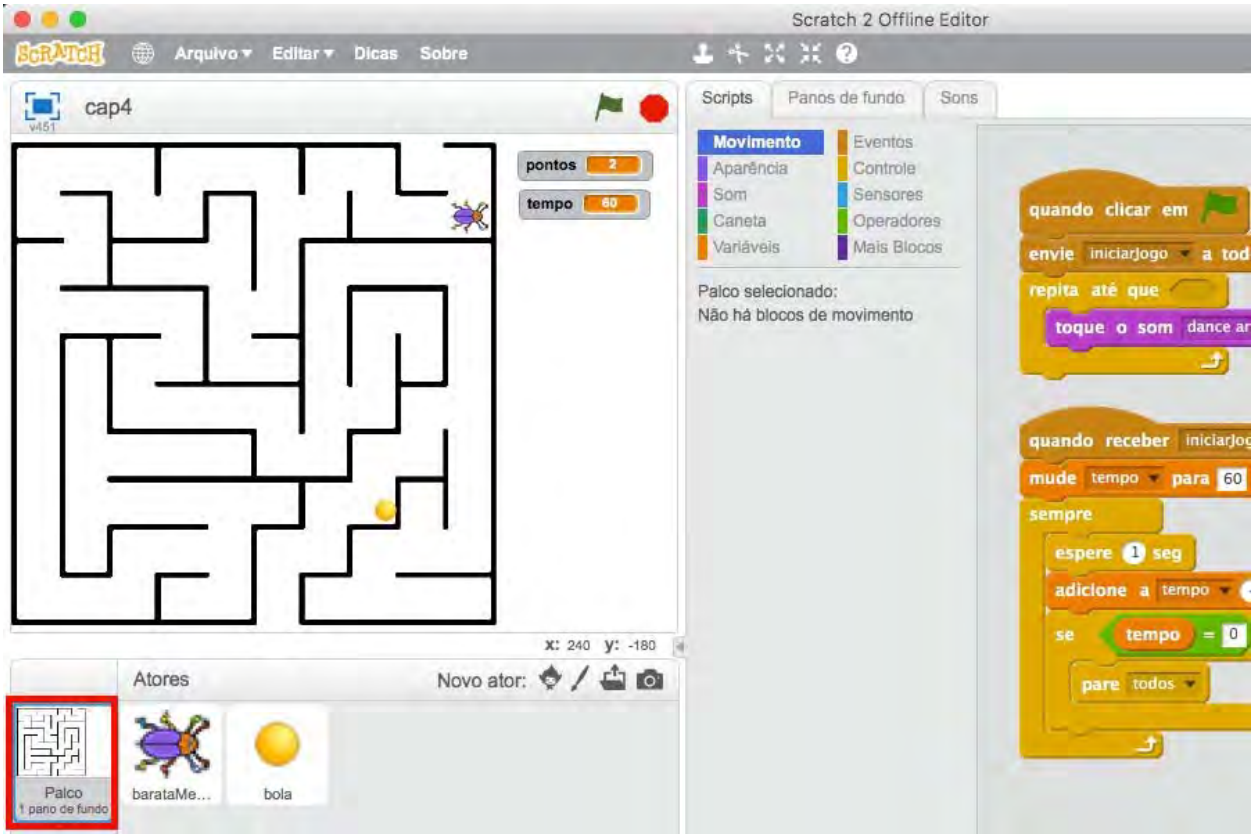
Vamos entender o funcionamento deste script. Quando o jogo iniciar:

- A variável tempo mudará para 60, pois vai decrescer até 0;
- Após isso, entrará num loop infinito;
- O script esperará 1 segundo para executar a próxima instrução;
- Adicionará o valor -1 à variável tempo, pois dessa forma que será decrementado o tempo;
- Se o tempo for igual a 0, o tempo acabou;
- Então, para a execução de todos os scripts, para assim encerrar o jogo.

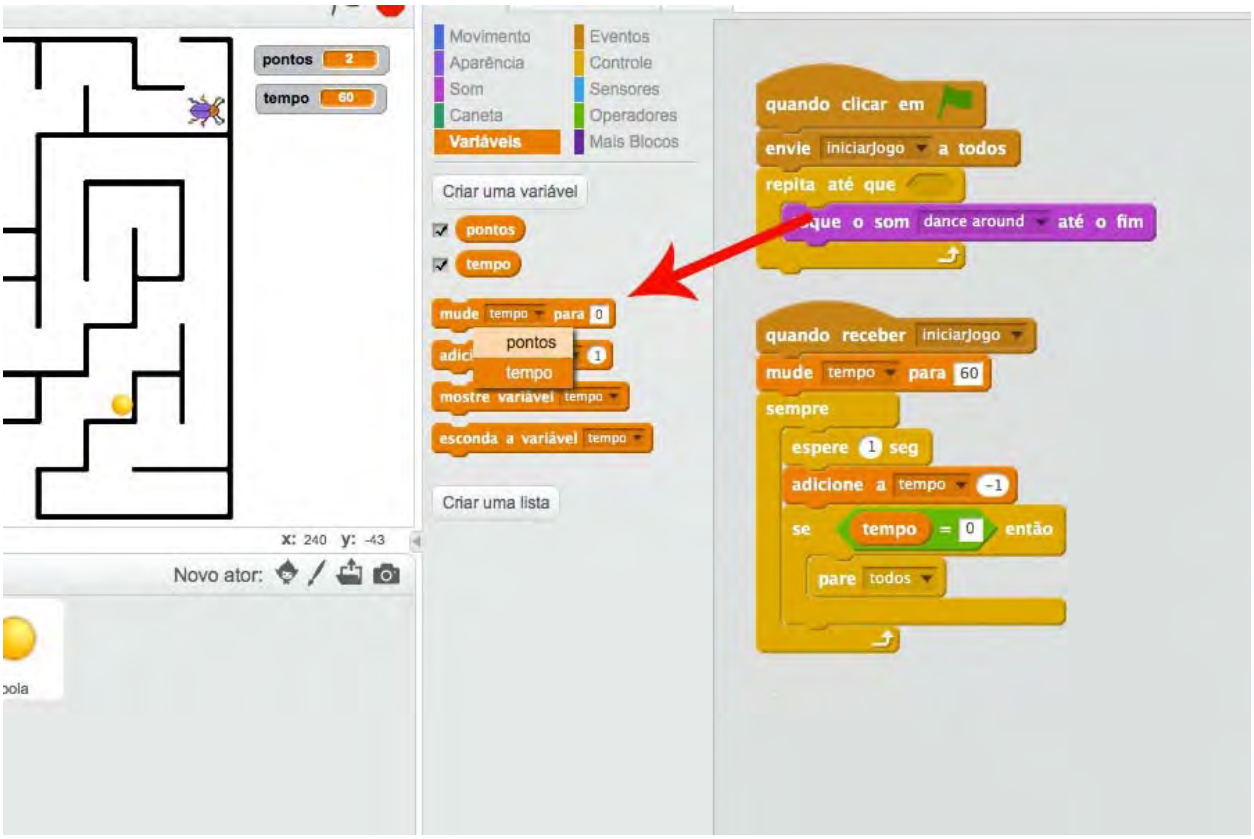
4.5 Reiniciando o jogo

Ainda há um problema com o sistema de pontuação. Se você parar o jogo e reiniciar, verá que a variável pontos não é zerada para um novo jogo. Vamos fazer o seguinte:

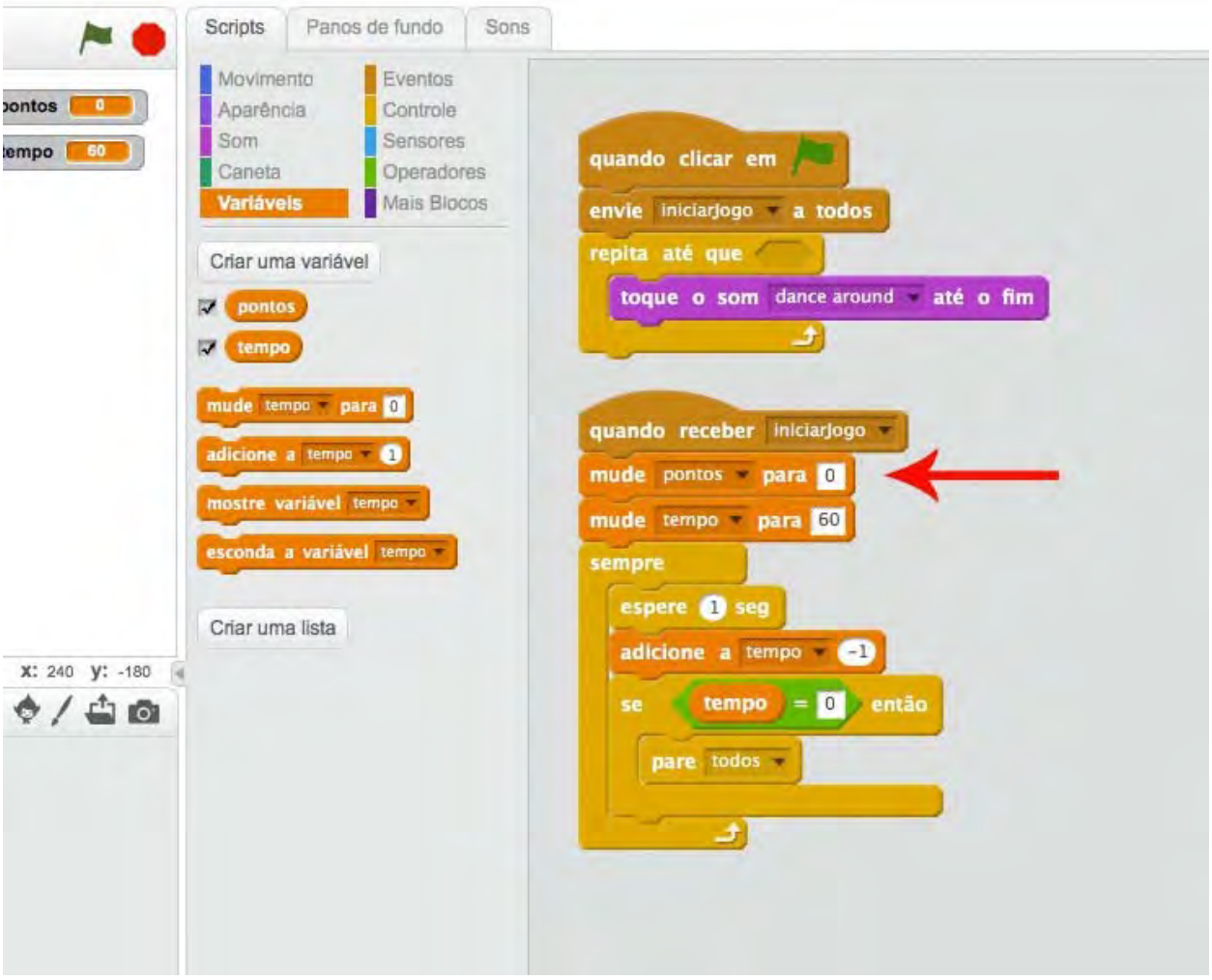
1. Clique em Palco.



2. Em Variáveis, arraste a variável mude pontos para 0. Lembre-se de alterar para pontos.



3. Ficará assim:

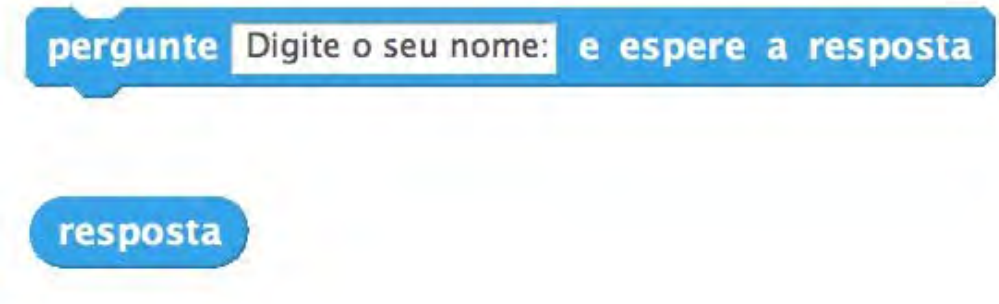


Agora, se você reiniciar o jogo, o tempo iniciará em 60 e a pontuação iniciará em 0. Dessa forma, fazemos com que, ao receber a mensagem iniciarJogo, as variáveis sejam zeradas para poder iniciar um novo jogo.

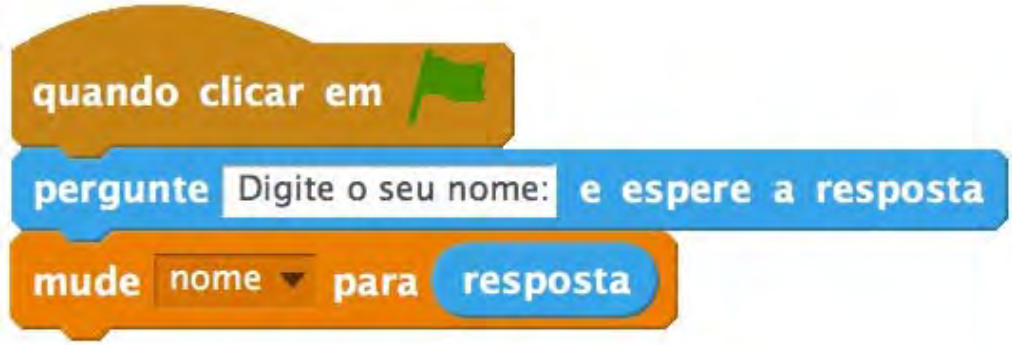
4.6 Obtendo entrada do usuário

Se quisermos armazenar nomes ou quaisquer dados fornecidos pelo usuário, devemos utilizar alguns comandos de entrada que permitirão guardar estas informações em uma variável. Por exemplo, se precisarmos armazenar o nome ou data de nascimento, teremos de fazer duas solicitações: uma requisitando o nome para logo após o programa ficar esperando o usuário digitar algo e confirmar, e outra a data de nascimento para logo após também ficar esperando o usuário digitar algo e confirmar.

são os blocos:



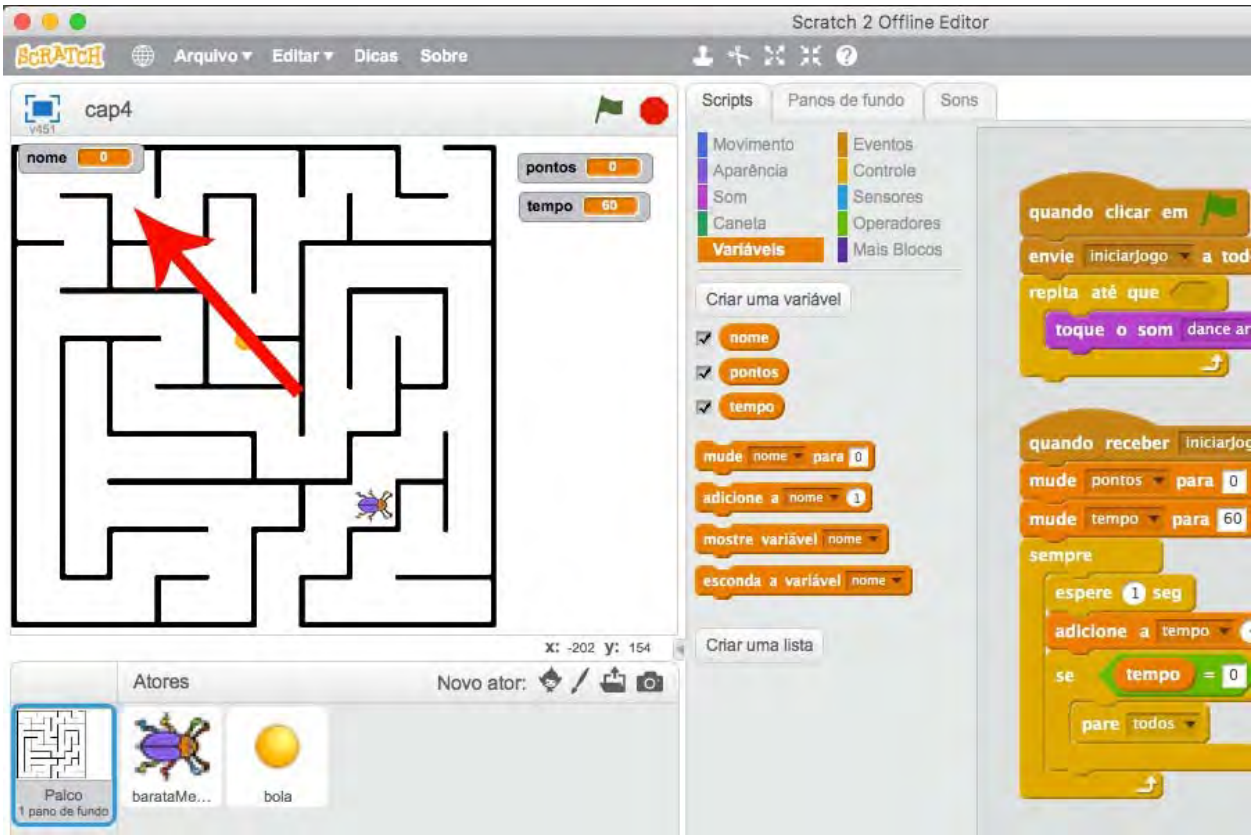
O bloco de pergunta permite digitar o que será solicitado, e o bloco de resposta serve para armazenar o que o usuário digitou. Entretanto, a cada solicitação, o bloco de resposta será alterado, sendo assim, o correto é guardarmos em alguma variável. Dessa maneira:



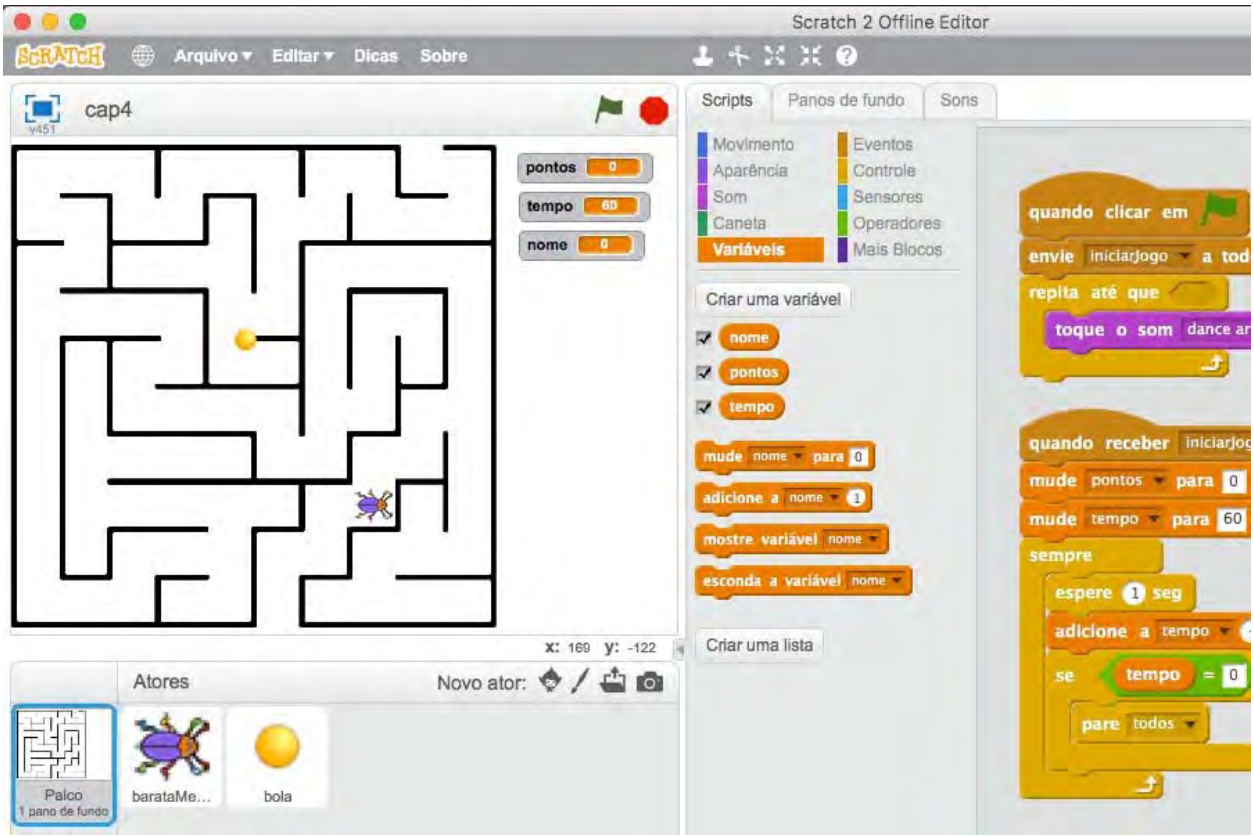
Para finalizar, vamos pedir o nome do jogador e armazená-lo em uma variável.

1. Ainda em Palco, crie outra variável chamada nome.

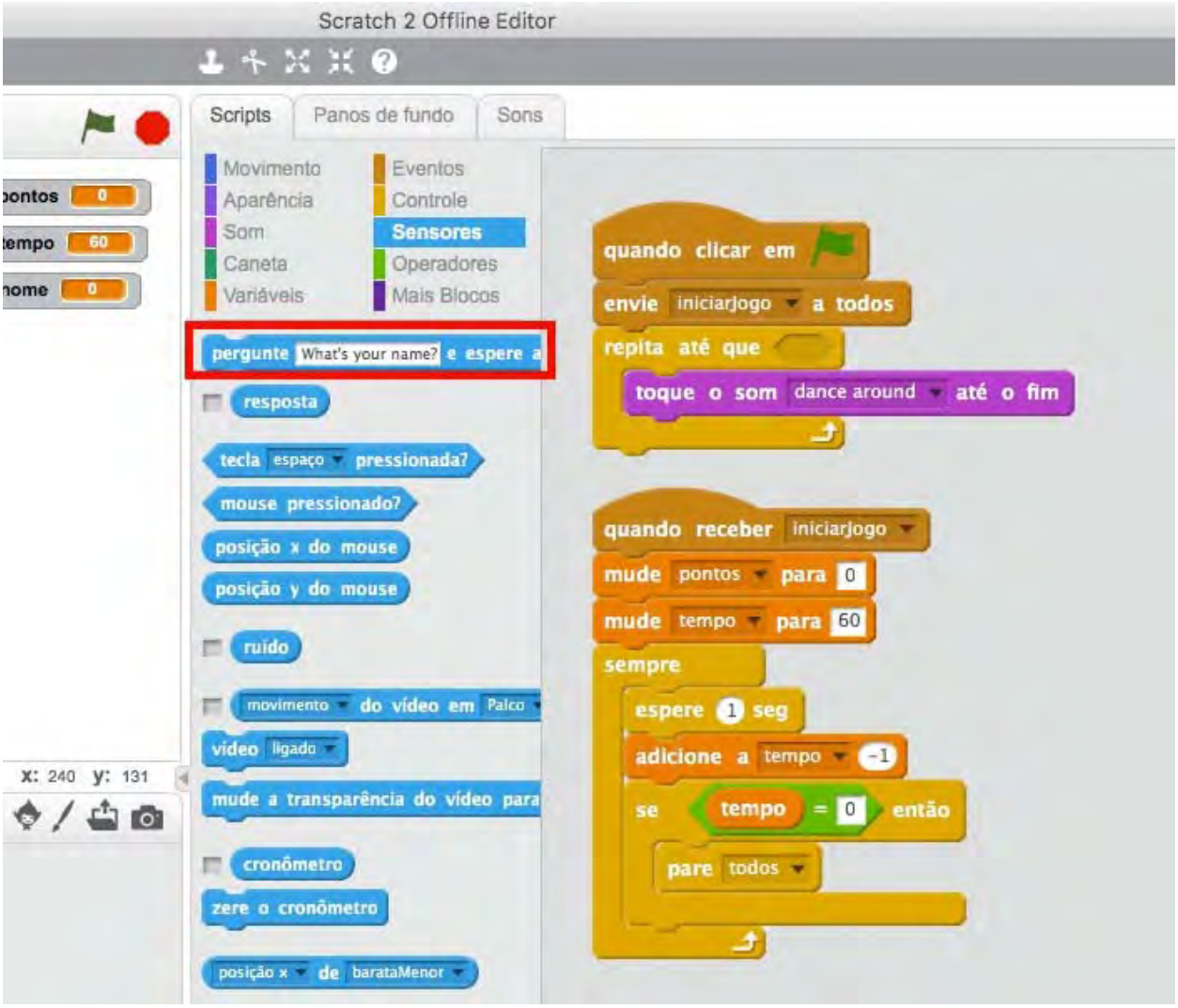
2. Veja que ela aparece do lado esquerdo, arraste-a para o lado direito para ficar mais organizado.



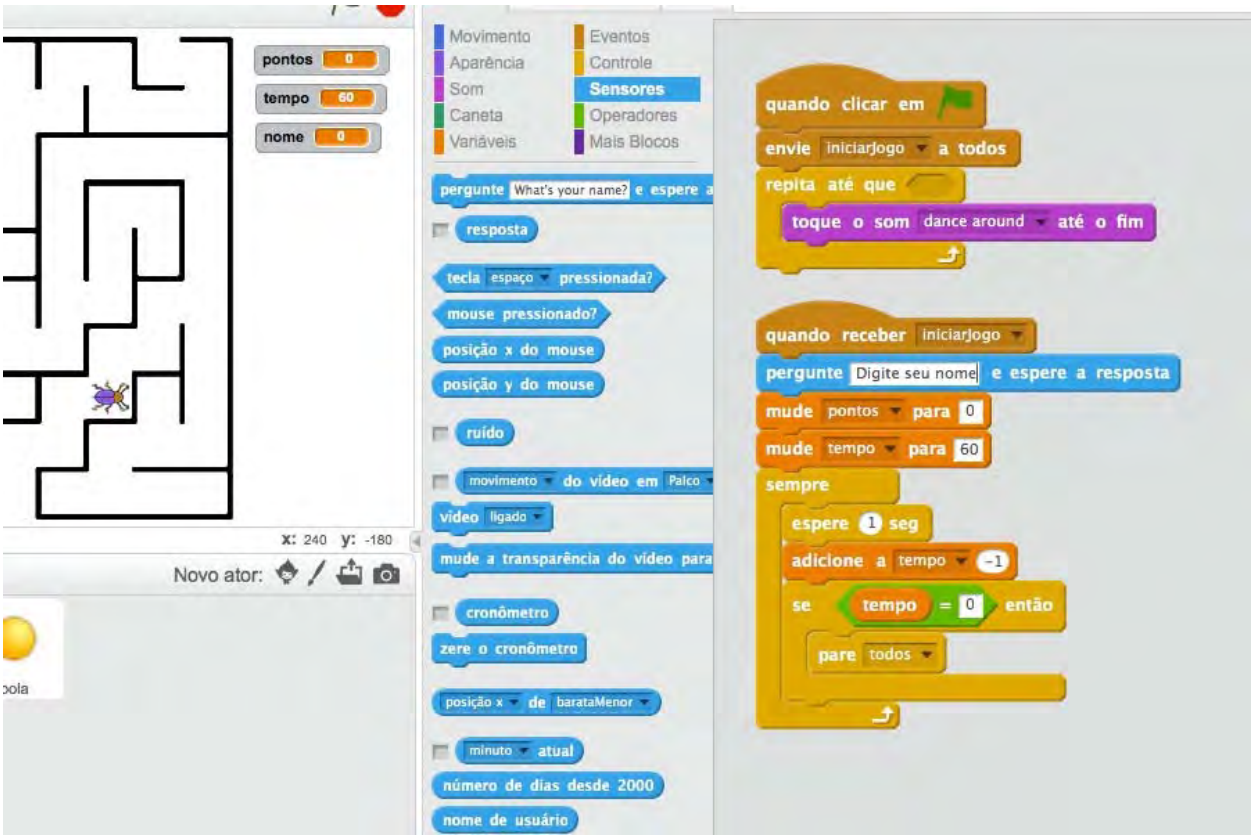
3. Ficará assim:



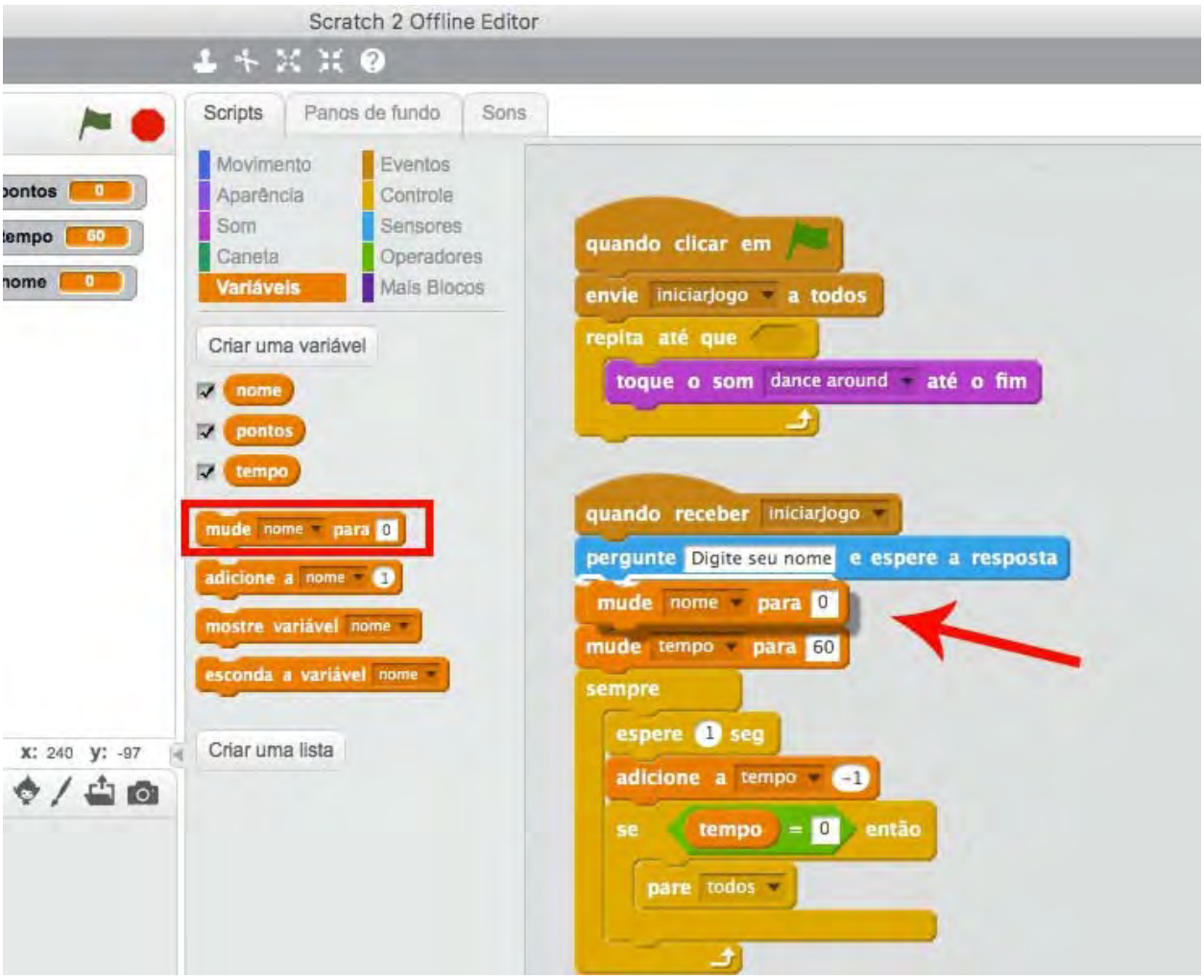
4. Vá em Sensores e arraste o bloco pergunte para baixo do bloco quando receber iniciarJogo. Assim conseguimos obter algum dado fornecido pelo usuário (no caso, o nome).



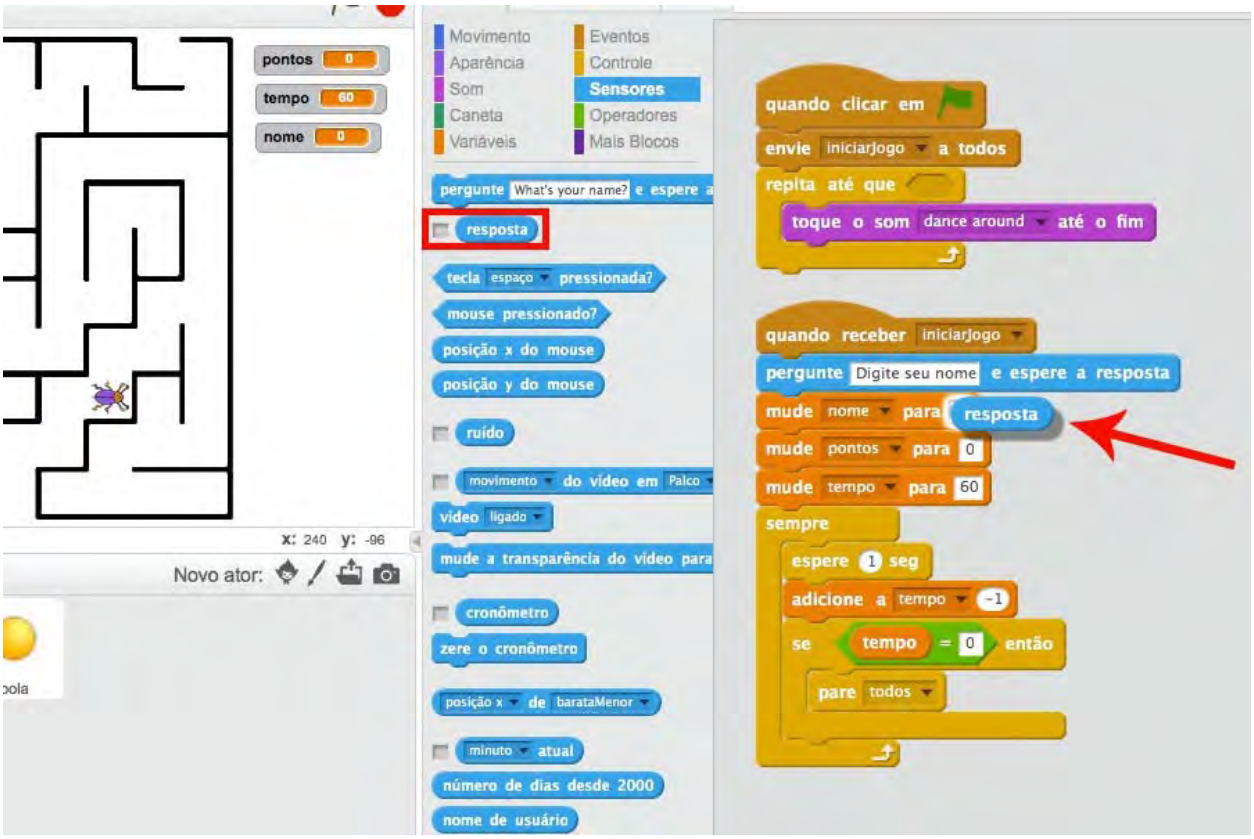
5. Ficará assim:



6. Arraste o bloco mude para baixo do bloco de pergunta. Lembre-se de alterar no bloco para nome. Dessa forma, o nome do jogador será armazenado na variável escolhida, no caso *nome*.



7. Novamente em Sensores, arraste o bloco resposta para dentro da área do bloco mude.



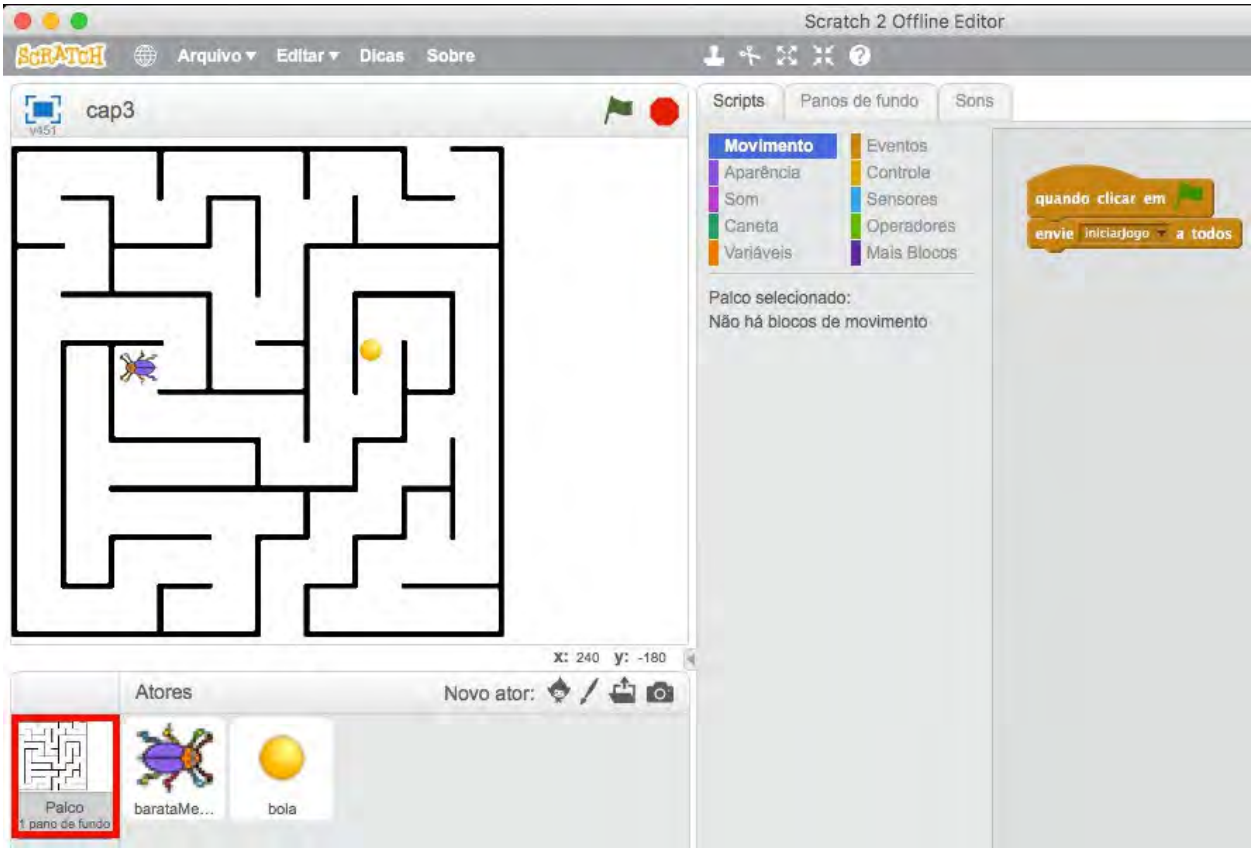
8. O script pronto ficará assim:

Este é o nosso contador de tempo que pede nome de usuário. Legal, né? Para ficar ainda mais interessante que tal se colocarmos um fundo musical? Vamos ver como fazer isso na próxima seção.

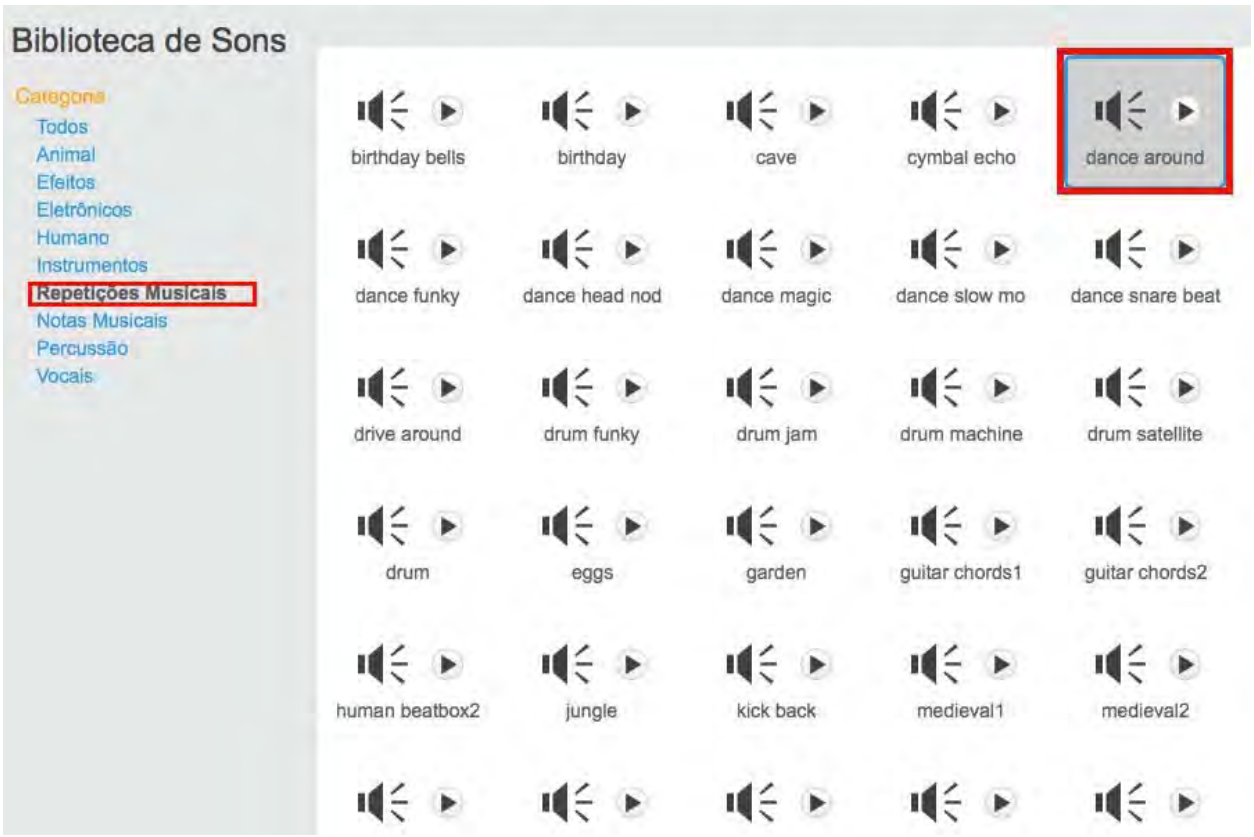
4.7 Implementando um fundo musical

Aprendemos no capítulo anterior como deixar o jogo mais divertido através da adição de sons, e vimos neste capítulo como utilizar variáveis. Com essas duas ferramentas em mãos, podemos agora implementar um fundo musical. Para isso, vamos utilizar o bloco de repetição, chamado repita até que.

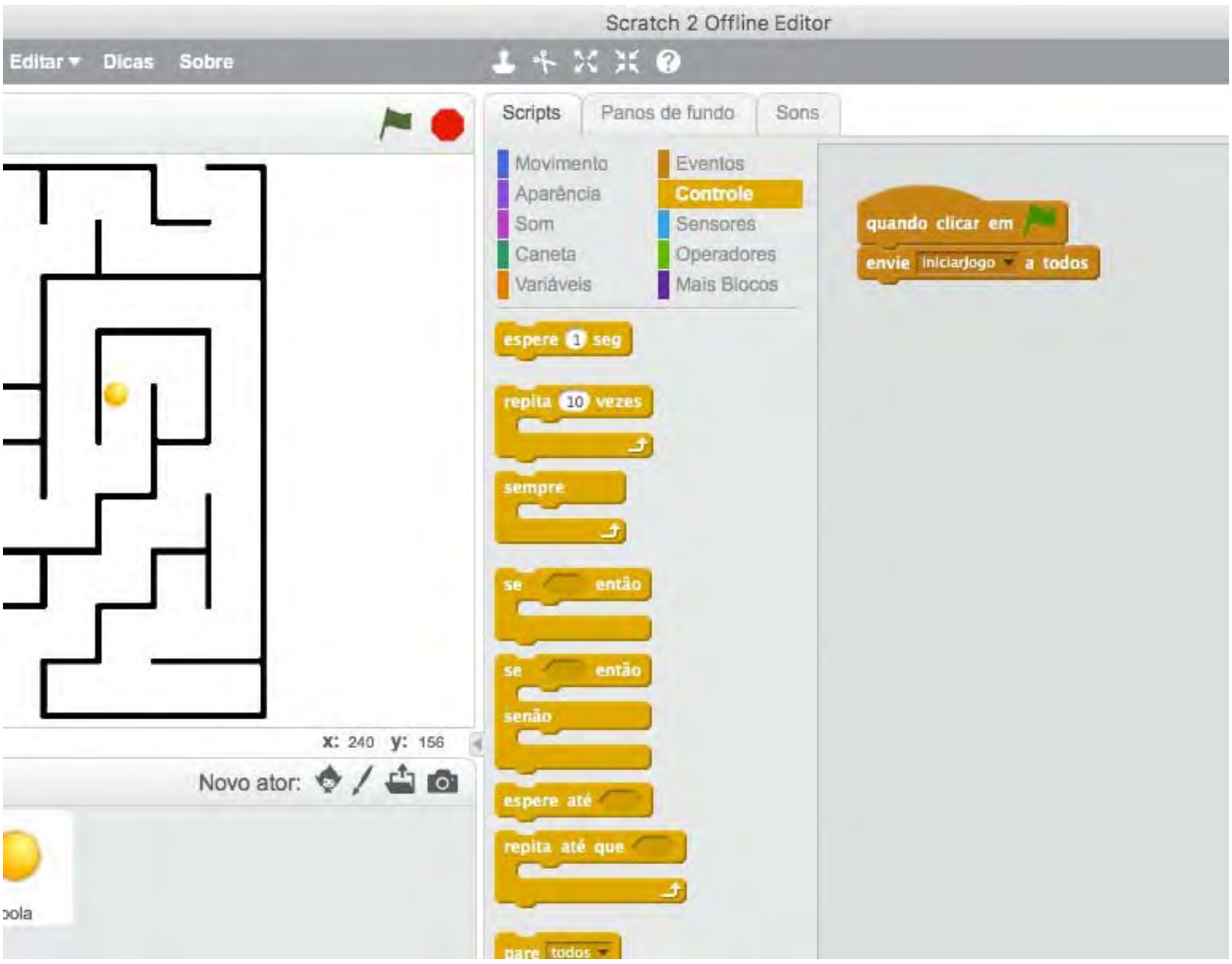
1. Clique em Palco.



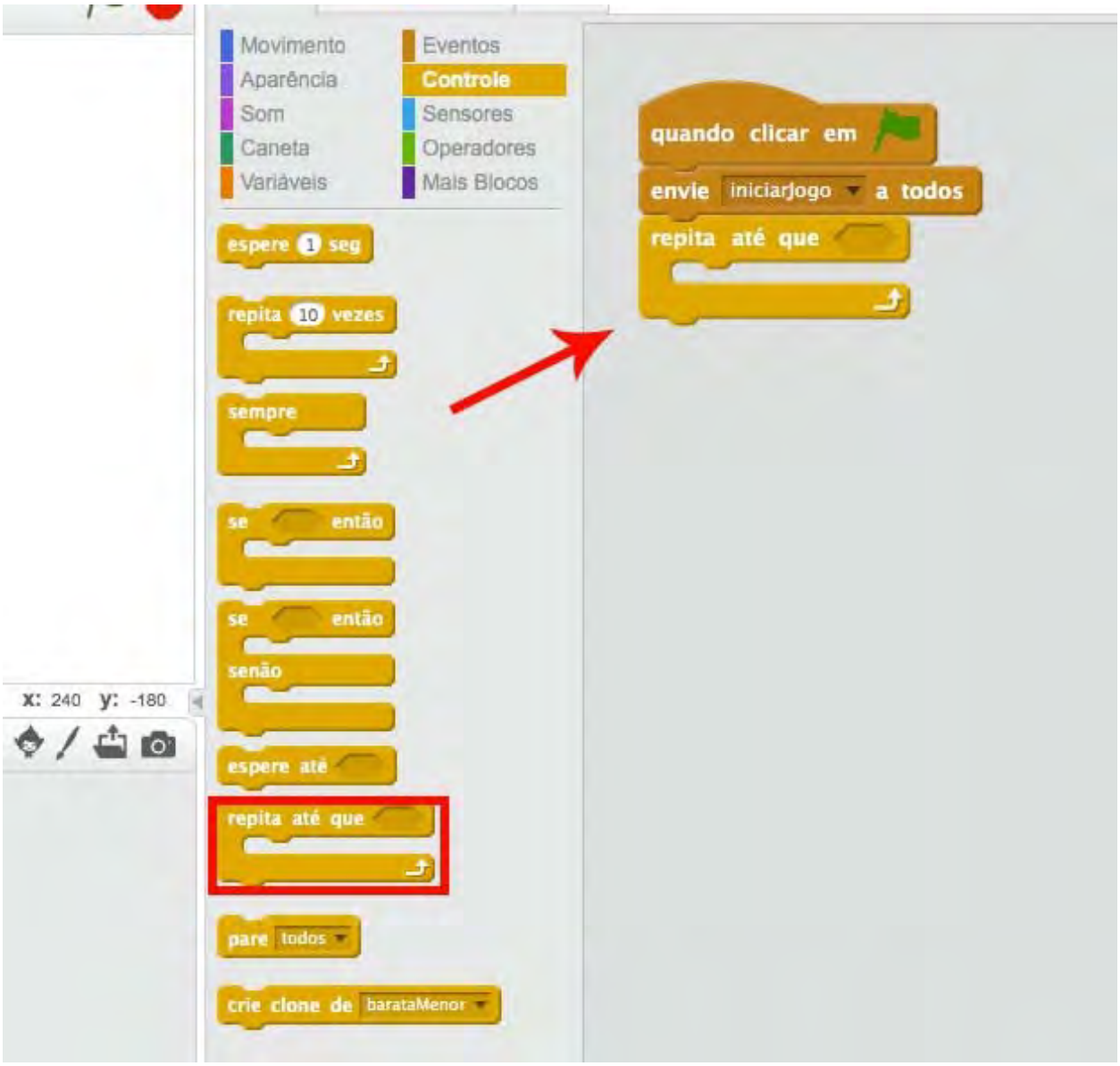
2. Em Repetições Musicais, selecione o som dance around. Lembre-se sempre de que você pode escolher seus próprios sons, mas, para fins didáticos, o dance around é o que melhor se encaixa ao jogo.



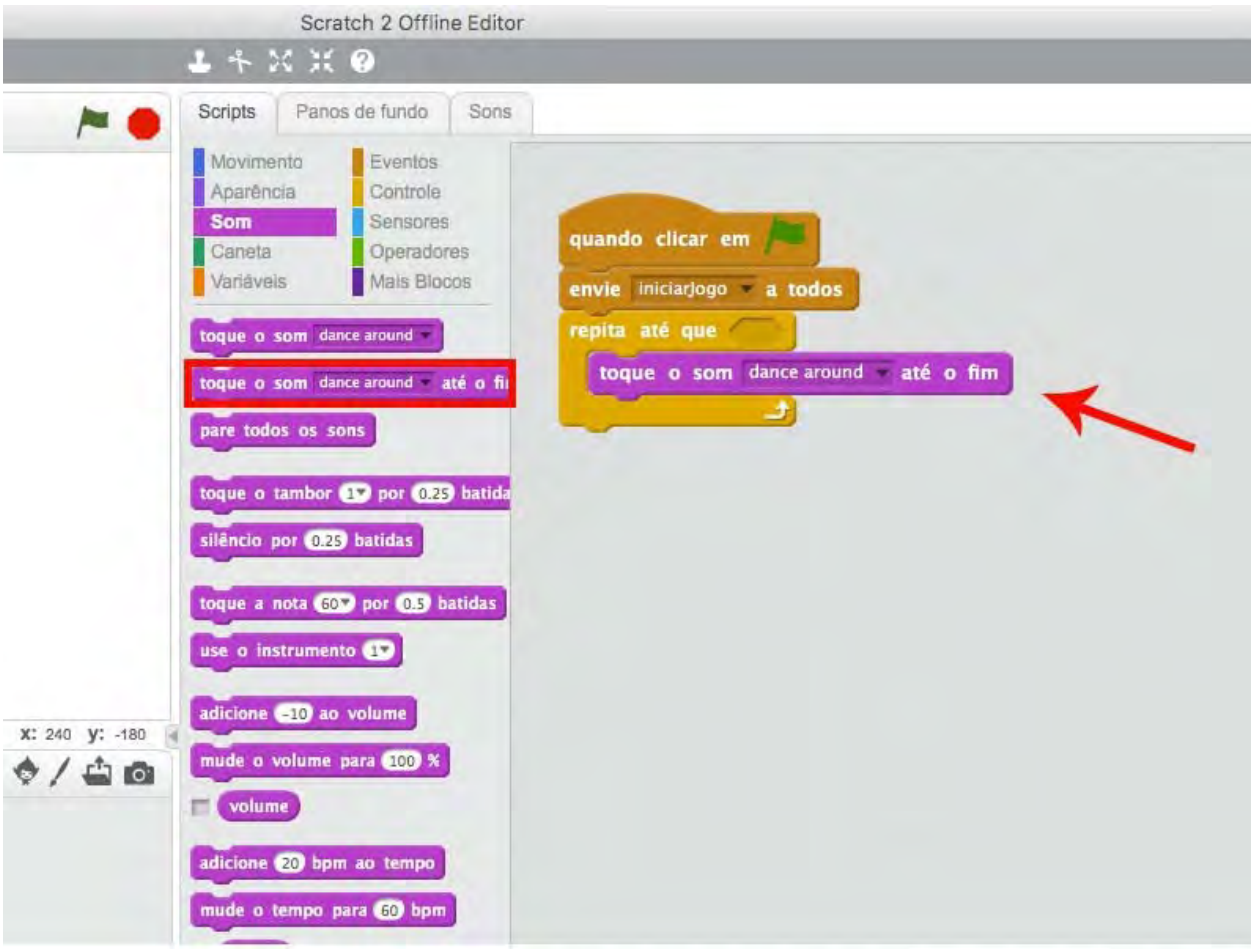
3. Vá para a aba Scripts e em Controle para encontrar os blocos de repetição.



4. Arraste o bloco repita até que.



5. Agora em Som, arraste o bloco toque o som até o fim.



A partir de agora, entramos na reta final do desenvolvimento do nosso jogo. A continuação da implementação do fundo musical você verá no próximo capítulo, que utilizará uma ferramenta chamada procedimentos, ou funções.

4.8 Conclusão

- Neste quarto capítulo, vimos variáveis globais, que são variáveis que podem ser acessadas por todos os atores.
- Vimos as variáveis locais que podem ser acessadas por somente um ator.
- Demos continuidade ao projeto labirinto e implementamos um sistema de pontuação e tempo.

A seguir, veremos procedimentos, que são blocos personalizados que permite o reaproveitamento de código, eliminando a necessidade de repetir um mesmo código várias vezes e em vários locais.

CAPÍTULO 5

Procedimentos e funções

5.1 O que são procedimentos e funções?

Imagine que alguém peça a você que crie um programa que calcule a média da sua turma na escola. Ou seja, você terá de pegar as notas de todos os alunos da sua turma, somar e depois dividir pela quantidade de alunos. Quantos estudantes existem na sala?

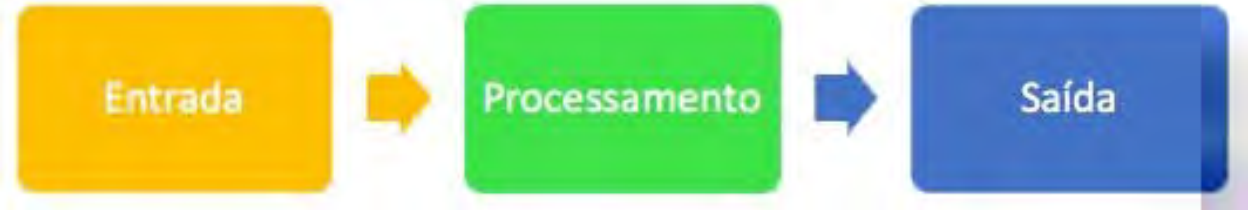
Teríamos então de escrever o código responsável pela entrada de todos os valores, criar variáveis para todas as notas, realizar a soma e, enfim, calcular a média. Você consegue imaginar o quanto isso seria trabalhoso? Precisamos repetir o código para 30, 40 ou 50 alunos. Isto o torna bagunçado e de difícil manutenção.

O que vamos aprender agora são os procedimentos e funções. Eles permitem definir um nome para um trecho de código e, com base nesse nome, usá-lo em outros locais. Portanto, procedimentos são trechos de códigos que podem ser reaproveitados em outros locais.

Veja o código anterior. O primeiro bloco, de cor roxa, define o nosso procedimento. Demos o nome de exemplo_procedimento. Abaixo dele, está o código que será executado cada vez que chamarmos o procedimento exemplo_procedimento.

Anteriormente nós temos o bloco que chama o procedimento exemplo_procedimento. Então, toda vez que clicarmos na bandeira, o procedimento exemplo_procedimento será executado. Bem simples, não? Você consegue visualizar o quanto isso facilita nossa vida?

Há um outro tipo de procedimento que se chama função. A função é diferente do procedimento que retorna alguma coisa. Por exemplo, imagine uma máquina. O que toda máquina tem:



- Entrada — onde você coloca o que será processado;
- Processamento — onde o que foi colocado será transformado em algo;
- Saída — o resultado do que foi transformado/processado.

As funções funcionam como máquinas. Você coloca alguma coisa, ele processa/transforma essa coisa e retorna o resultado disso. Podemos ver também como funções matemáticas. Você se lembra das funções da aula de matemática?

Imagine uma função $f(x) = x + 1$, e a entrada desta máquina é o X. O processamento é o $x + 1$, e a saída é o resultado desta soma. Imagine as funções então como máquinas. Agora as possibilidades que nós temos com funções são infinitas.

Podemos criar nosso programa em partes, em que cada função realiza uma parte do programa. Se tivermos um programa muito complexo, poderíamos simplesmente dividi-lo em partes menores e programar cada parte do nosso programa com funções. Fantástico, não é mesmo?

Então lembre-se:

- **Procedimentos:** blocos de código que não retornam nada;
- **Funções:** blocos de código que retornam alguma coisa (um número, uma letra, valor lógico etc.).

5.2 Labirinto

Partindo para a prática agora, vamos continuar nosso projeto de labirinto e adicionar um ponto de chegada no jogo. Usaremos procedimentos para definir se o jogador ganhou ou não o jogo. Dessa forma, ele vai exibir e tocar algum som avisando que o jogador ganhou ou não.

Quais seriam as possiblidades ao chegar no final? Lembre-se de que o objetivo do jogo é capturar a quantidade máxima de bolinhas em menor tempo. Logo, as possibilidades ao chegar no final do jogo são:

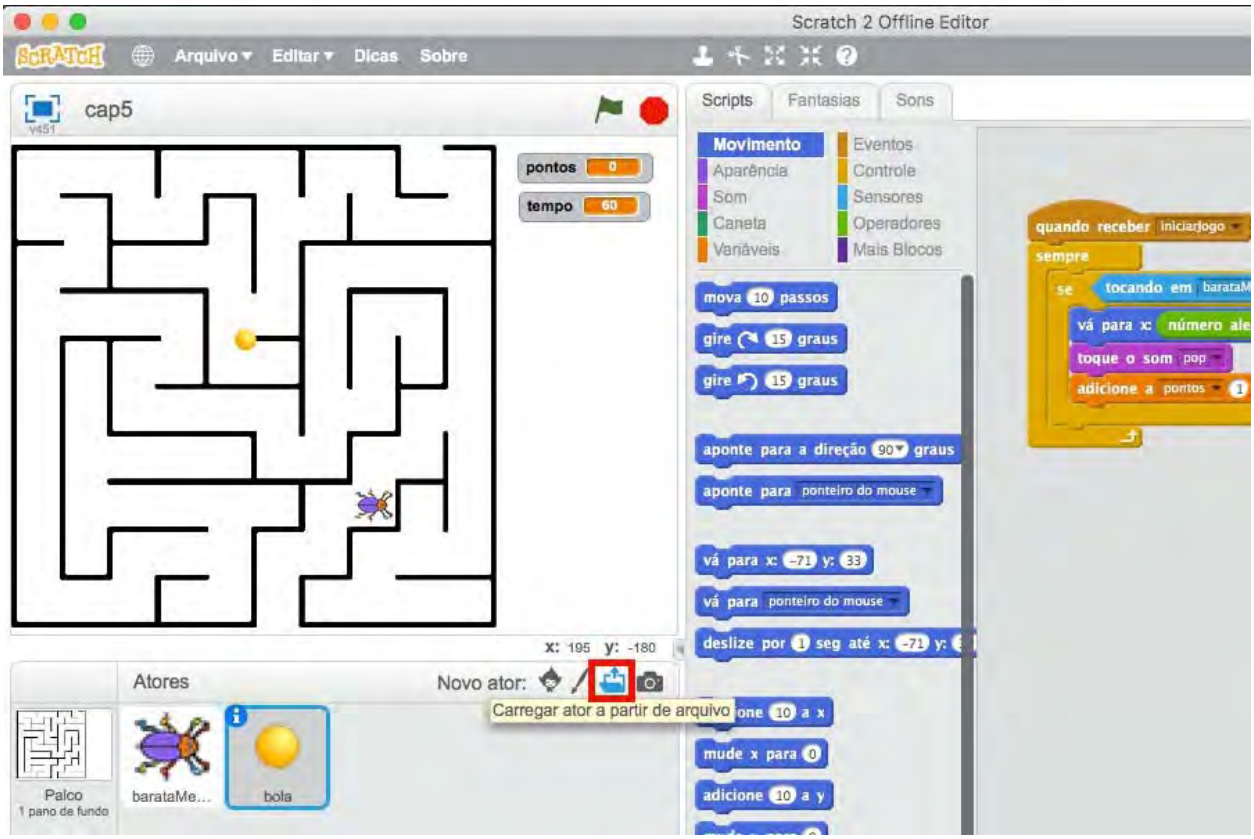
1. **Tempo** > 0 e **Pontos** > 0
significa que ele está dentro do tempo permitido e fez pelo menos 1 ponto.
Resultado: ***GANHOU***

2. **Tempo** > 0 e **Pontos** = 0
significa que ele ainda está dentro do tempo permitido porém não fez pontos.
Resultado: ***PRECISA CAPTURAR BOLINHAS***

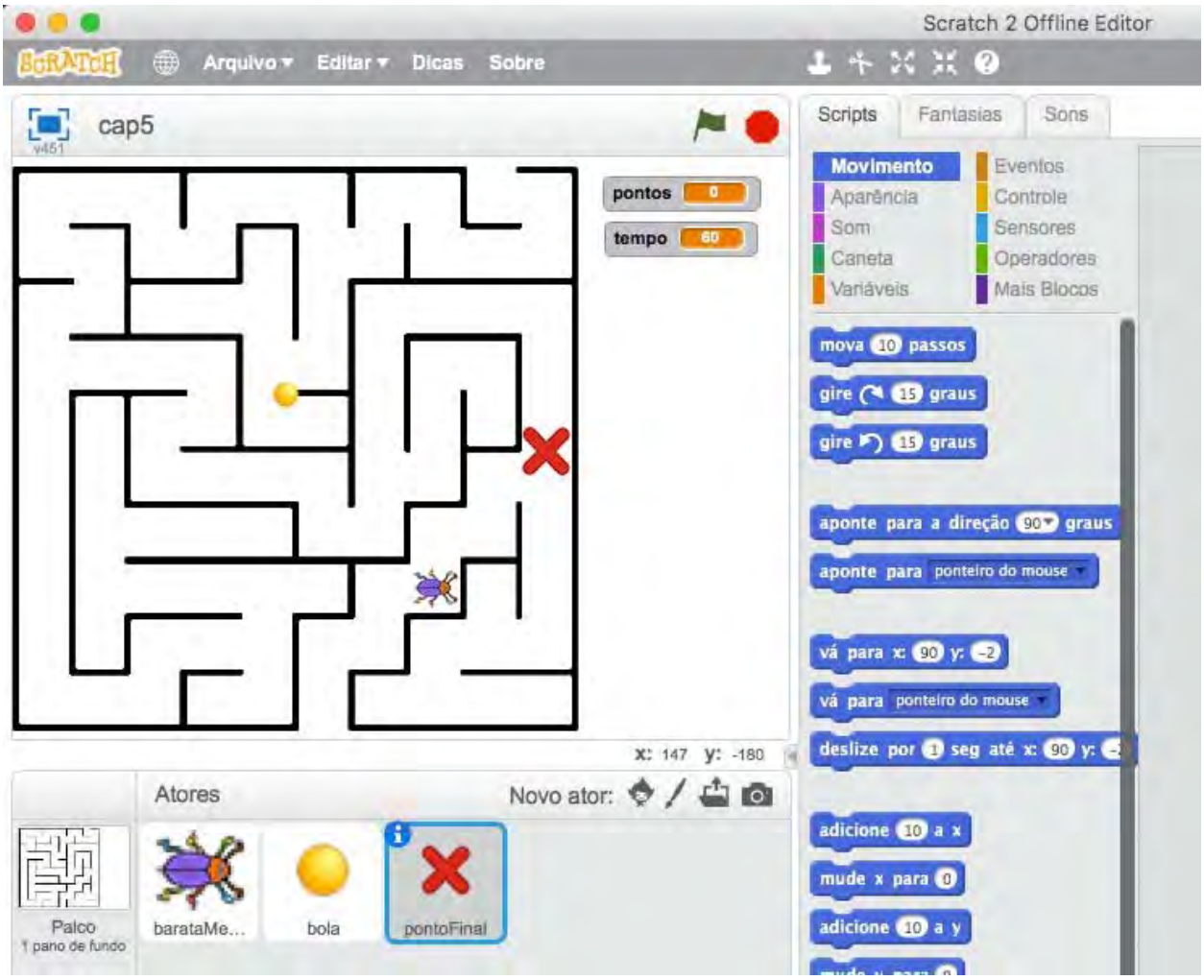
3. (**Tempo** = 0 e **Pontos** > 0) ou (**Tempo** = 0 e **Pontos** = 0)
significa que acabou o tempo, porém fez pontos ou não fez pontos. Em qualquer um dos casos ele não ganhou, pois acabou o tempo.
Resultado: ***PERDEU***

Vamos desenvolver o código responsável por fazer esta verificação ainda neste capítulo.

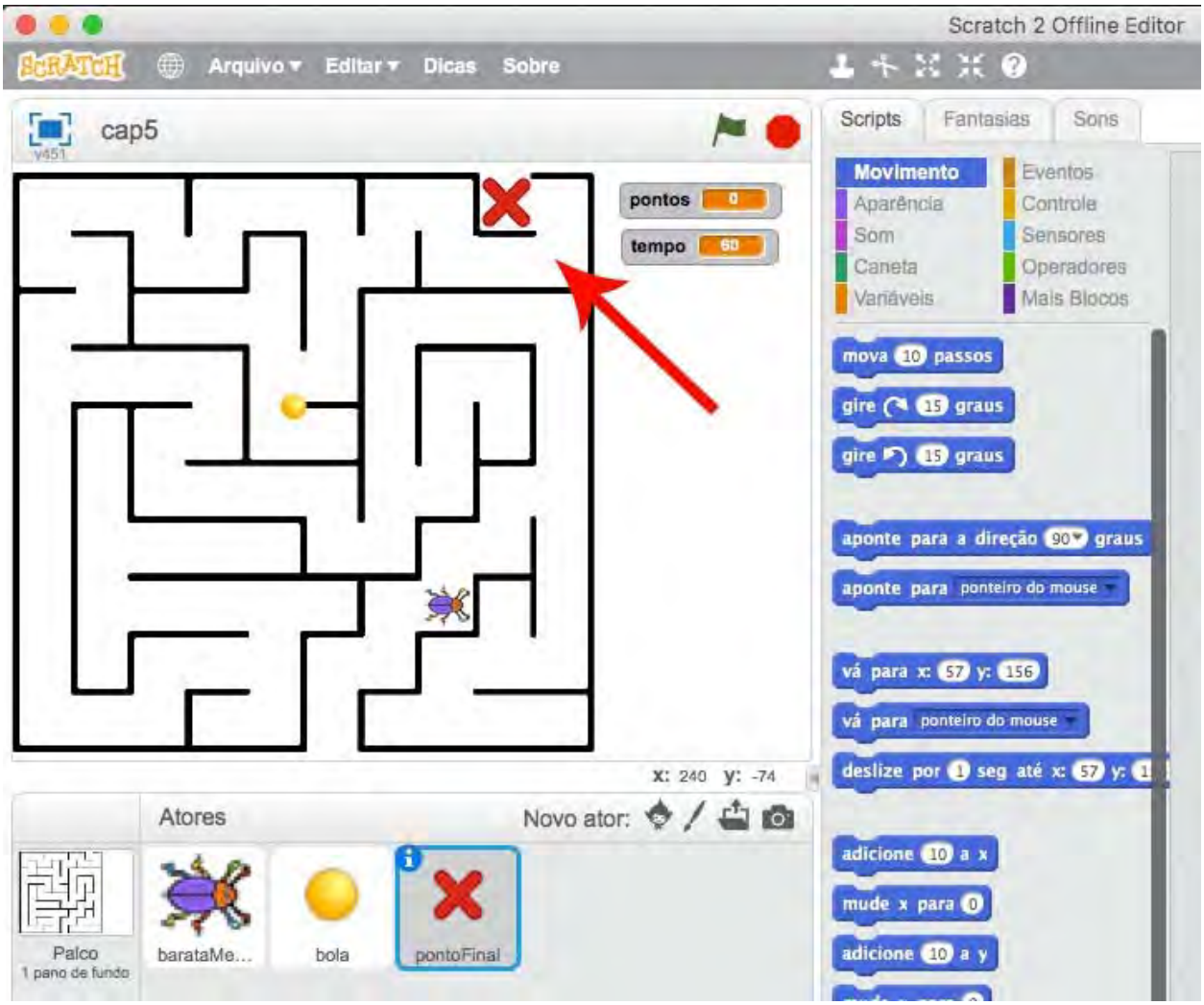
1. Abra o arquivo salvo no capítulo anterior.
2. Para representar onde o jogador deve chegar, vamos adicionar um novo ator que representará o ponto final. Clique em Carregar ator a partir de arquivo e carregue o arquivo PontoFinal na pasta em que você baixou as imagens.



3. Após carregado, ficará mais ou menos assim, posicionado de forma totalmente aleatória:

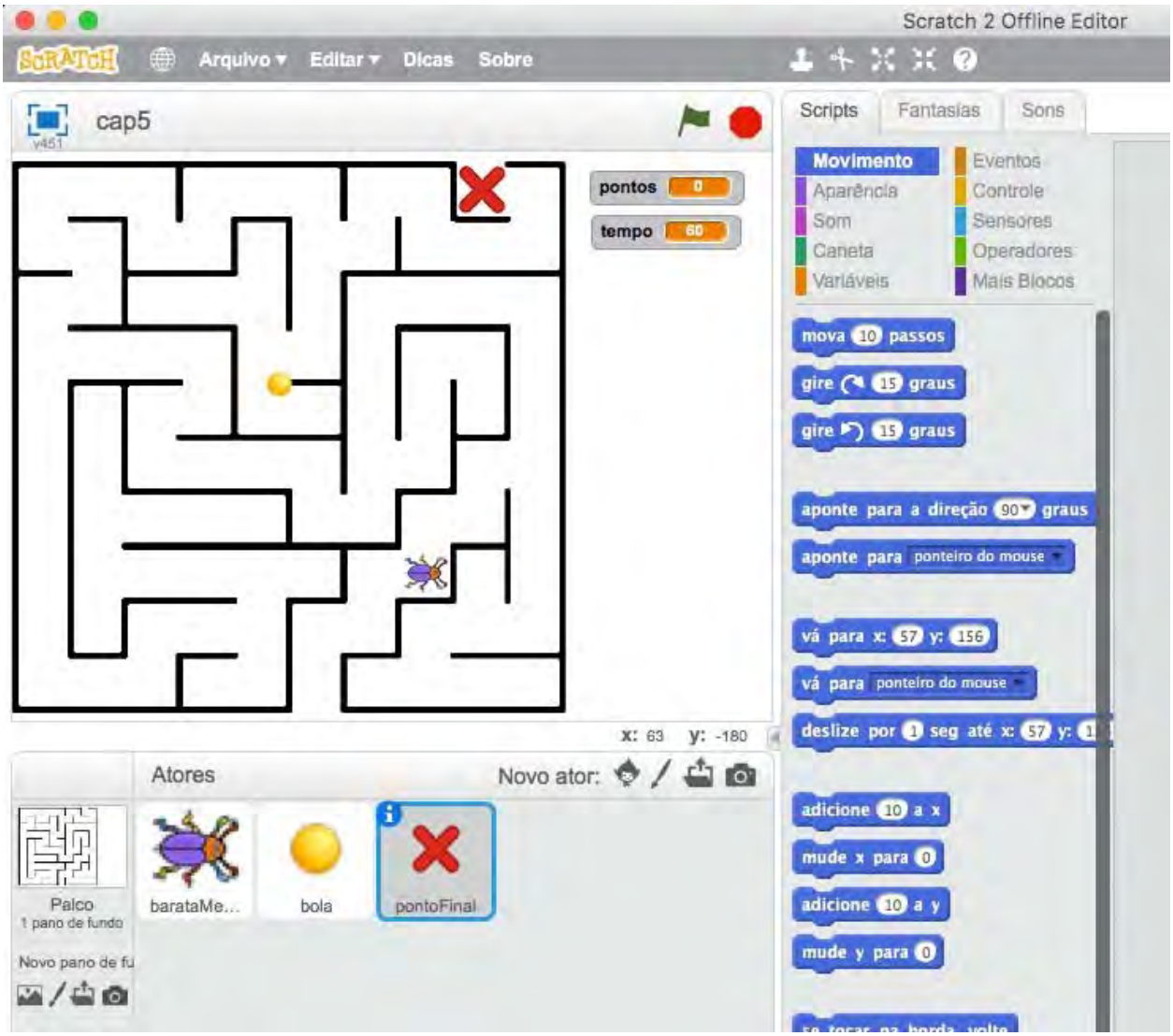


4. É necessário que ele fique posicionado no ponto de chegada do labirinto. Se por acaso você utilizar outro mapa, fará sentido você posicionar onde fica o ponto de chegada. Mas neste caso, vamos posicionar dessa forma:

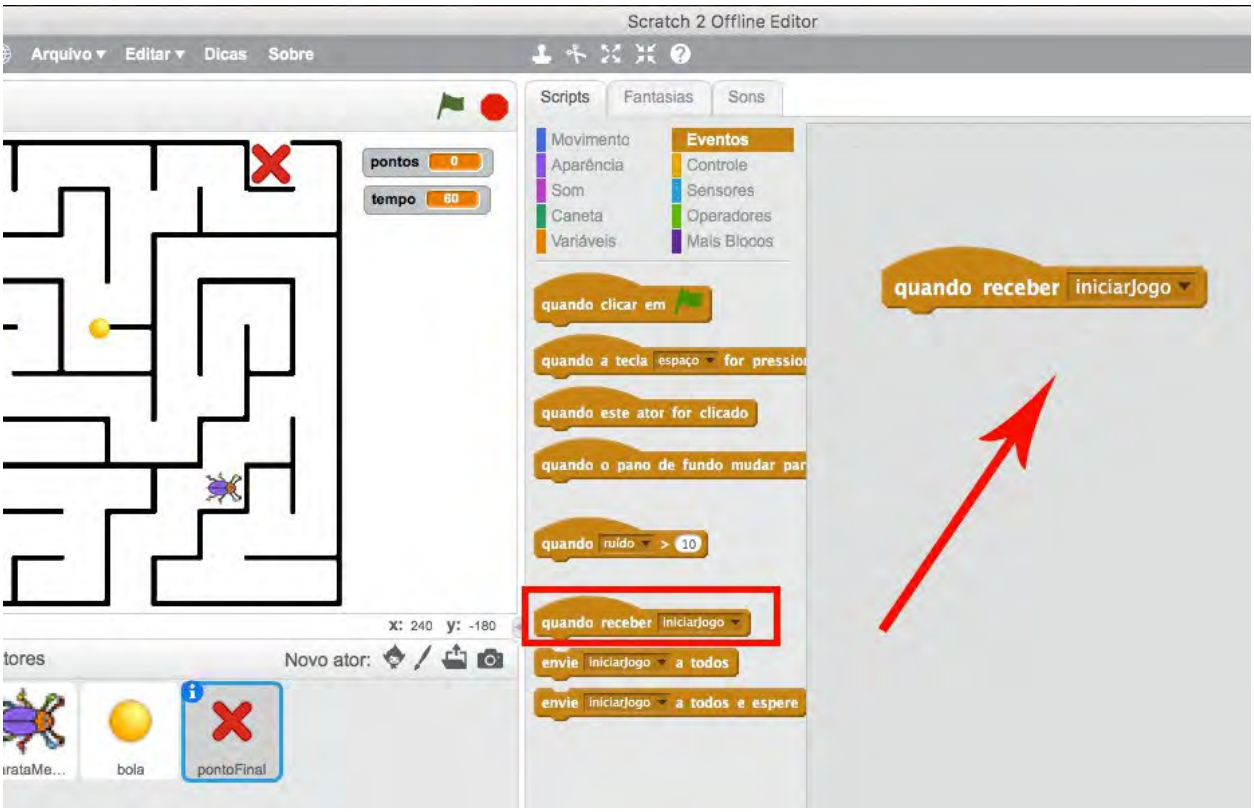


A partir de agora, vamos programar a verificação para saber se o jogador ganhou ou não, de acordo com todas as possibilidades vistas anteriormente.

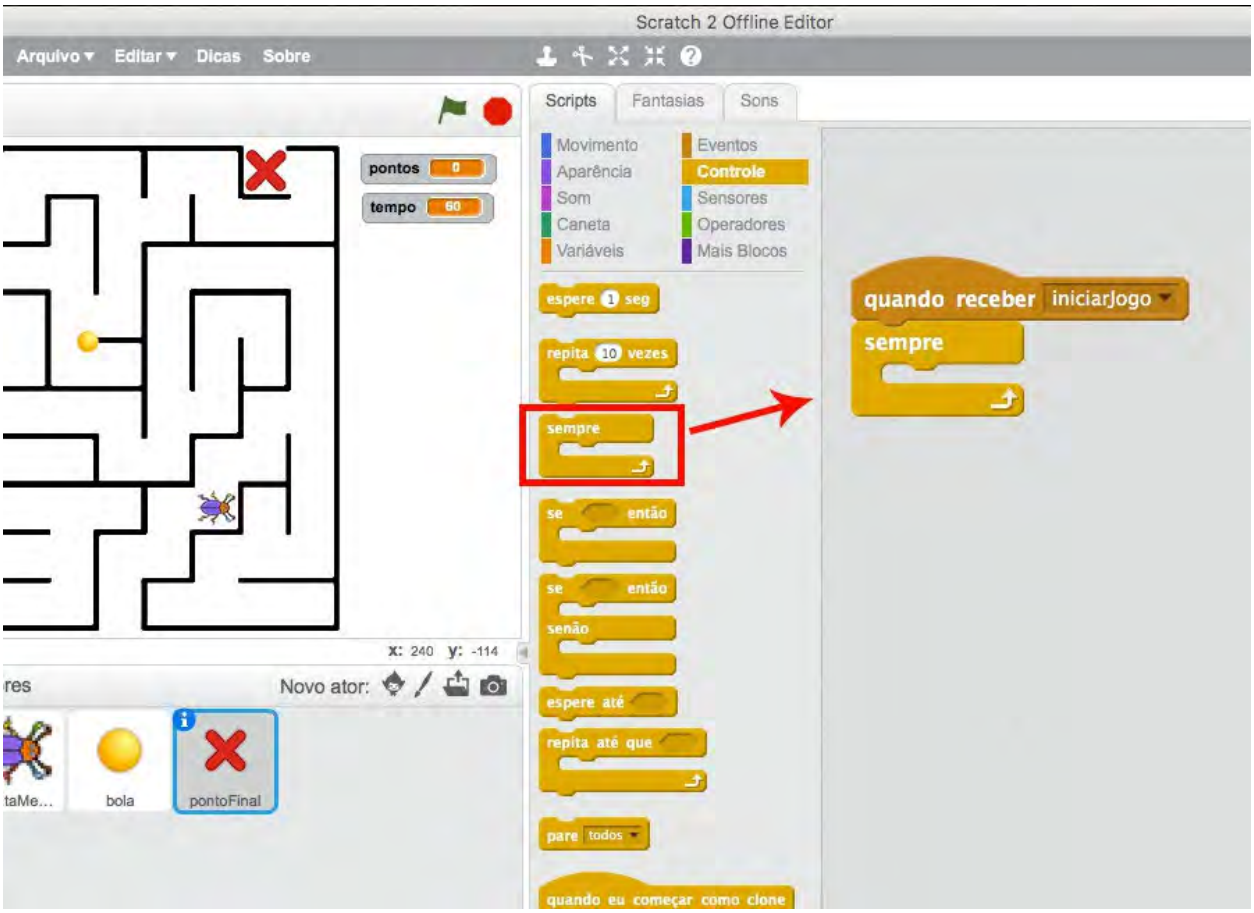
5. Com o ator pontoFinal posicionado, selecione-o e vá para a Área de Scripts.



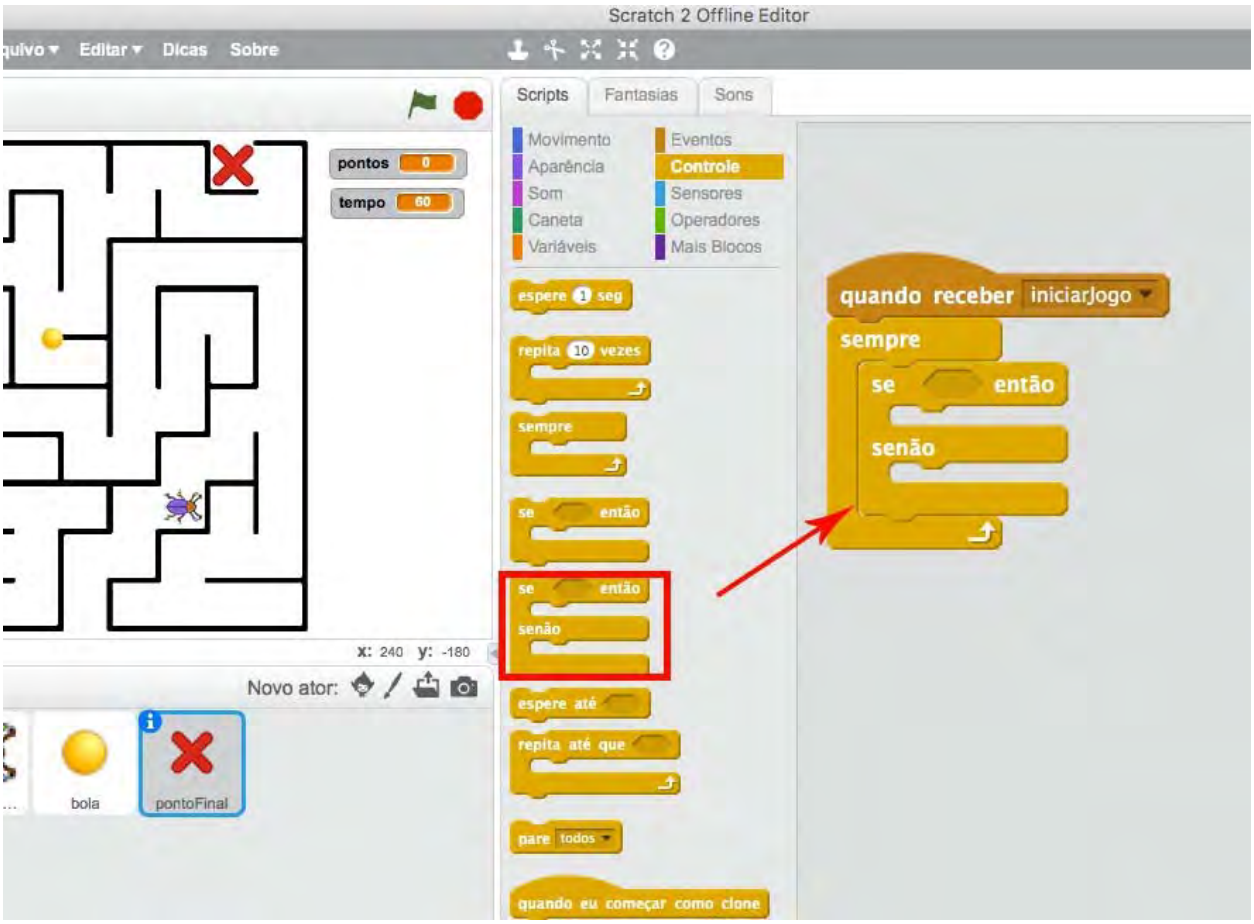
6. Vá em Eventos e arraste o bloco quando receber iniciarJogo. Nesta parte, estamos fazendo com que o verificador comece logo que iniciar o jogo.



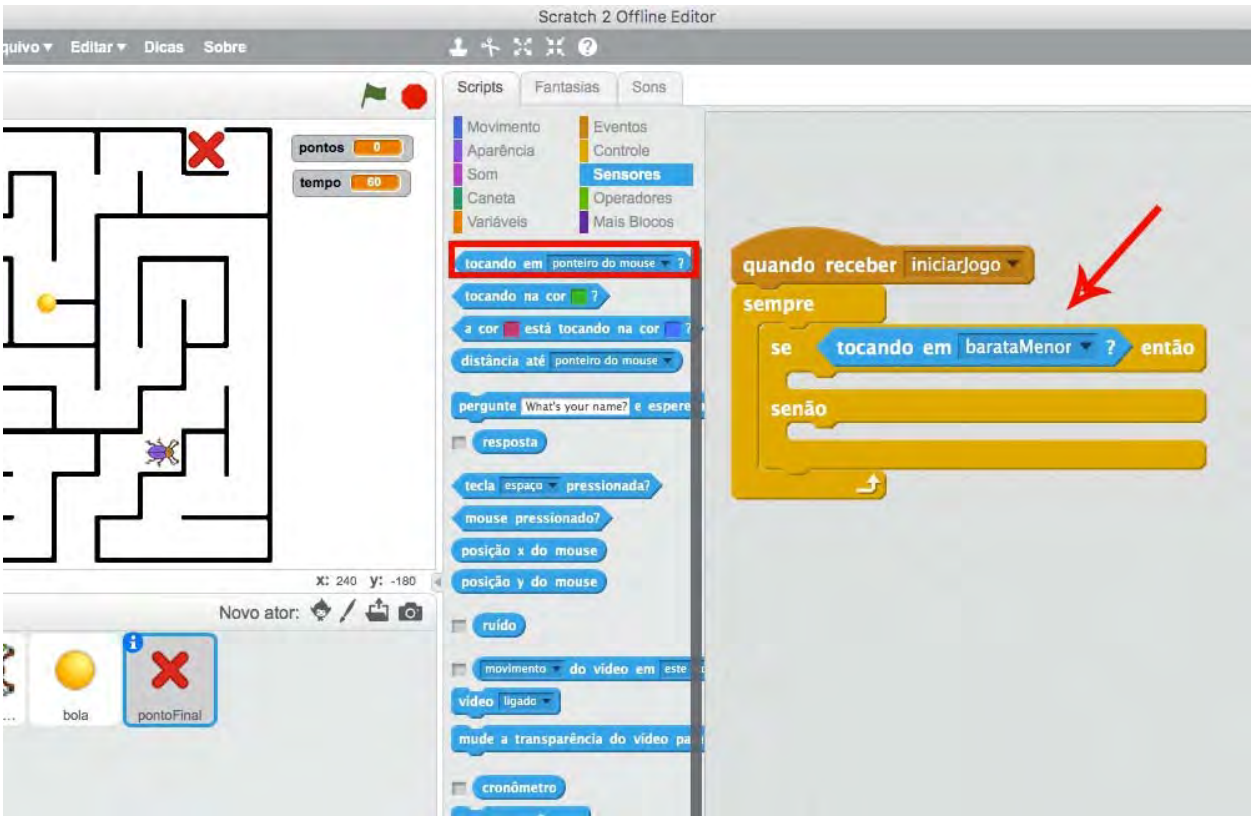
7. Vá em Controle e arraste o bloco Sempre. É necessário o bloco Sempre, pois o verificador funcionará em *loop* infinito.



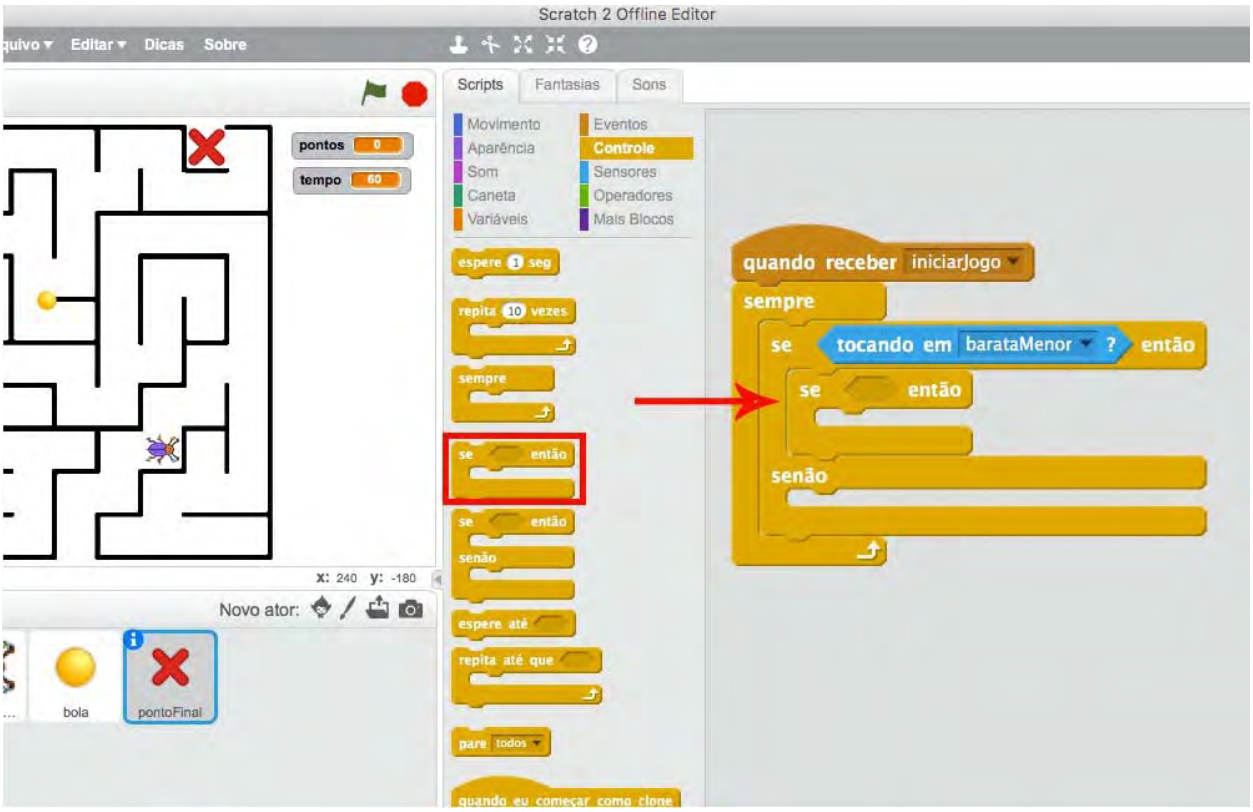
8. Ainda em Controle, arraste o bloco se senão.



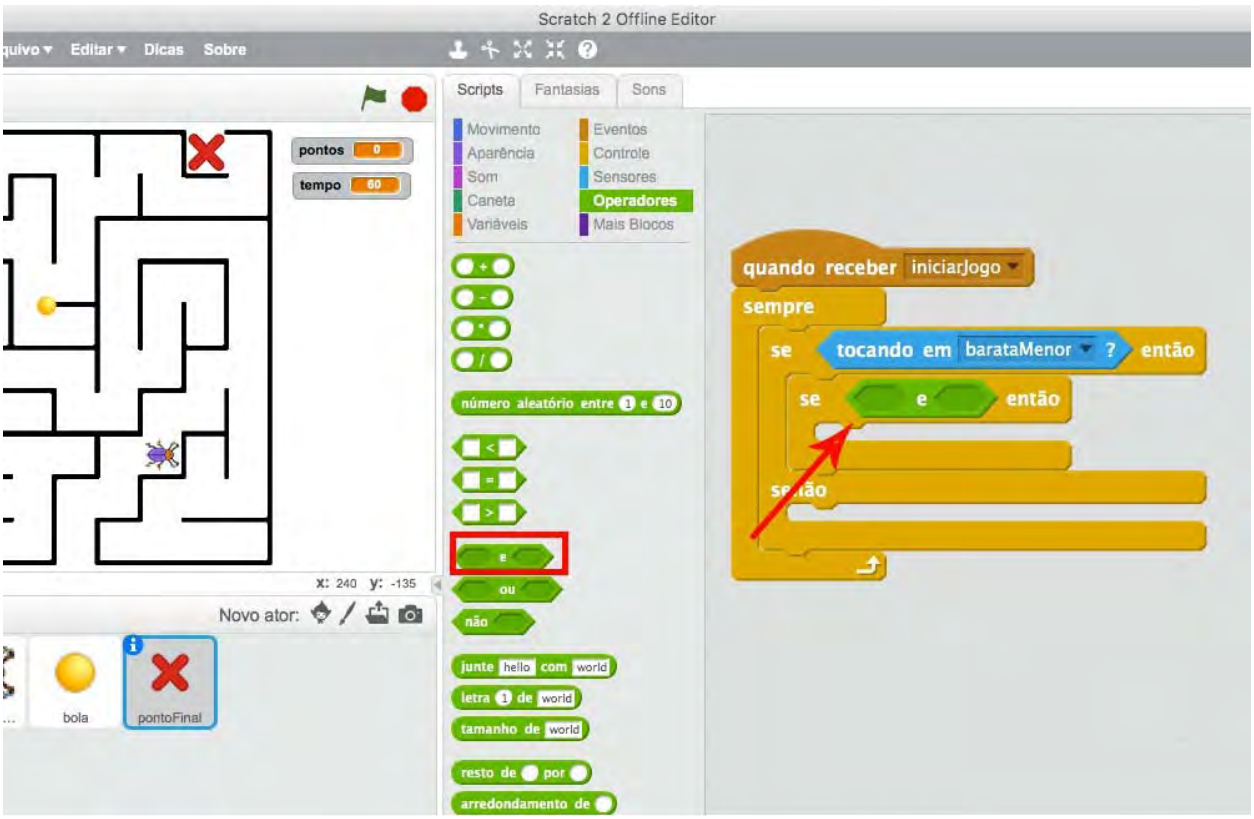
9. Para verificar se o jogador chegou ao ponto de chegada, verificaremos se está encostando no ator. Para isso, vá em Sensores e arraste o bloco tocando em barataMenor.



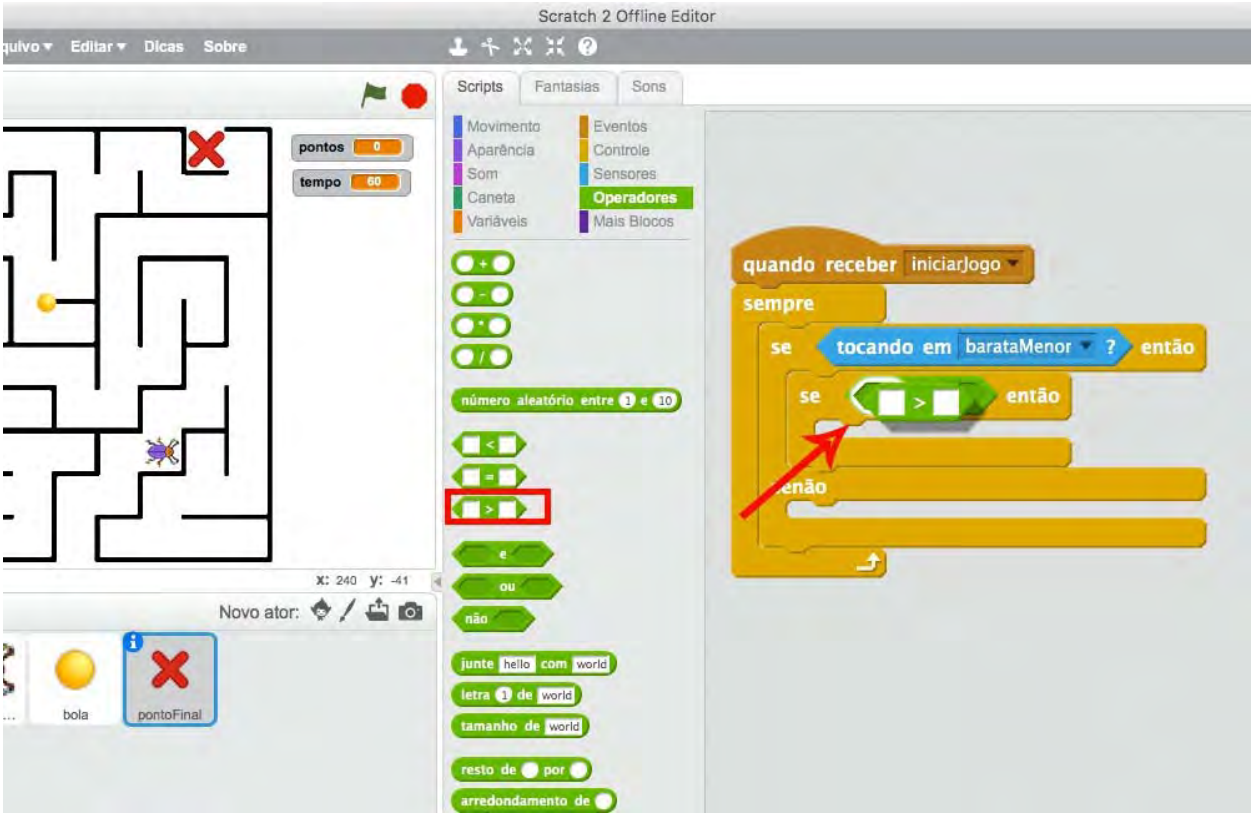
10. Você se lembra da primeira possibilidade? Se não, retorne algumas páginas e dê uma olhada nas três possibilidades. Estamos agora programando a primeira, ou seja, caso o jogador tenha pontos > 0 e tempo > 0. Vá em Controle e arraste o bloco se então.



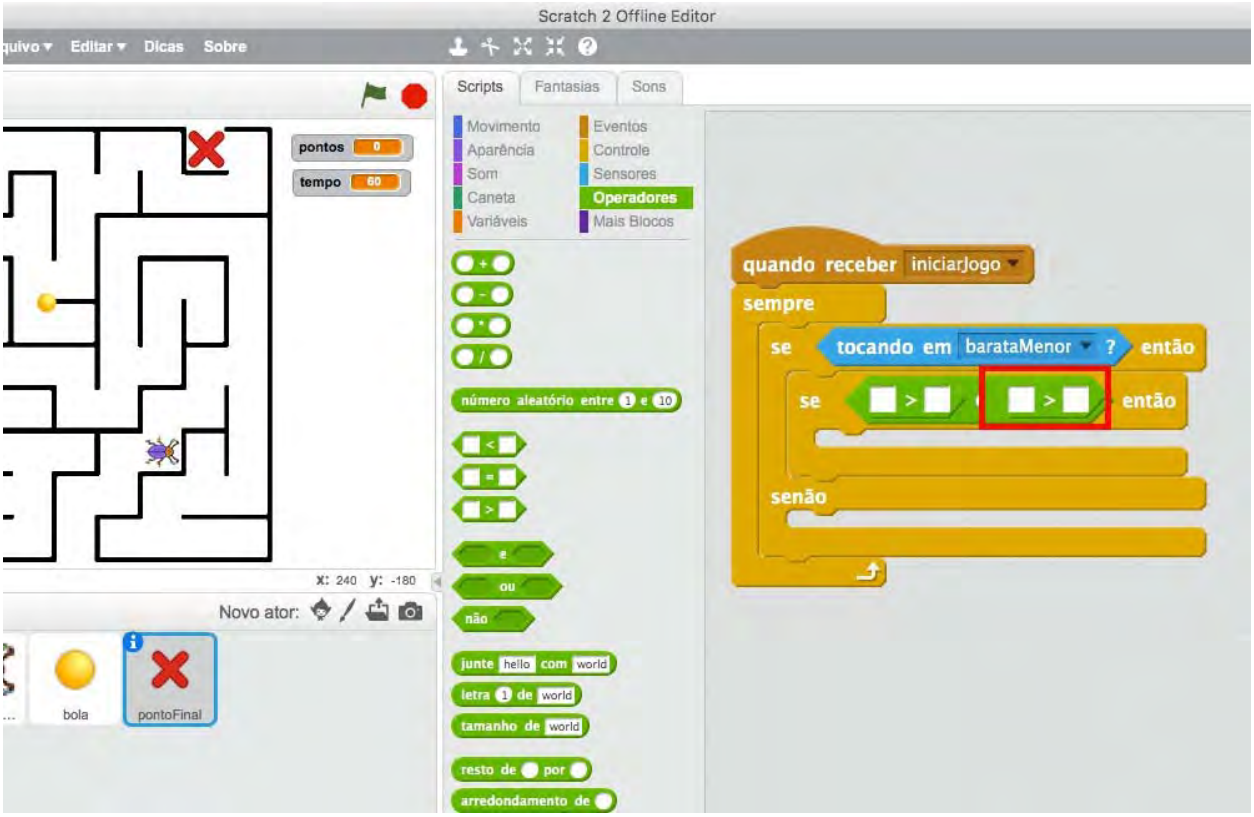
11. Vá em Operadores e arraste o operador lógico e para dentro da área hexagonal.



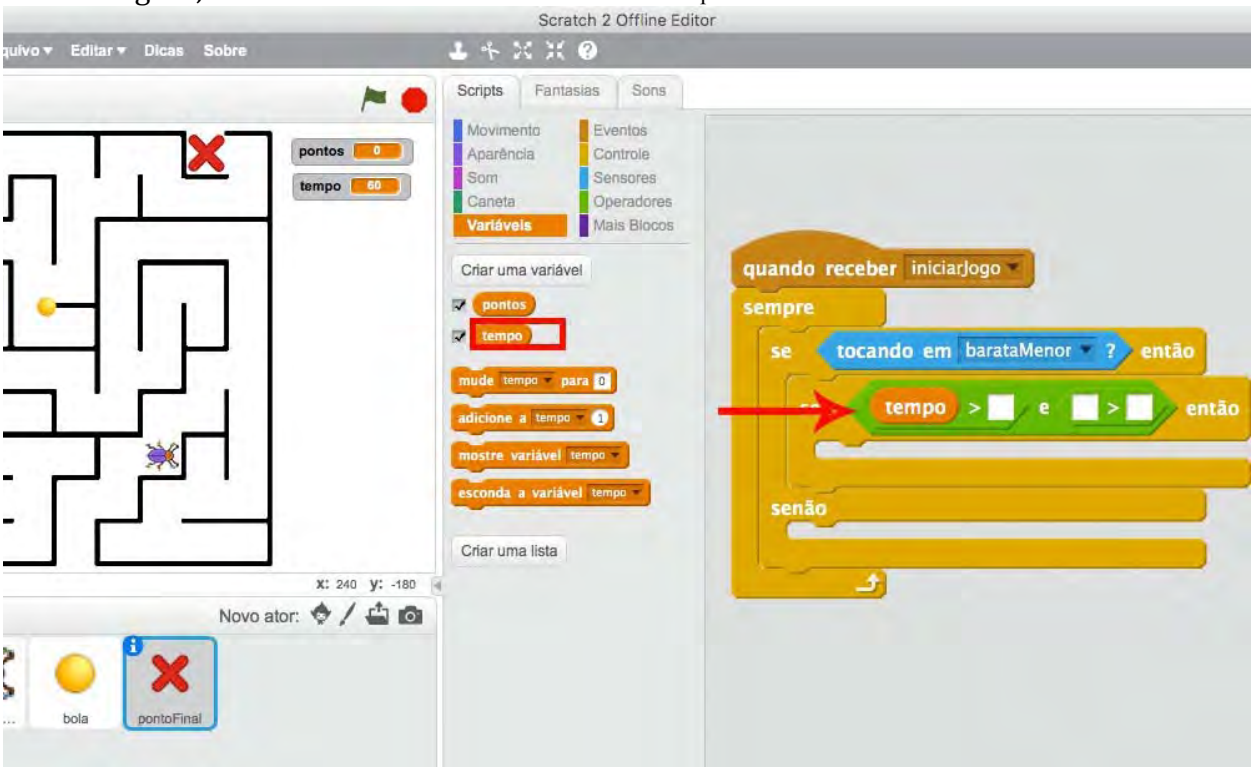
12. Ainda em Operadores, arraste o operador de comparação > para dentro do bloco lógico e.



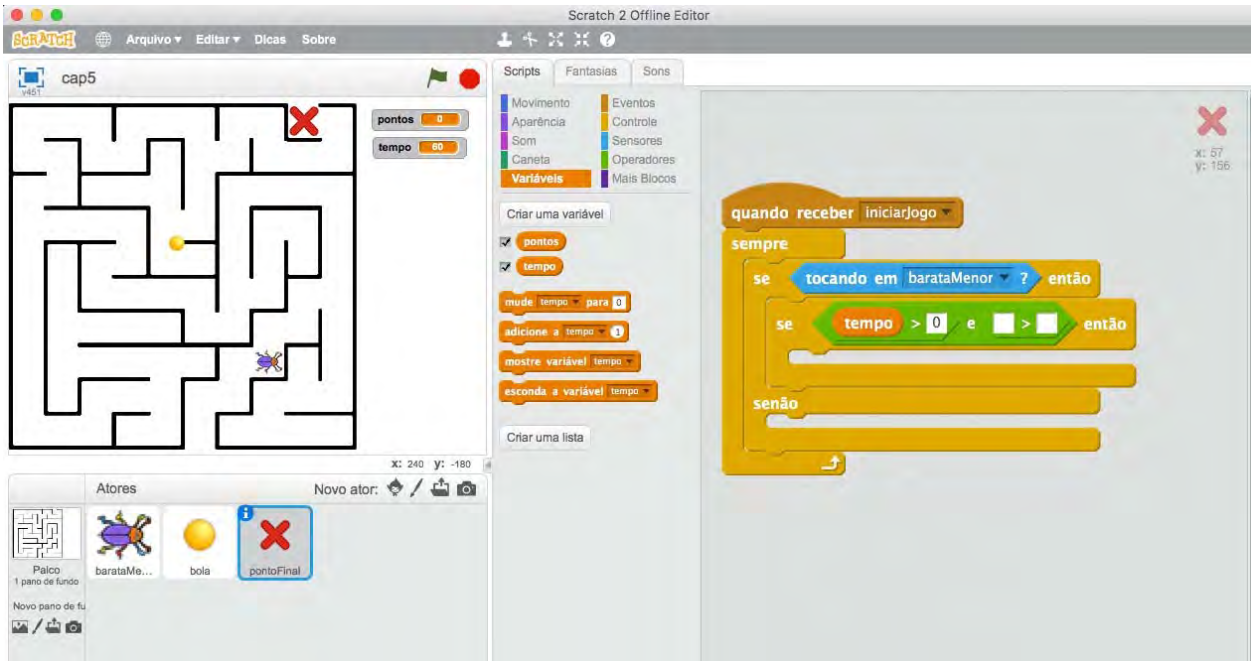
13. Repita o passo anterior para a caixa que sobrou do operador e.



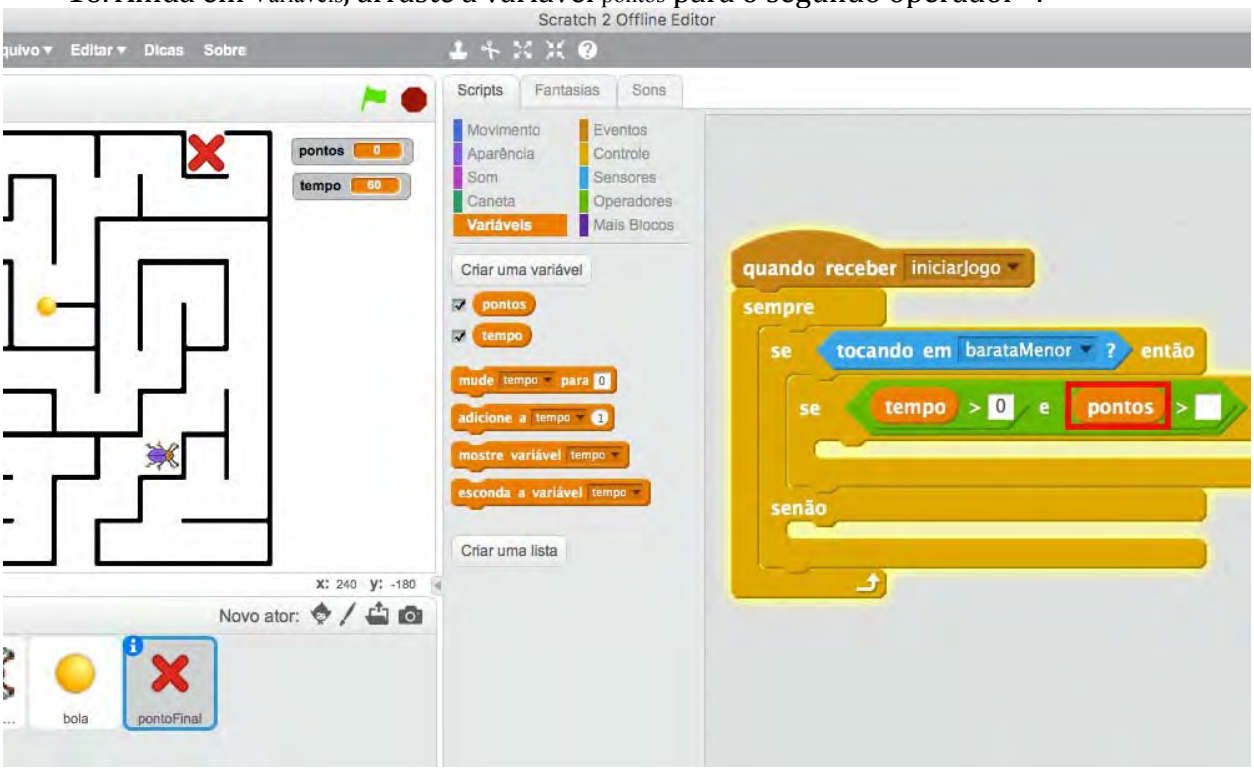
14. Agora, vá em Variáveis e arraste a variável tempo.



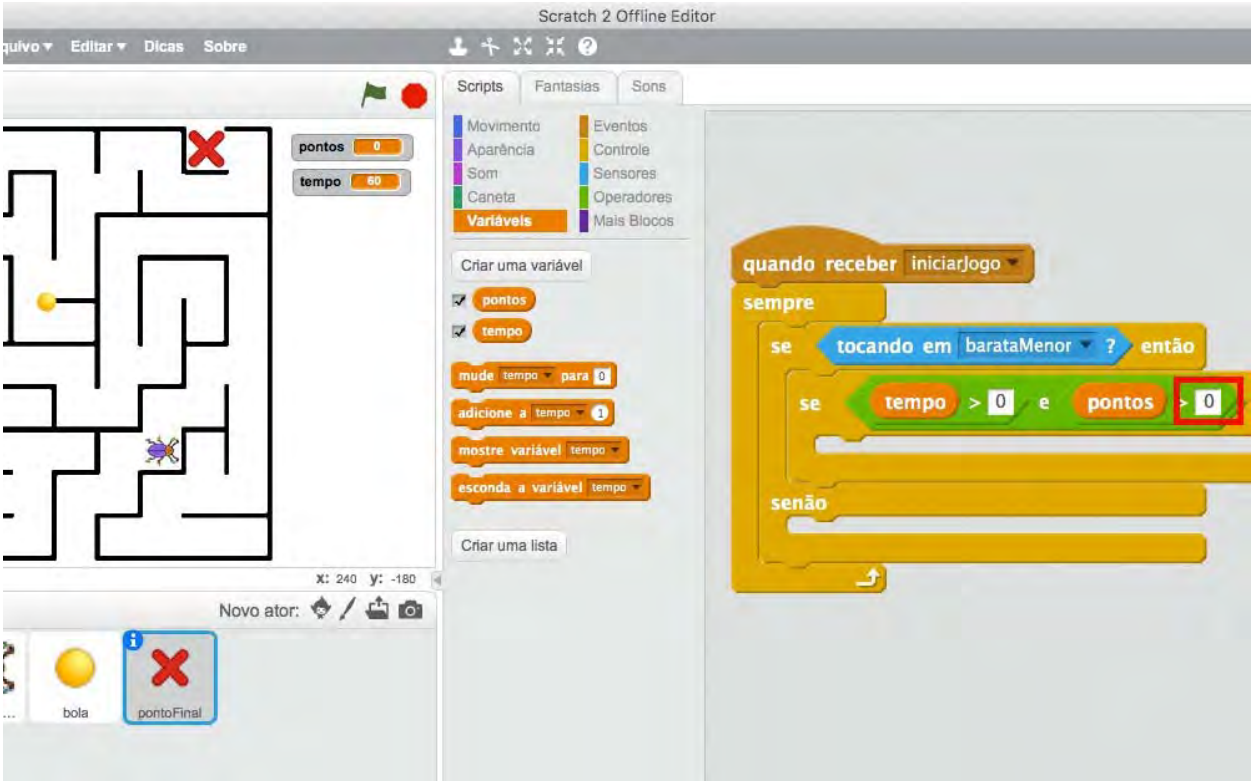
15. Preencha com o valor 0 a segunda caixa do operador >.



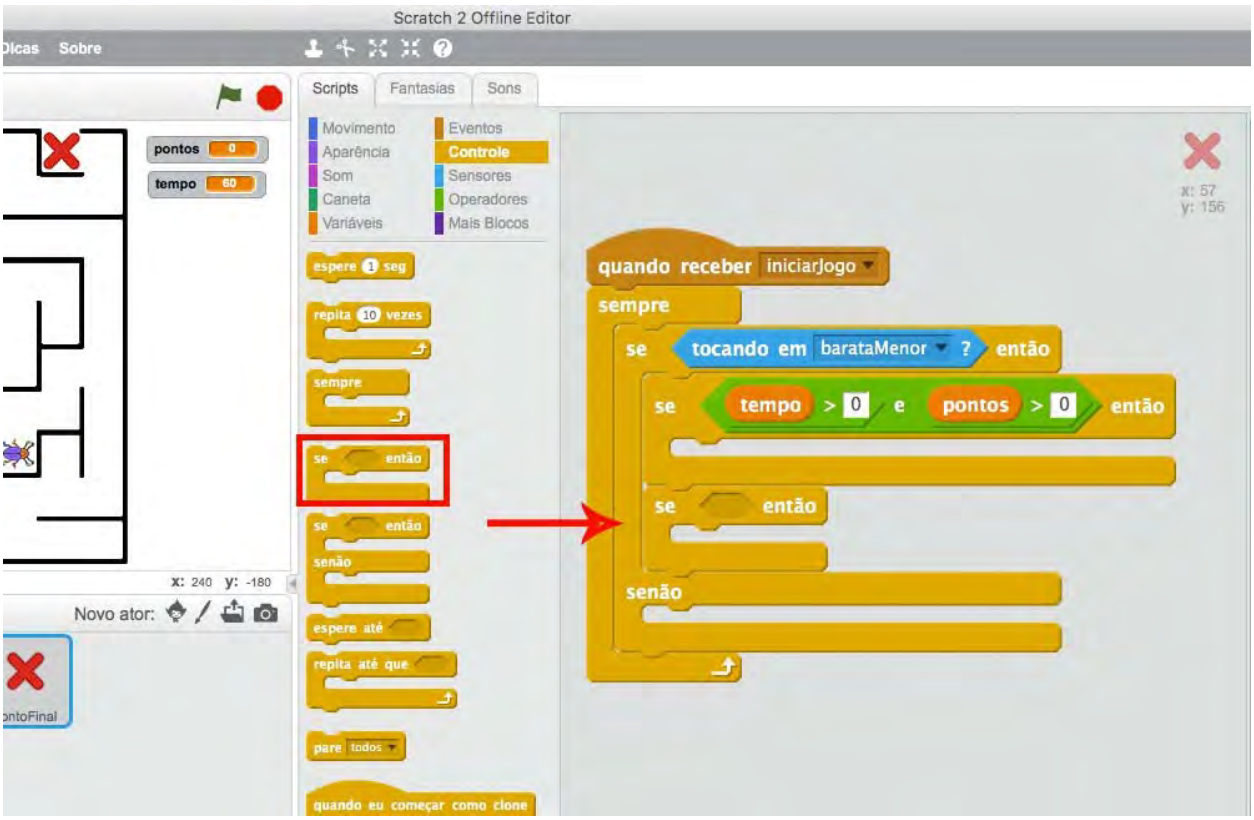
16. Ainda em Variáveis, arraste a variável pontos para o segundo operador >.



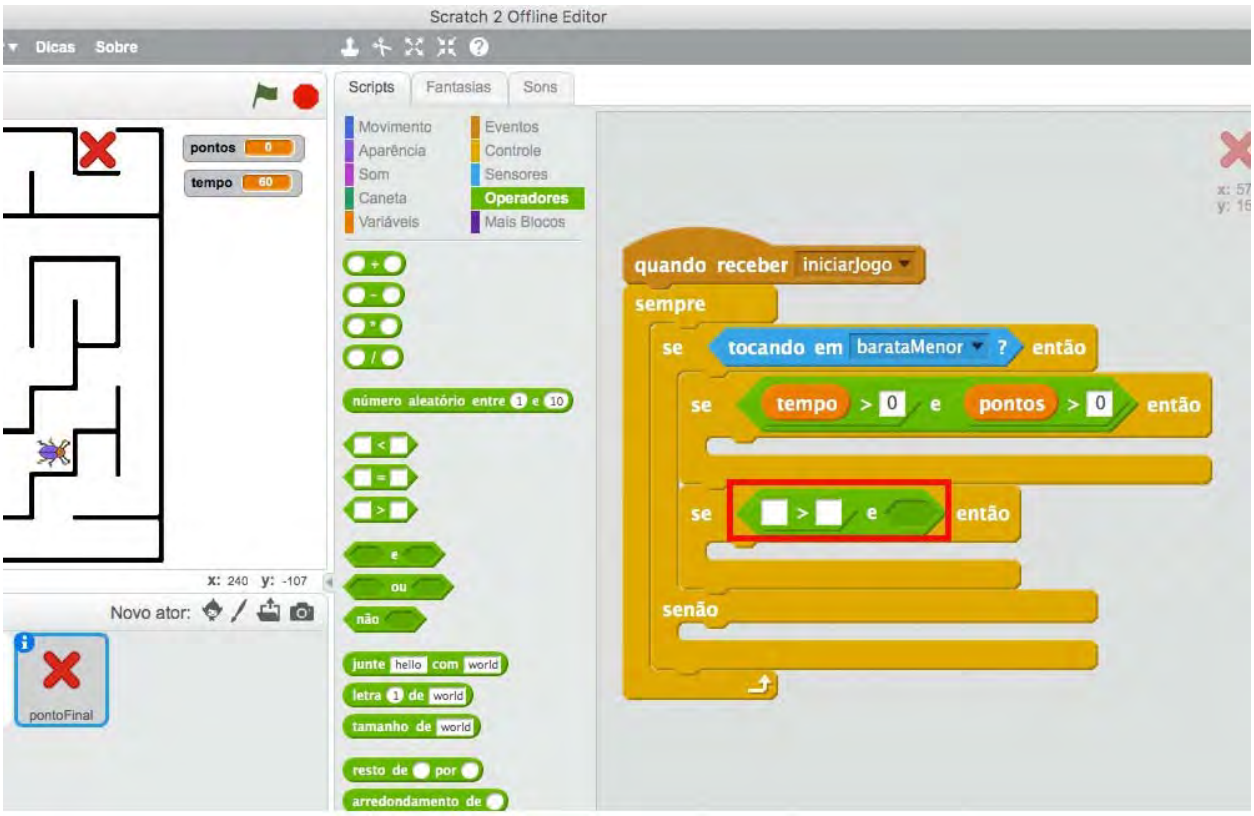
17. Preencha também com o valor 0. Finalizamos a primeira possibilidade. Caso o jogador tenha tempo > 0 e pontos > 0, será executada "alguma coisa". Você consegue imaginar o que é? Caso você tenha pensando "procedimento", parabéns! Ninguém o segura mais.



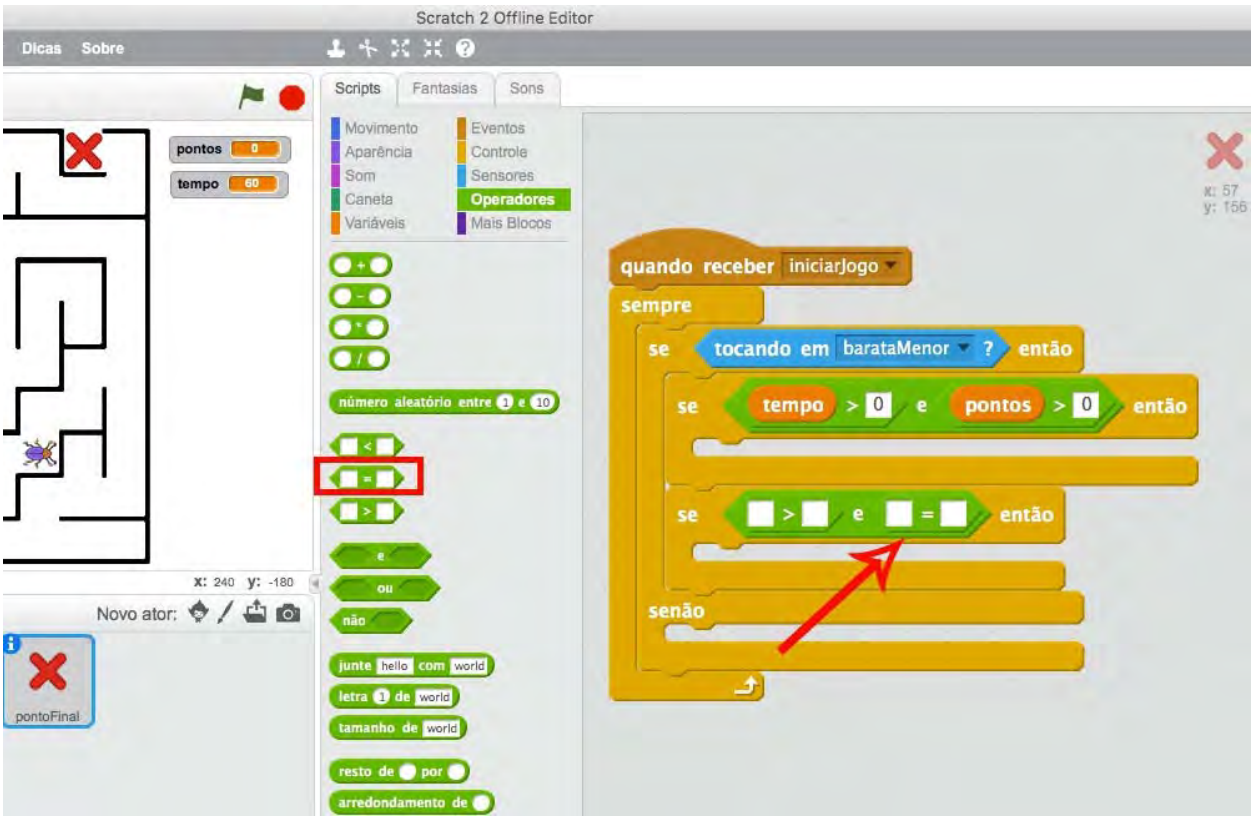
18. Vá em Controle e arraste outro bloco se então para baixo do bloco adicionado anteriormente. Estamos programando a segunda possibilidade. Preste bastante atenção e tente entender o que está acontecendo.



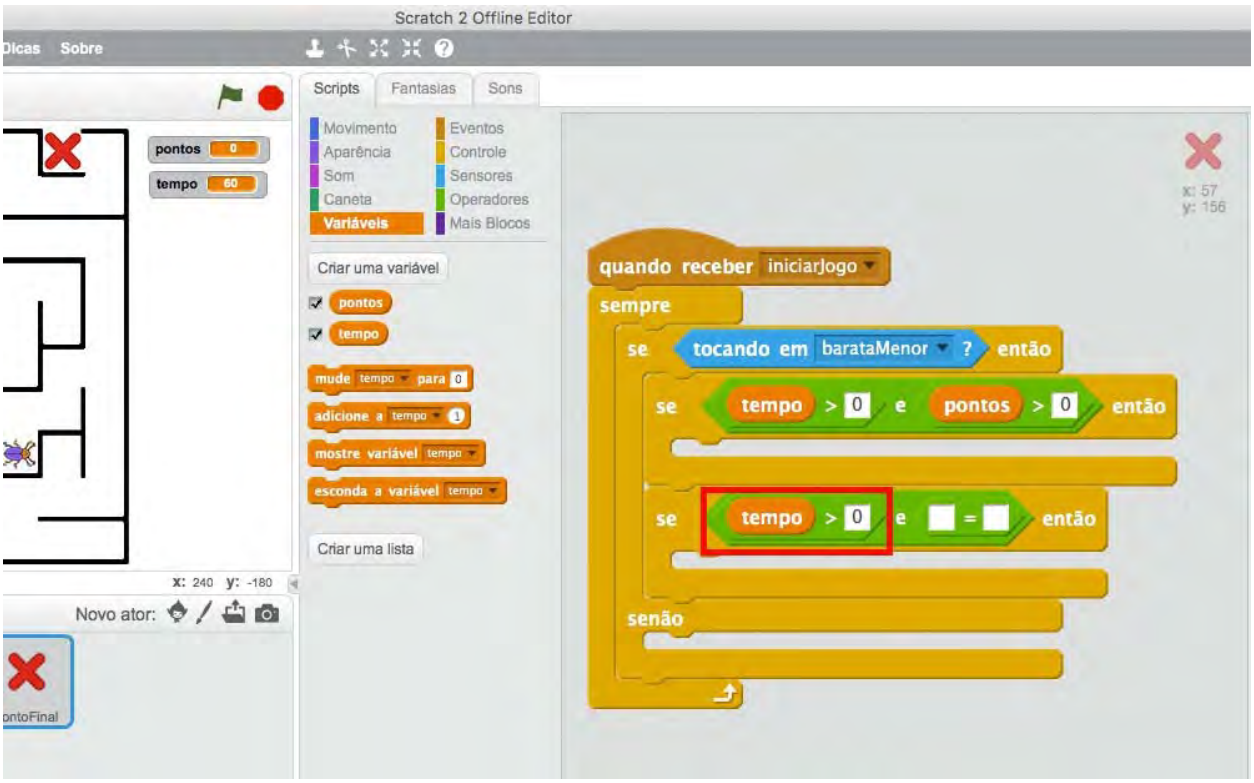
19. Vamos repetir o processo anterior. Vá em Operadores e adicione um operador e e um operador >.



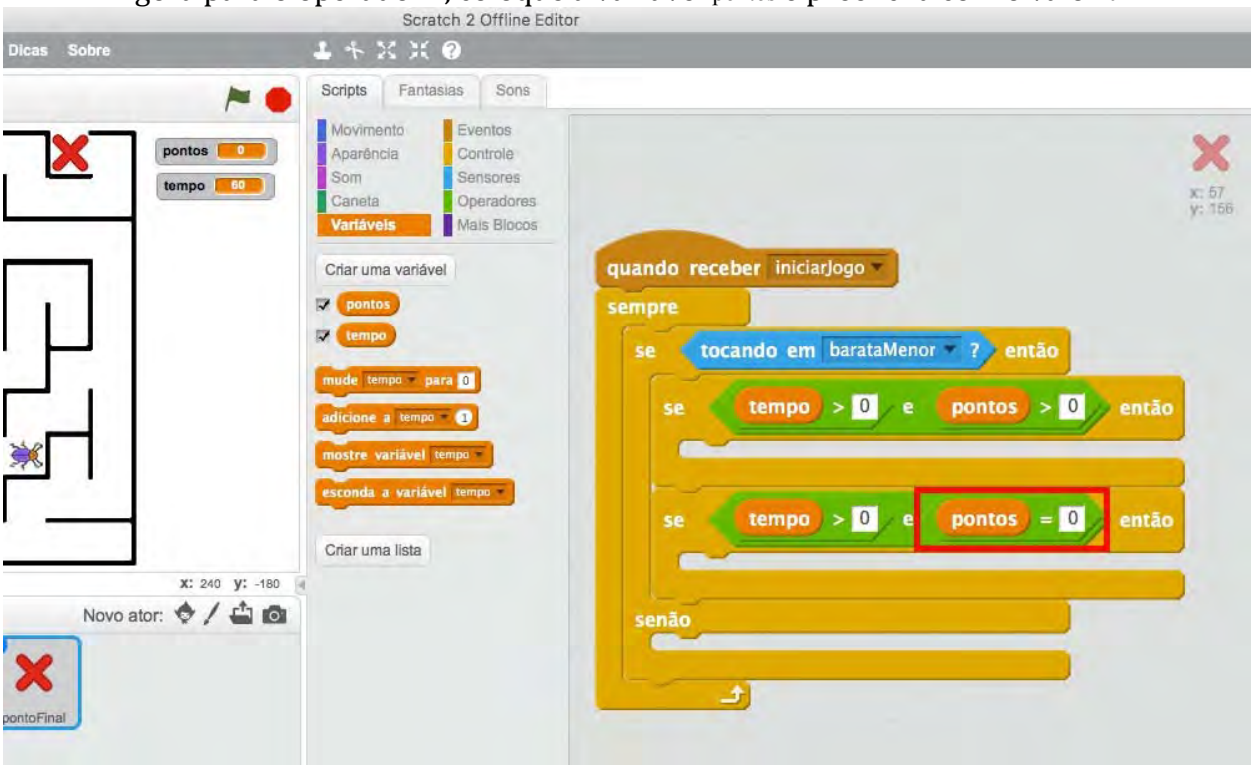
20. Ainda em Operadores, coloque um operador =, na segunda caixa do operador e.



21. Agora vá em Variáveis e, no operador >, coloque a variável tempo e preencha a outra caixa com o valor 0.

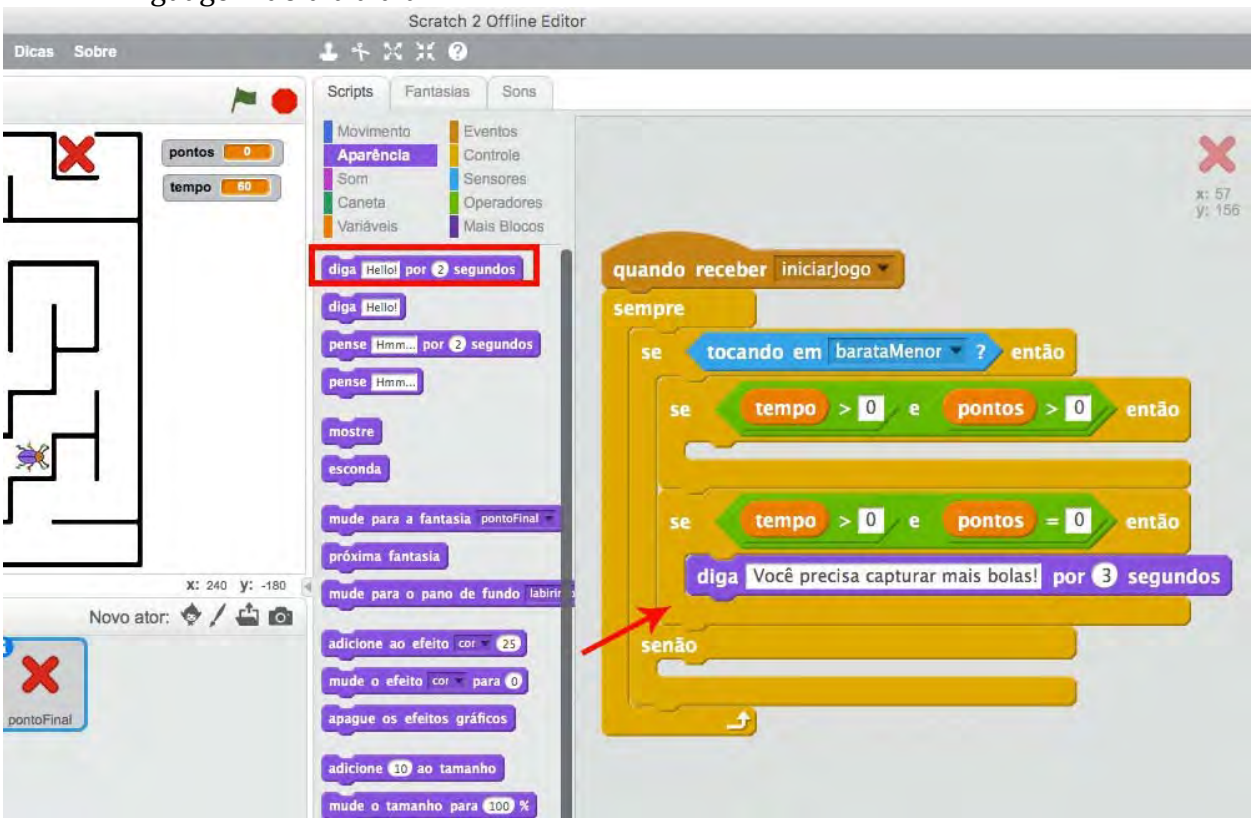


22. Agora para o operador =, coloque a variável pontos e preencha com o valor 0.



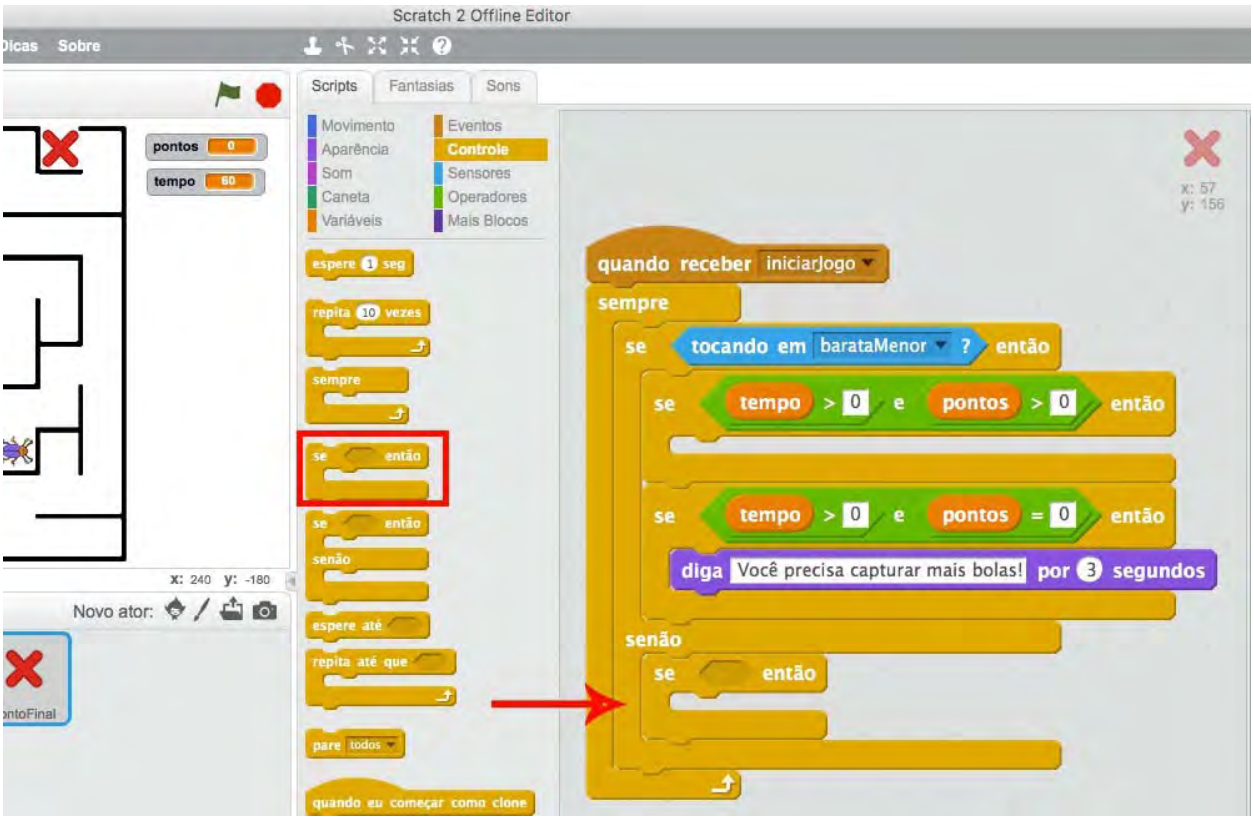
23. Vá em Aparência e arraste o bloco diga por x segundos. Preencha os campos conforme a figura. Finalizamos a segunda possibilidade! E dessa vez não executamos nenhum

procedimento, pois o jogador ainda precisa capturar algo, somente exibimos uma mensagem. Você viu como as expressões "e, ou" são parecidas com a nossa linguagem do dia a dia?

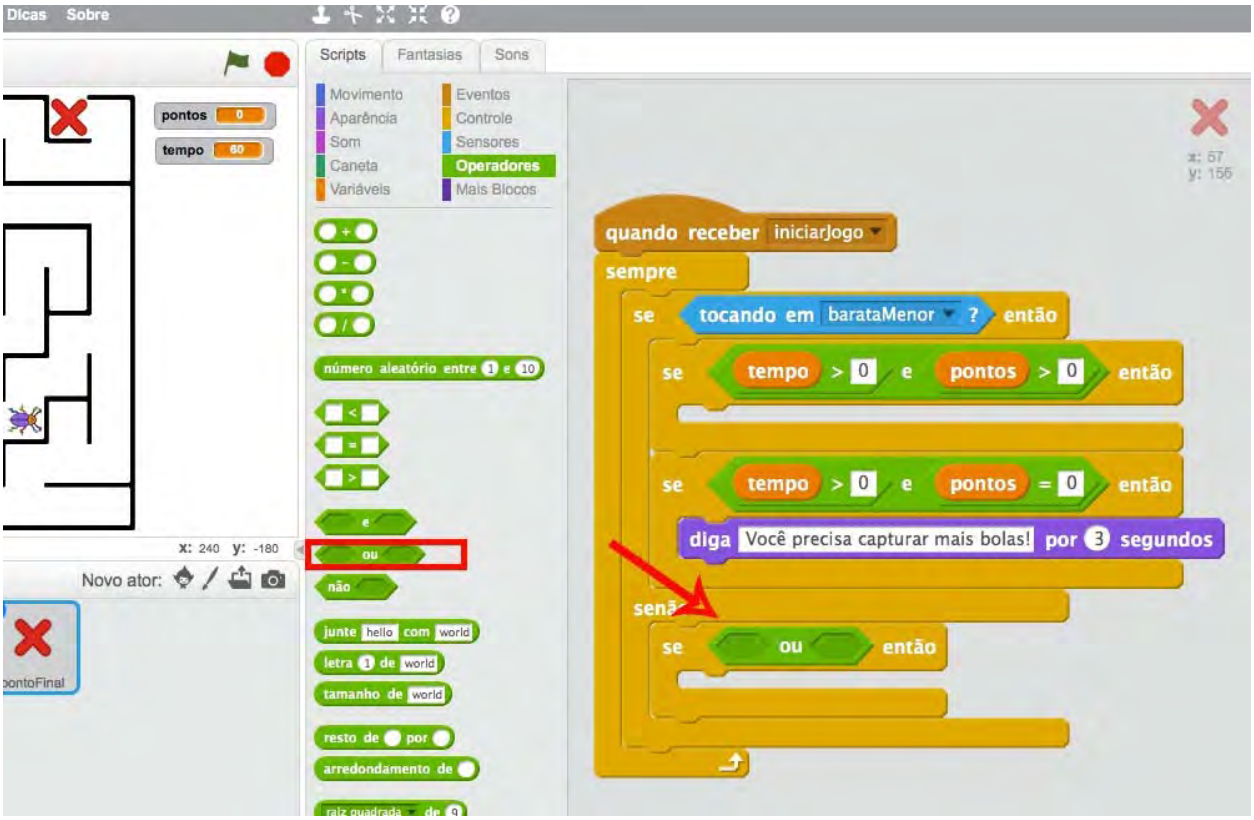


A próxima parte é um pouco mais complicadinha e pode confundir bastante. Veja com bastante atenção e tente entender como funciona essa terceira possibilidade. Vamos programar uma expressão enorme cheia de "e, ou".

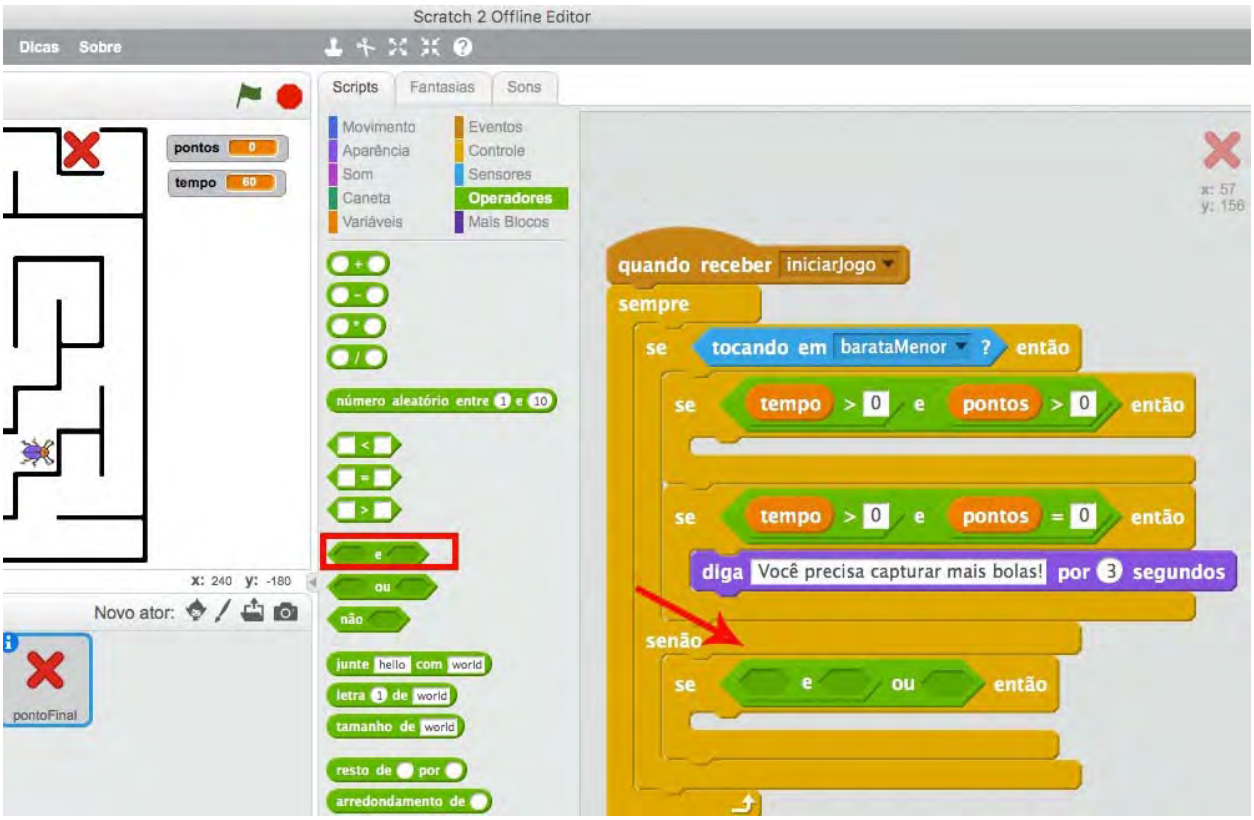
24. Vamos adicionar o último bloco se então. Dessa vez, coloque-o na parte do Senão. Veja a figura.



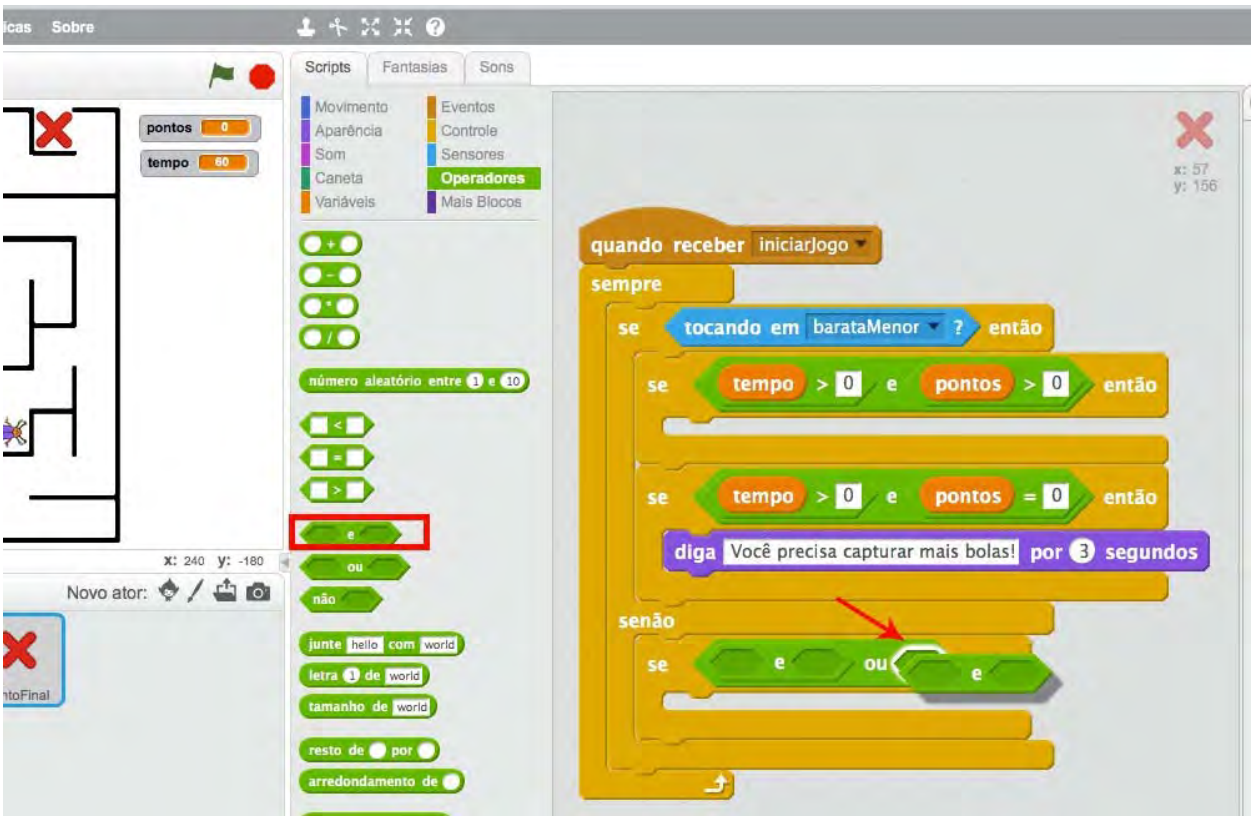
25. Vá em Operadores e arraste o operador ou.



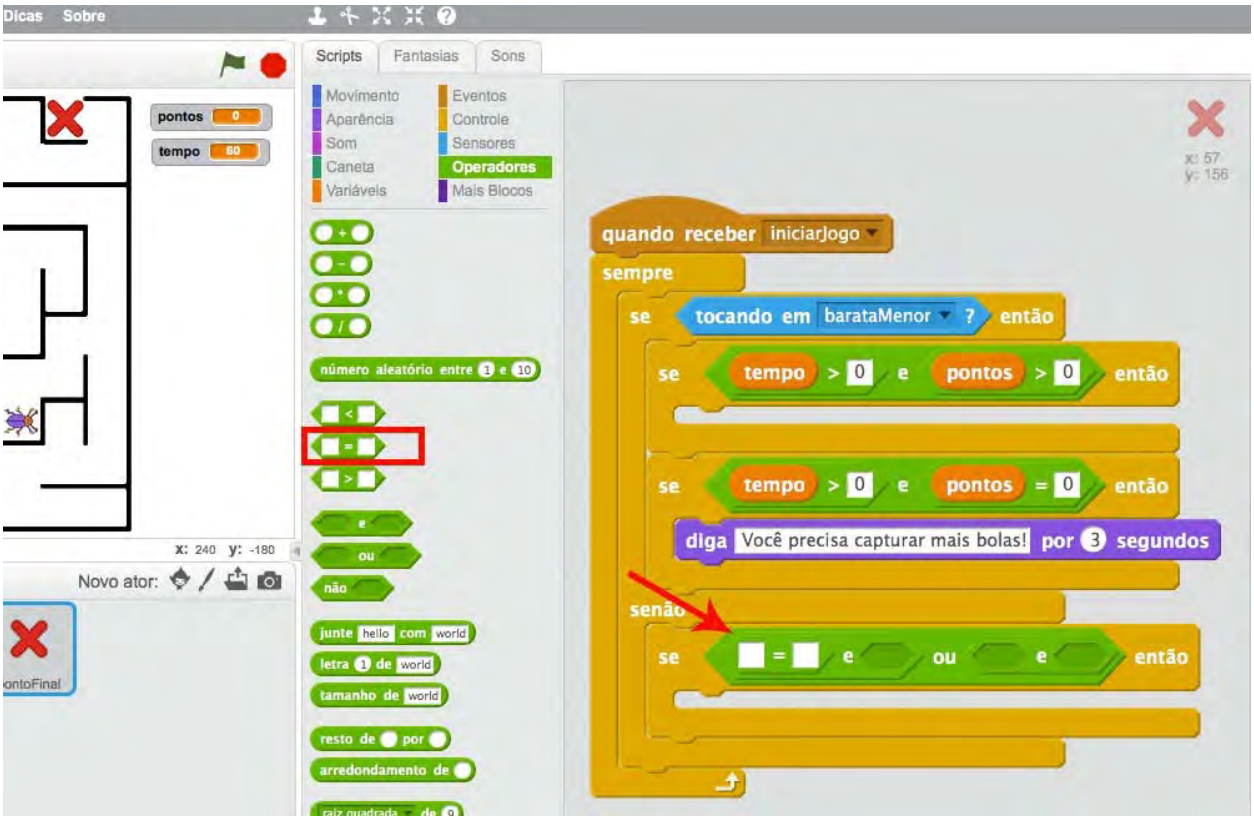
26. Ainda em Operadores, arraste o operador e dentro da área hexagonal do bloco ou.



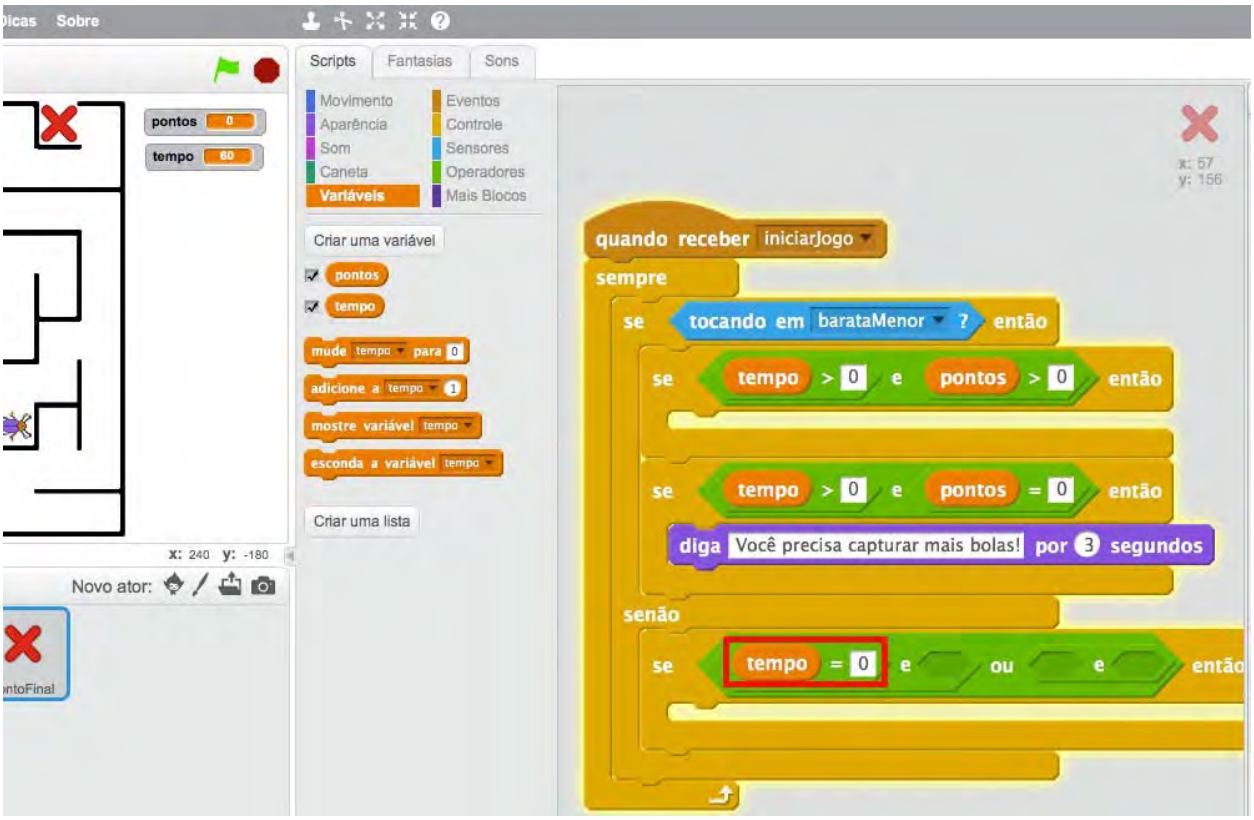
27. Faça o mesmo para a outra área vazia. Retorne algumas páginas e dê uma olhada na terceira possibilidade para tentar acompanhar o código a partir daqui.



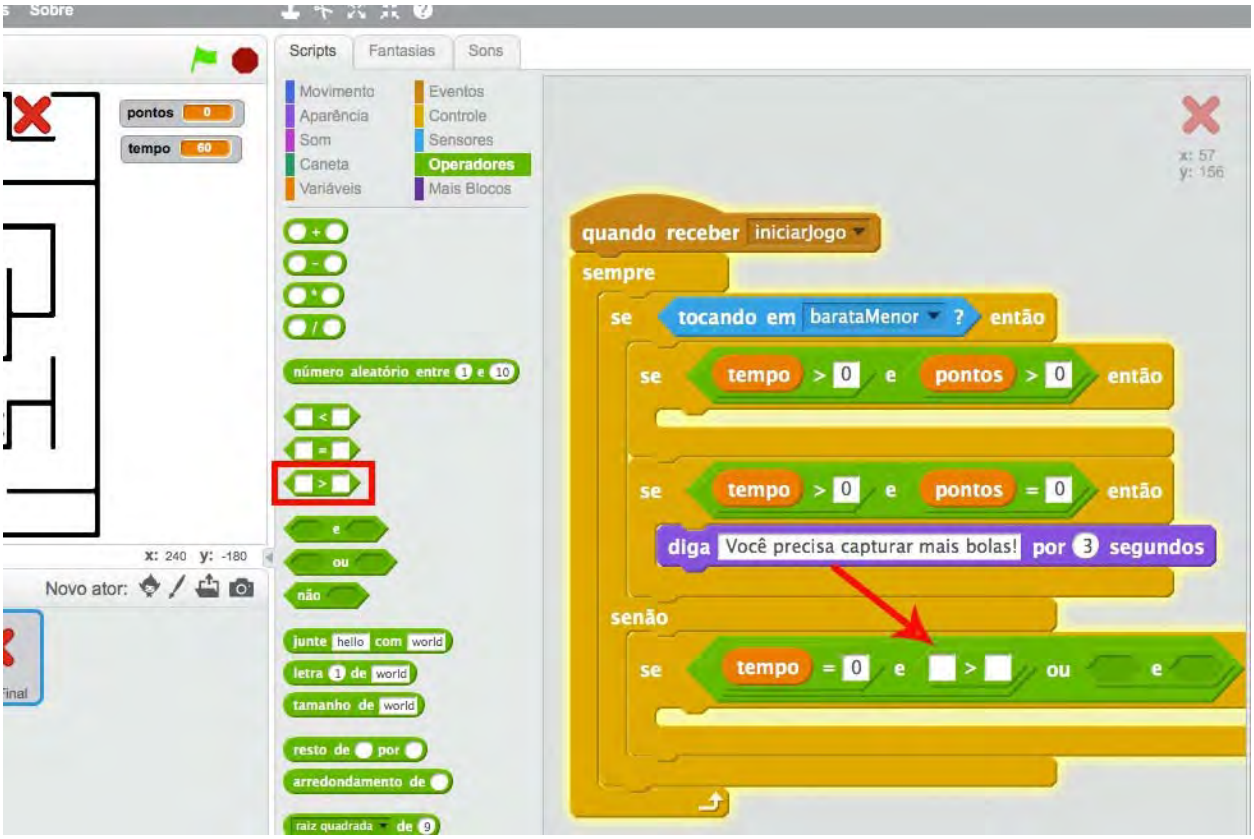
28. Agora arraste o operador = dentro do bloco e.



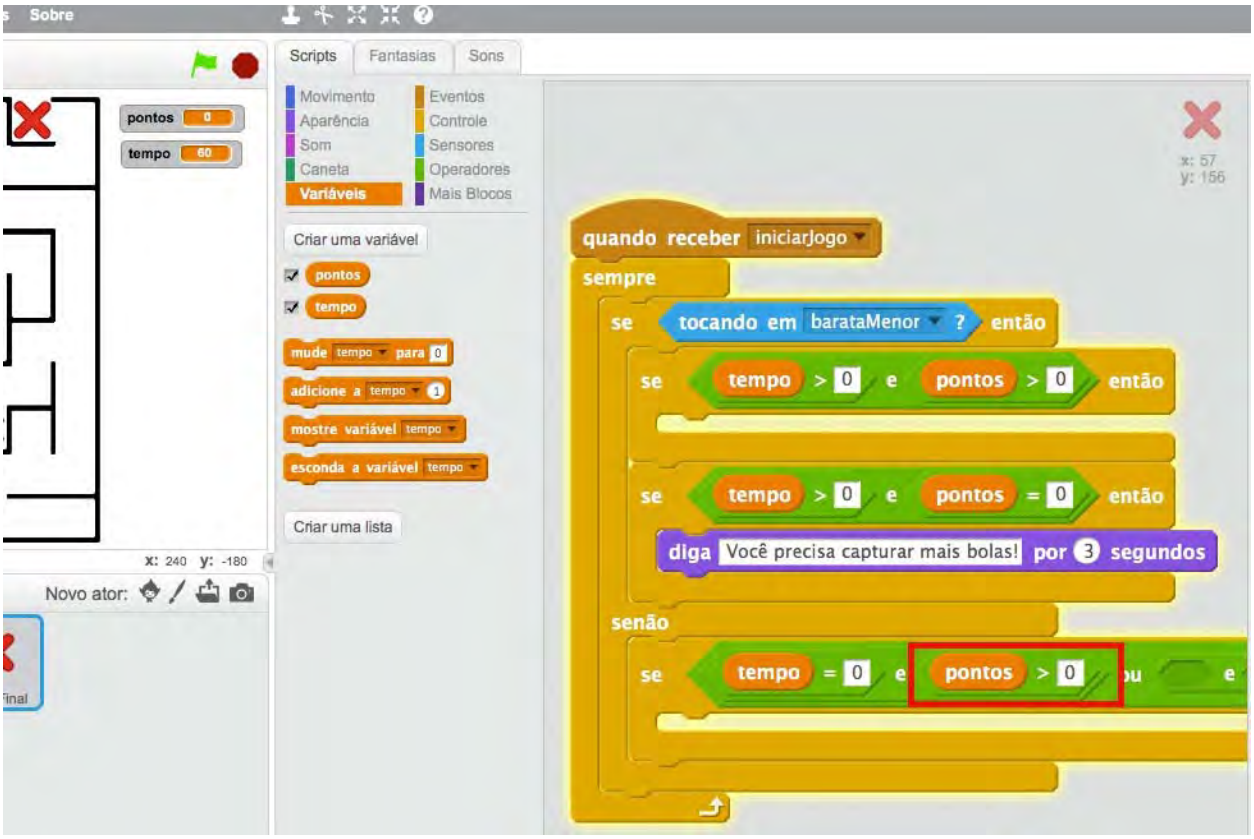
29. Preencha a primeira área do operador = com a variável tempo, e a segunda área com o valor 0.



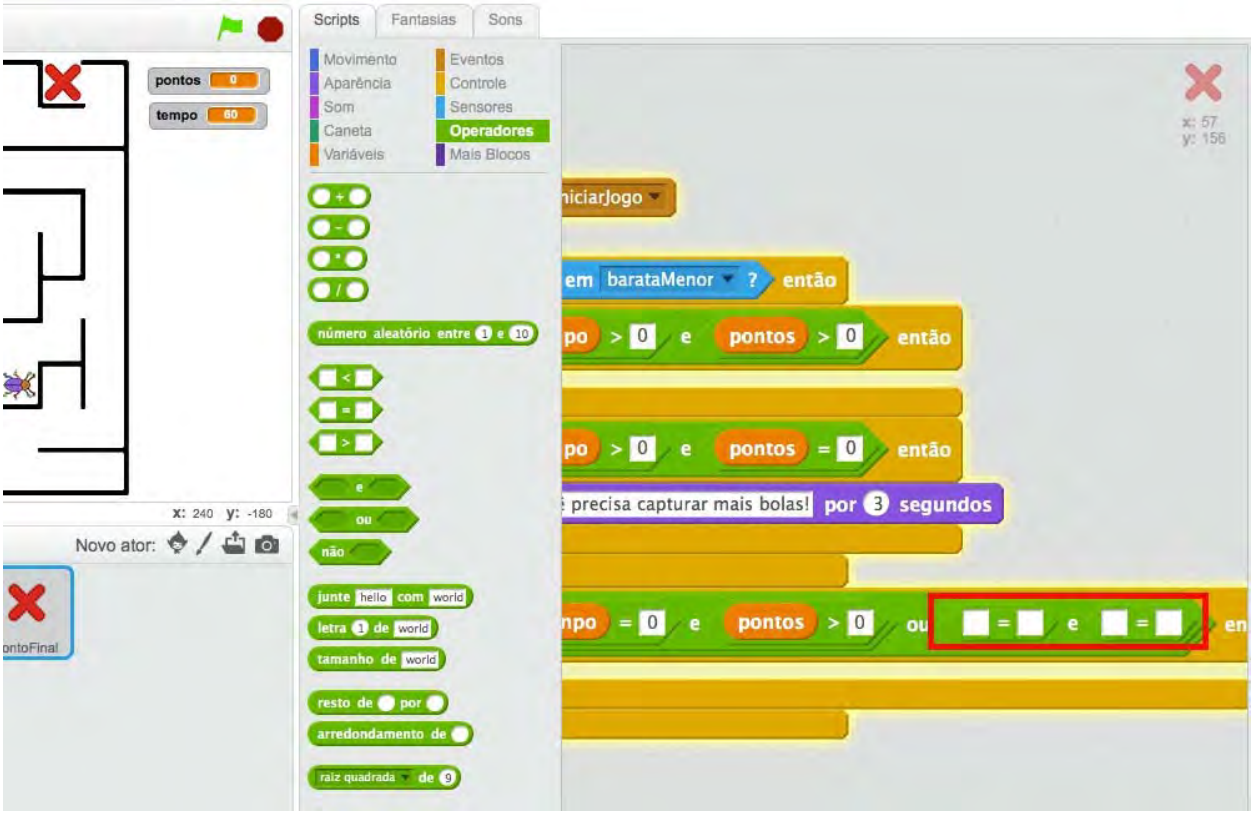
30. Na segunda área do bloco e, arraste o operador >.



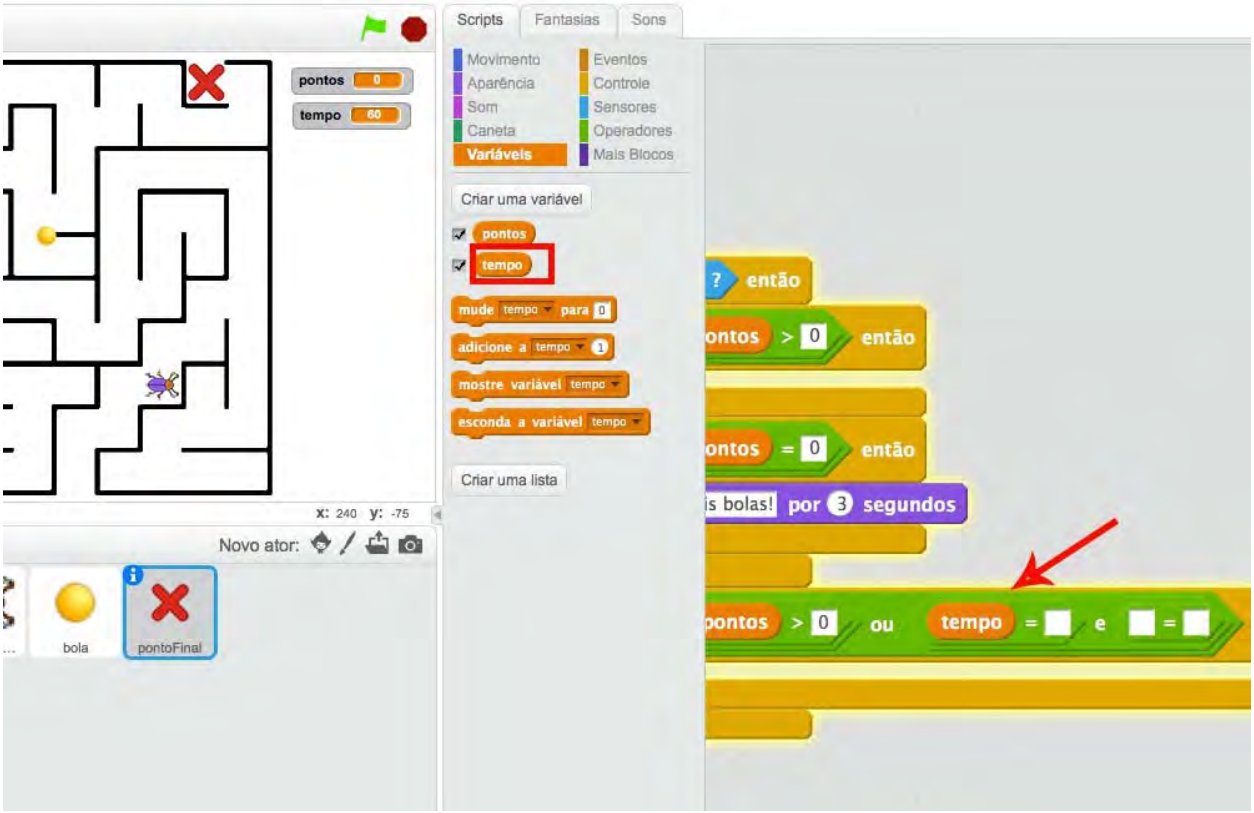
31. Preencha a primeira área com a variável pontos, e a segunda área com o valor 0.



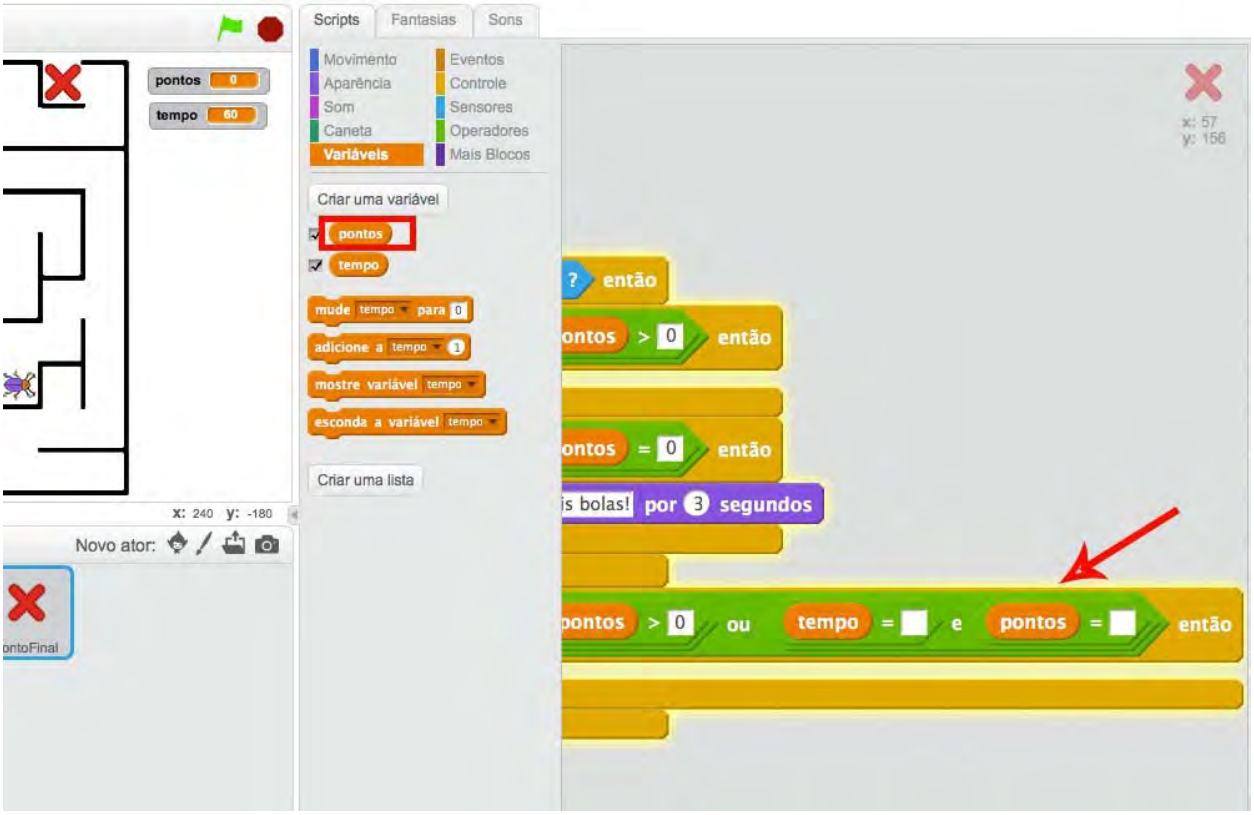
32. Vamos agora para a outra condição do operador ou. Preencha com o operador = a primeira e segunda área do operador e.



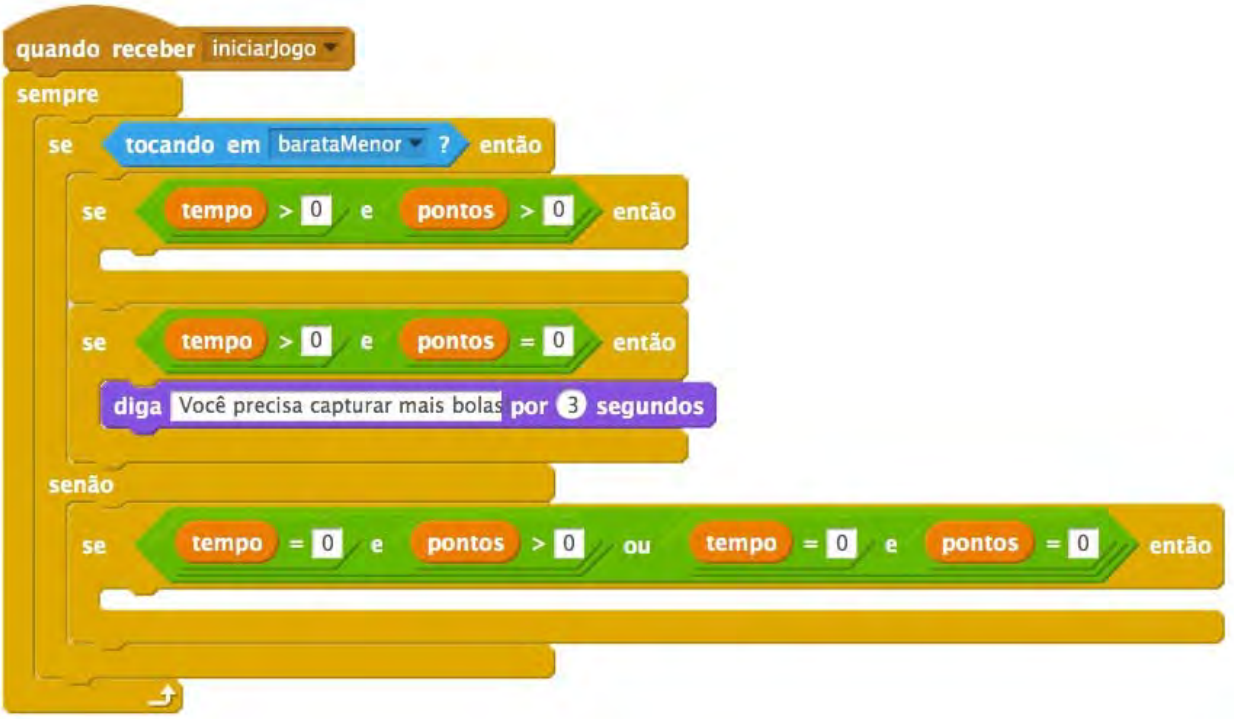
33. Agora no operador =, preencha a primeira área com a variável tempo.



34. No segundo operador = preencha com a variável pontos.



35. Nas caixas restantes, coloque o valor 0. Seu código completo ficará assim:

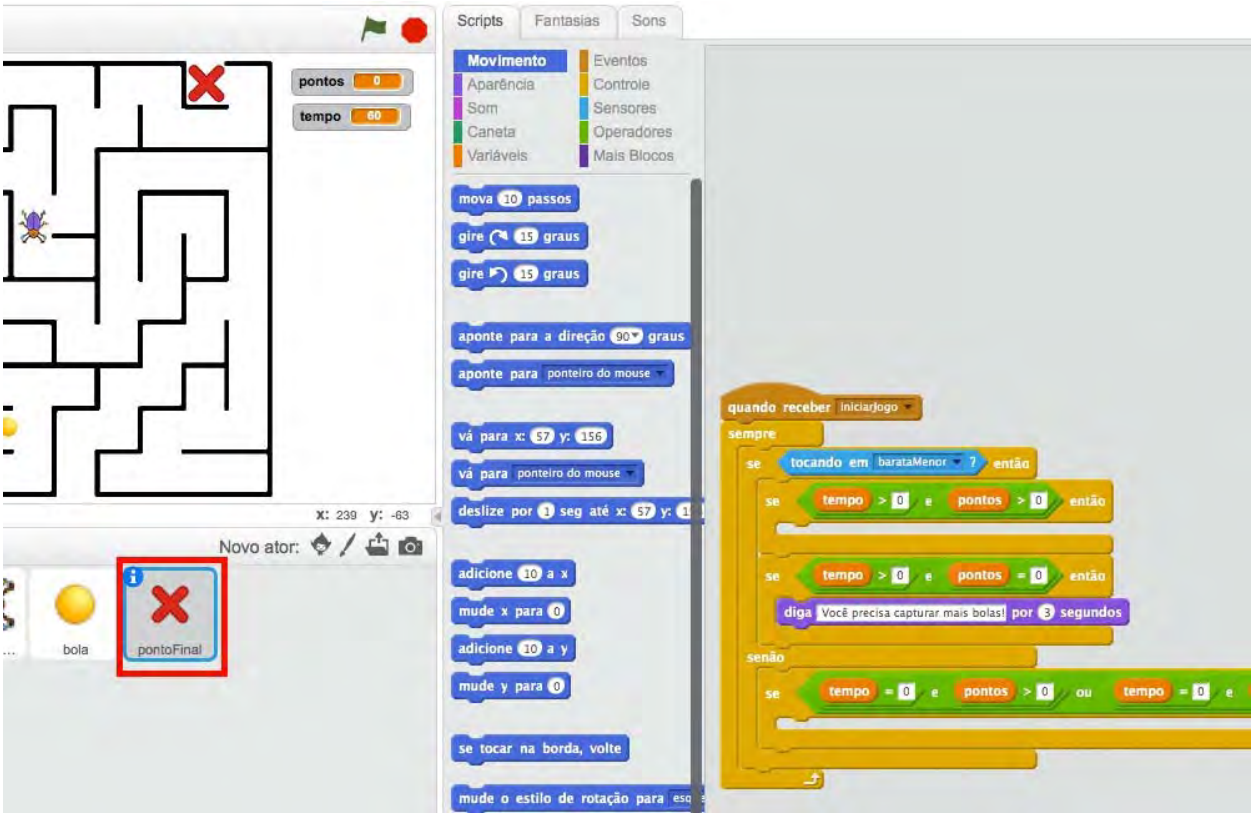


O que esse código está fazendo? Basicamente ele é responsável por fazer a verificação das possibilidades que nós vimos no começo do capítulo.

1. Quando o ator receber a mensagem iniciarJogo;
2. Entra em loop infinito;
3. E verifica se o ator está tocando no pontoFinal;
4. Caso **verdadeiro**, verifica a primeira possibilidade e, depois, a segunda possibilidade;
5. Caso **falso**, verifica a terceira possibilidade.

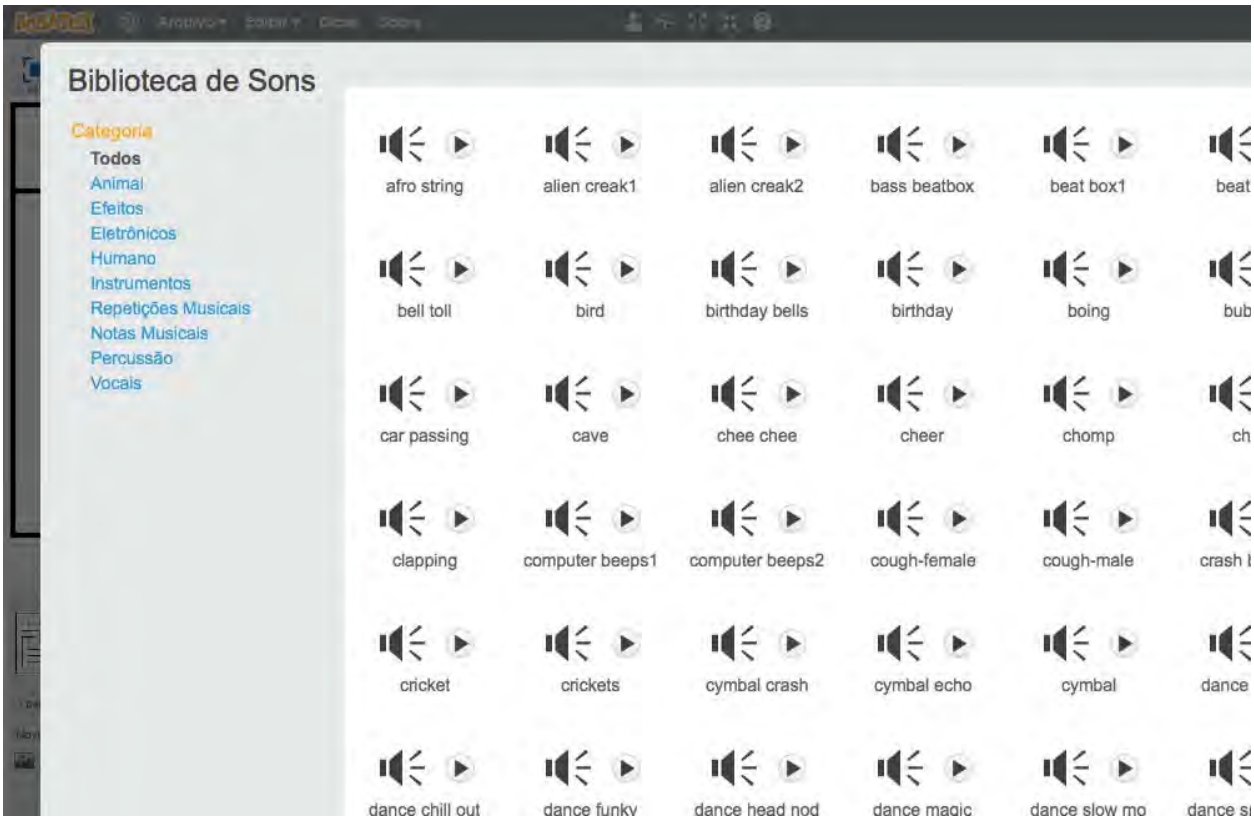
Agora sim nosso joga vai ficar interessante! Vamos criar os procedimentos responsáveis por dar a mensagem avisando se o jogador ganhou ou não, e para encerrar o jogo. Essa parte é fundamental para que o jogo tenha um fim.

1. Com o ator pontoFinal selecionado, deixe um espaço vazio na parte superior da Área de Scripts. Vamos deixar este espaço simplesmente para deixar essa área mais organizada. Se você quiser, pode colocar o código na parte debaixo também, não há nenhum problema. Somente para facilitar a visualização, deixaremos na parte de cima.

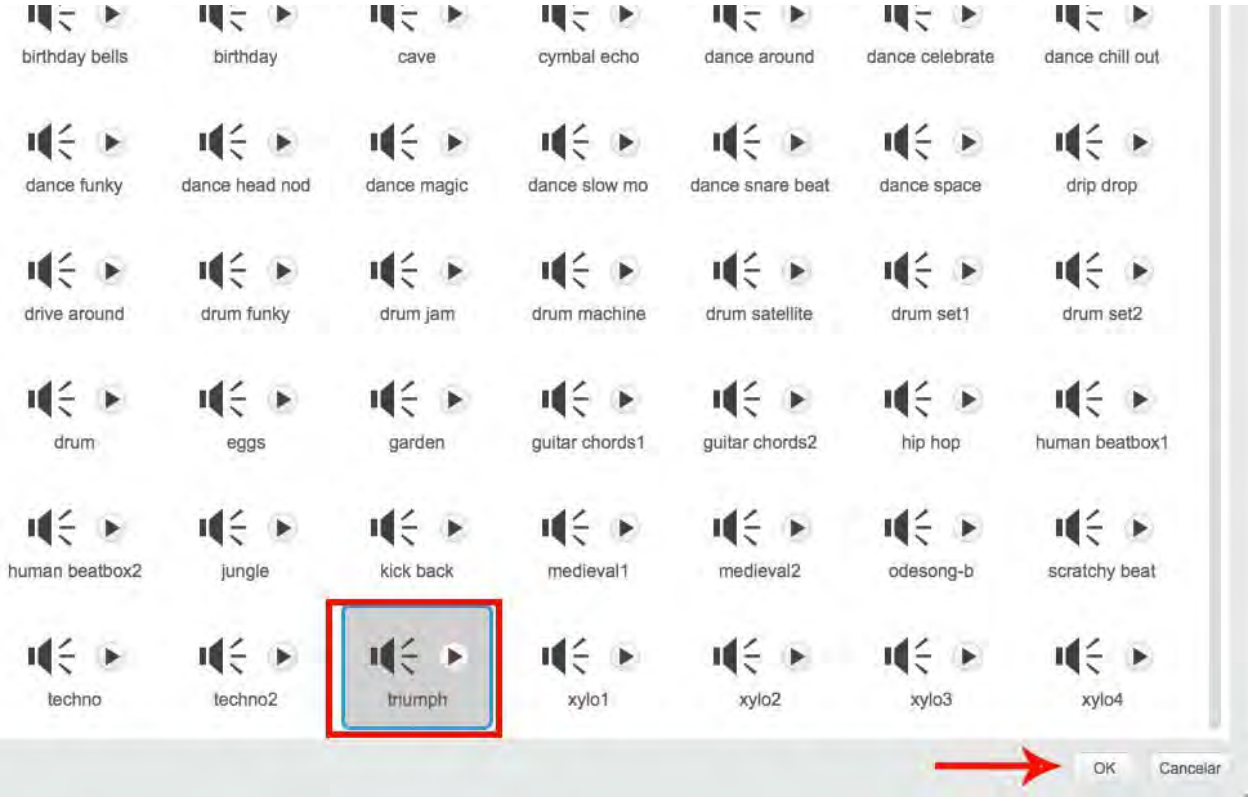


2. Vamos deixar o jogo mais divertido? Que tal se colocarmos um som que será executado toda vez que o procedimento for executado? Assim nosso jogo fica mais

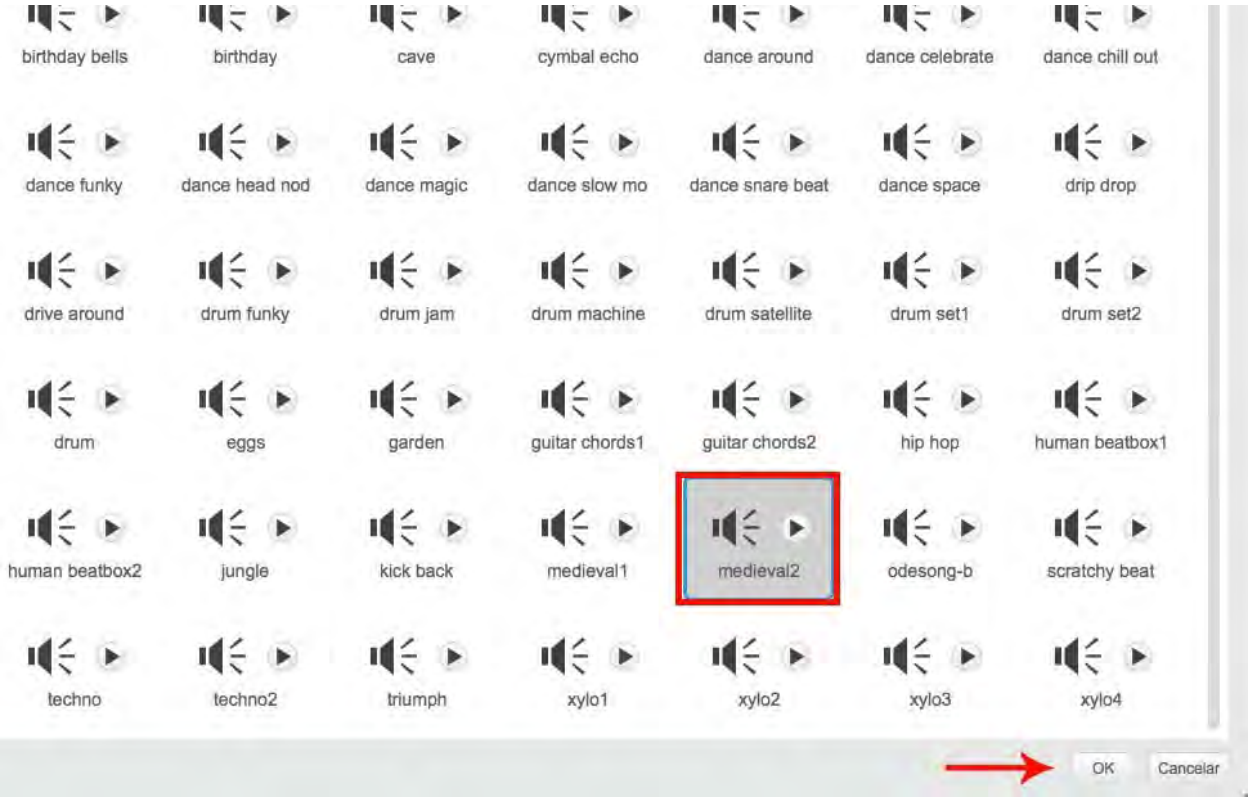
dinâmico, não acha? Você se lembra como adiciona um som? Se não, volte no *capítulo 3* e relembre como fazer.



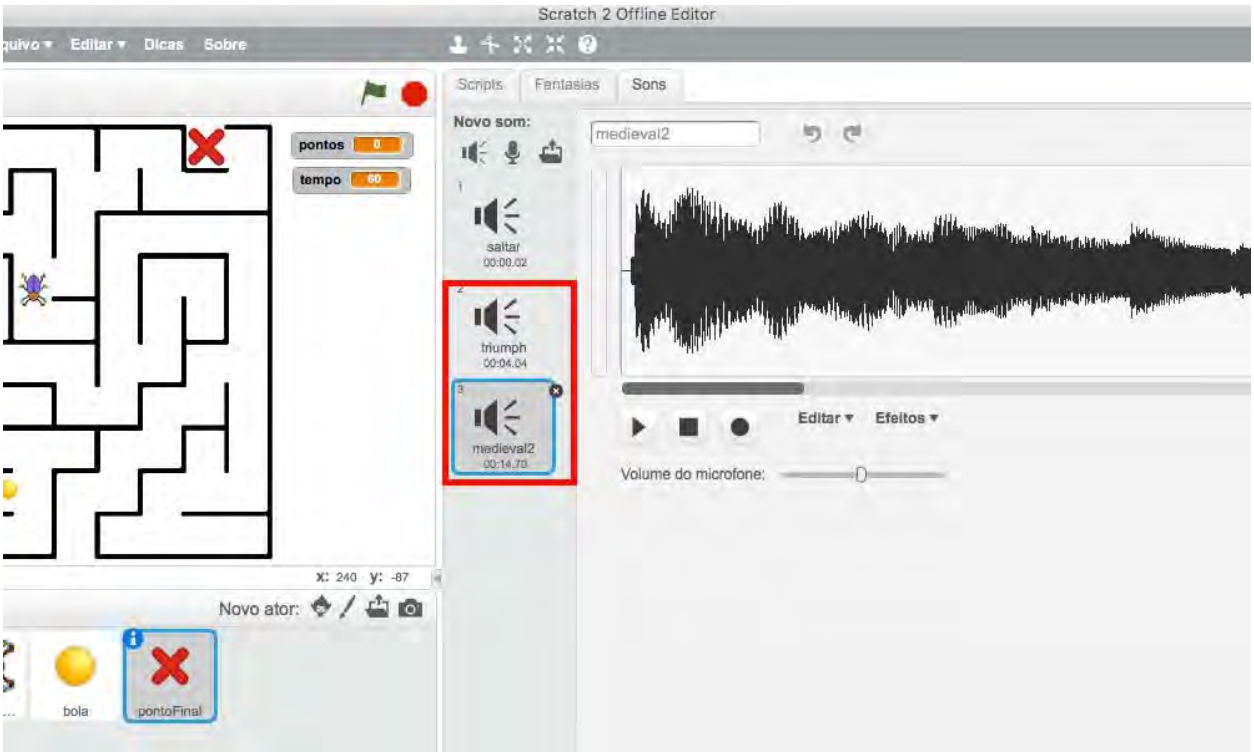
3. Adicione o som triumph. Este é uma sugestão, mas você pode escolher qualquer um destes disponíveis.



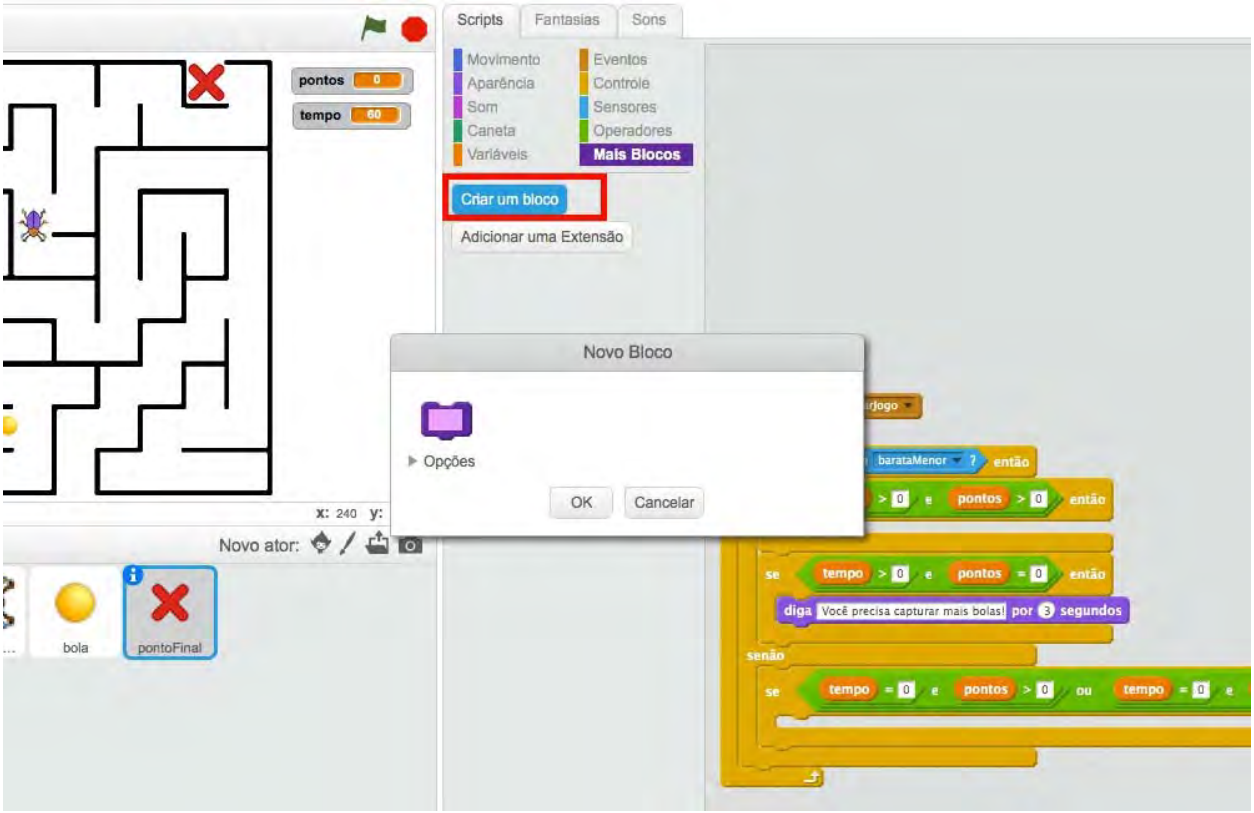
4. Adicione também o som medieval2, ou escolha um que você goste.



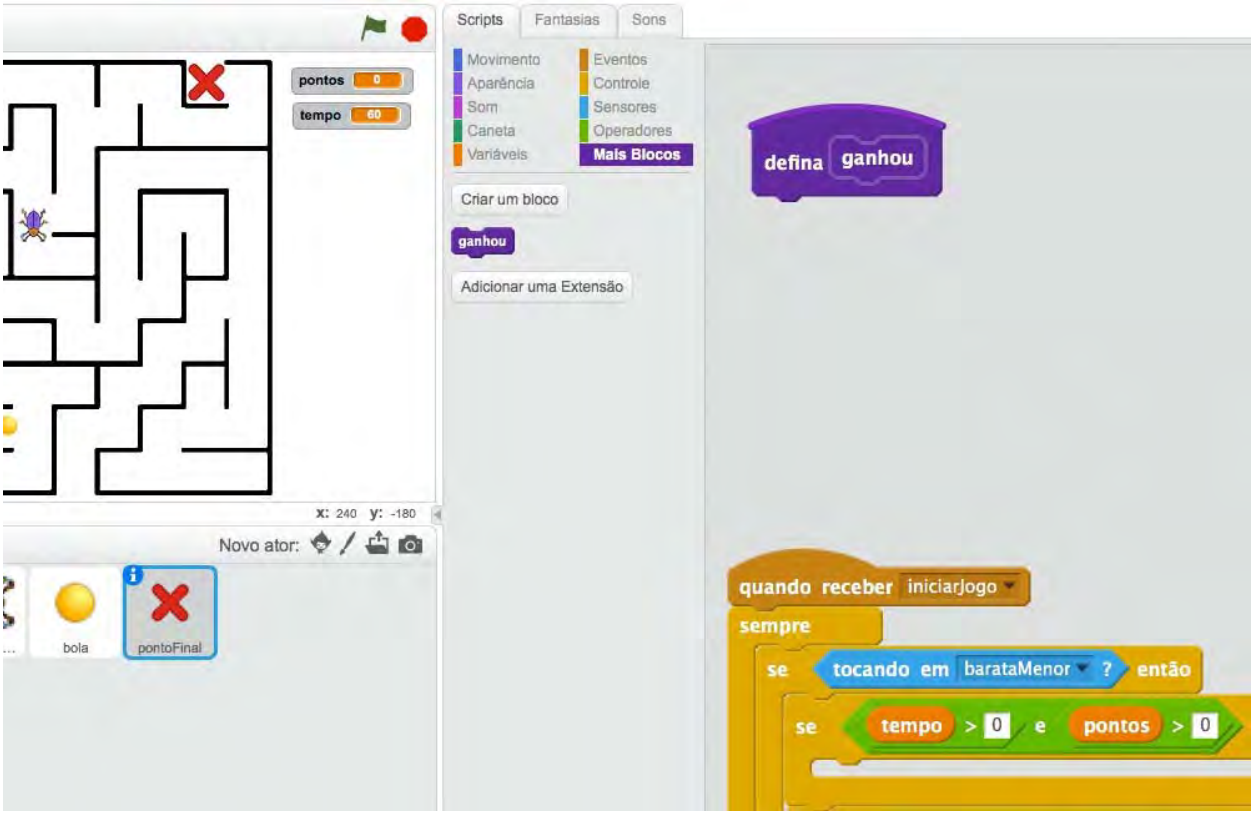
5. Veja a disposição dos dois e verifique se está como na imagem. Não é necessário que esteja nessa ordem necessariamente, mas é necessário que os dois apareçam nesta área, pois isso significa que o som foi adicionado corretamente.



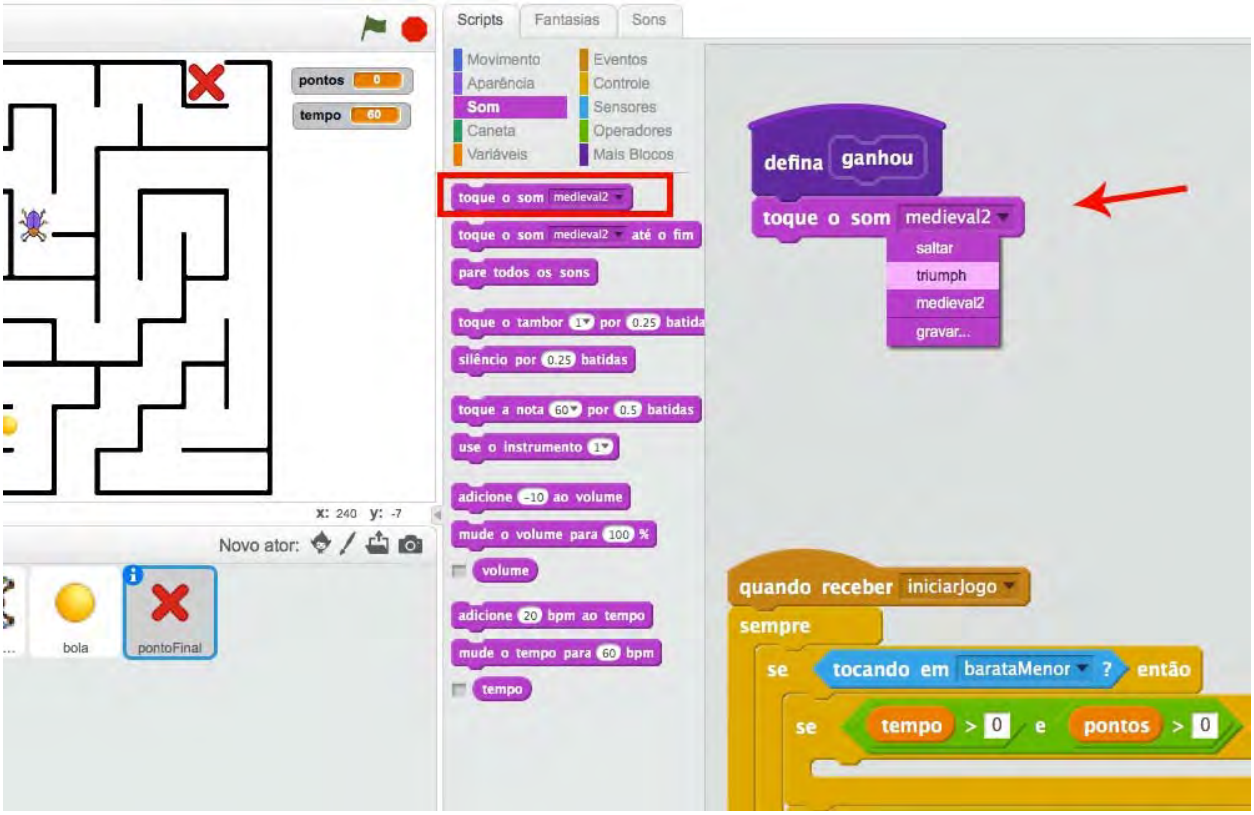
6. Vamos agora criar o primeiro procedimento. O primeiro procedimento será um pedaço de código que vai ser executado toda vez que o jogador "ganhar". Precisamos dele para que nosso jogo tenha um fim. Para isso, vá em Mais Blocos e clique em Criar um bloco. Quando abrir a janela, digite a palavra ganhou.



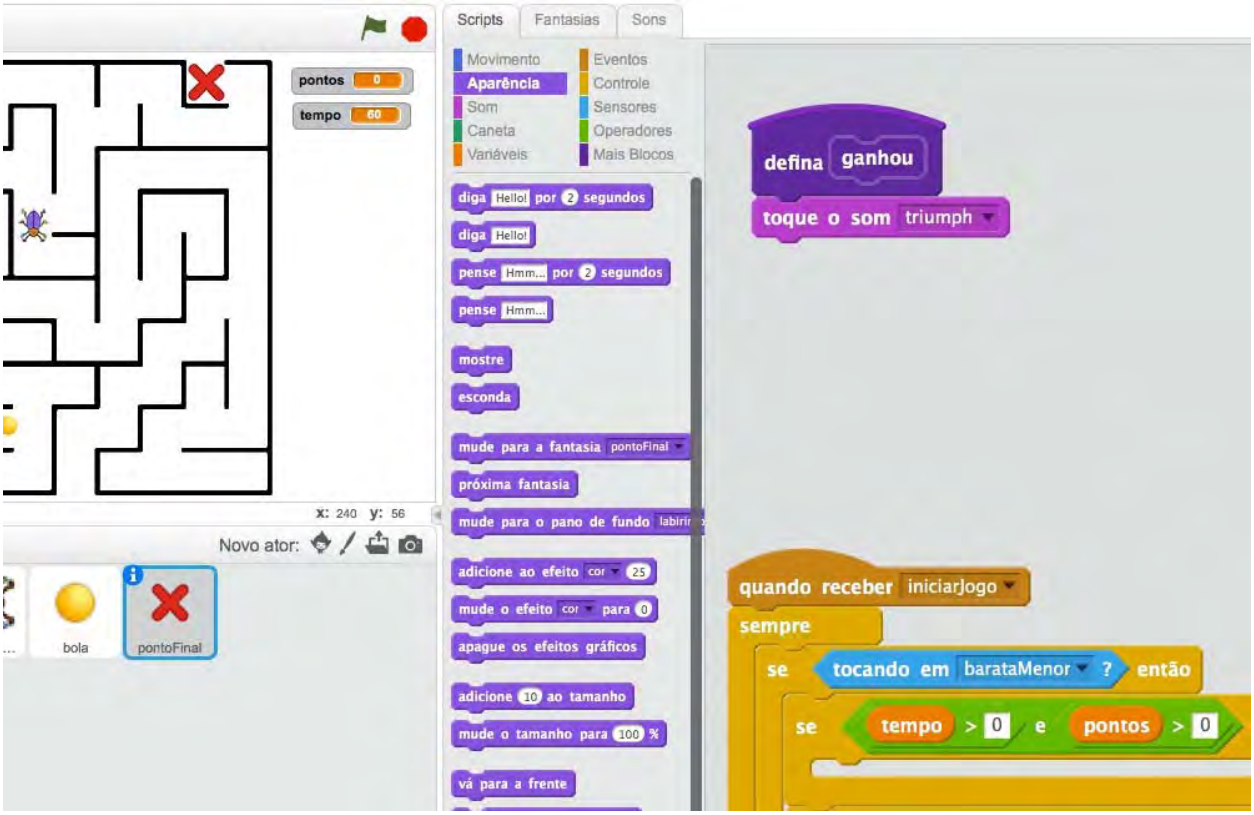
7. Veja como ficará:



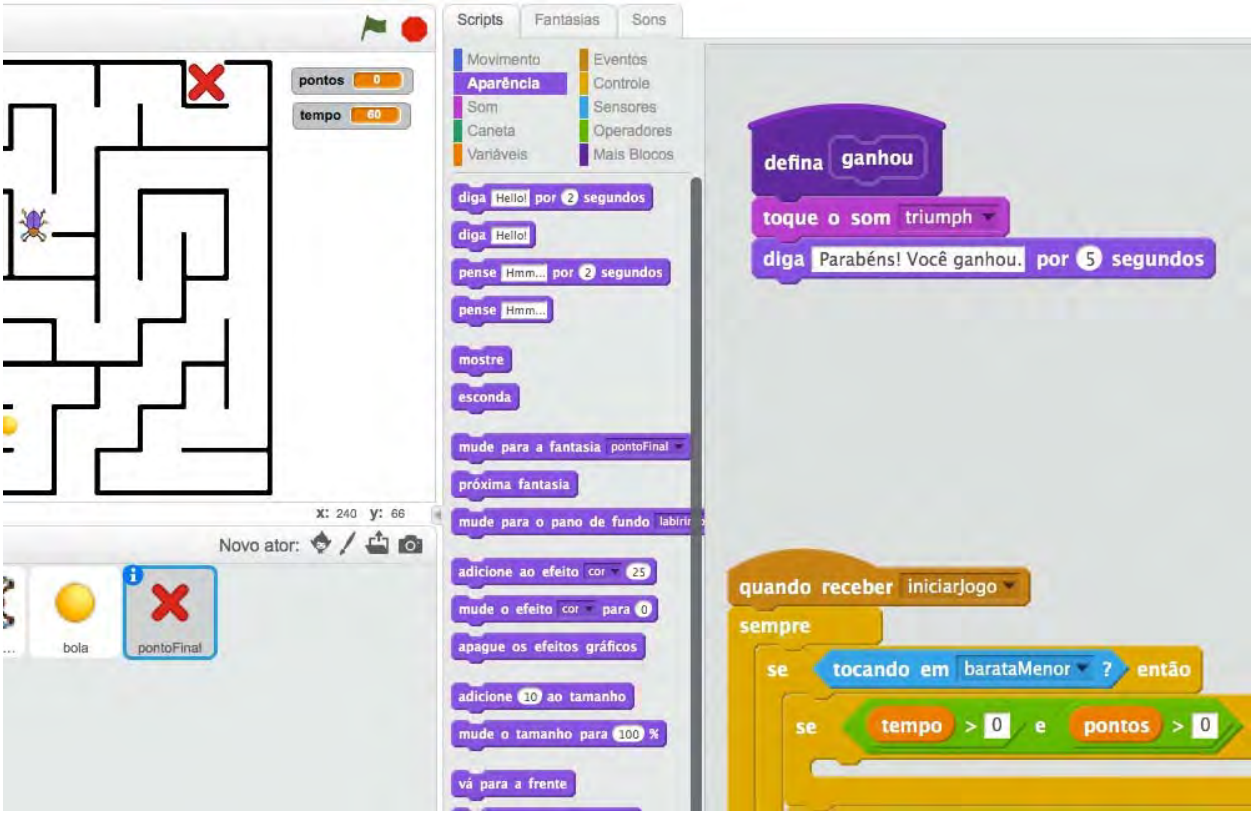
8. Seria legal se tocasse um som quando o jogador ganhasse, não? Então vamos adicionar o som. Vá em Som e adicione o bloco toque o som. Se necessário altere para triumph.



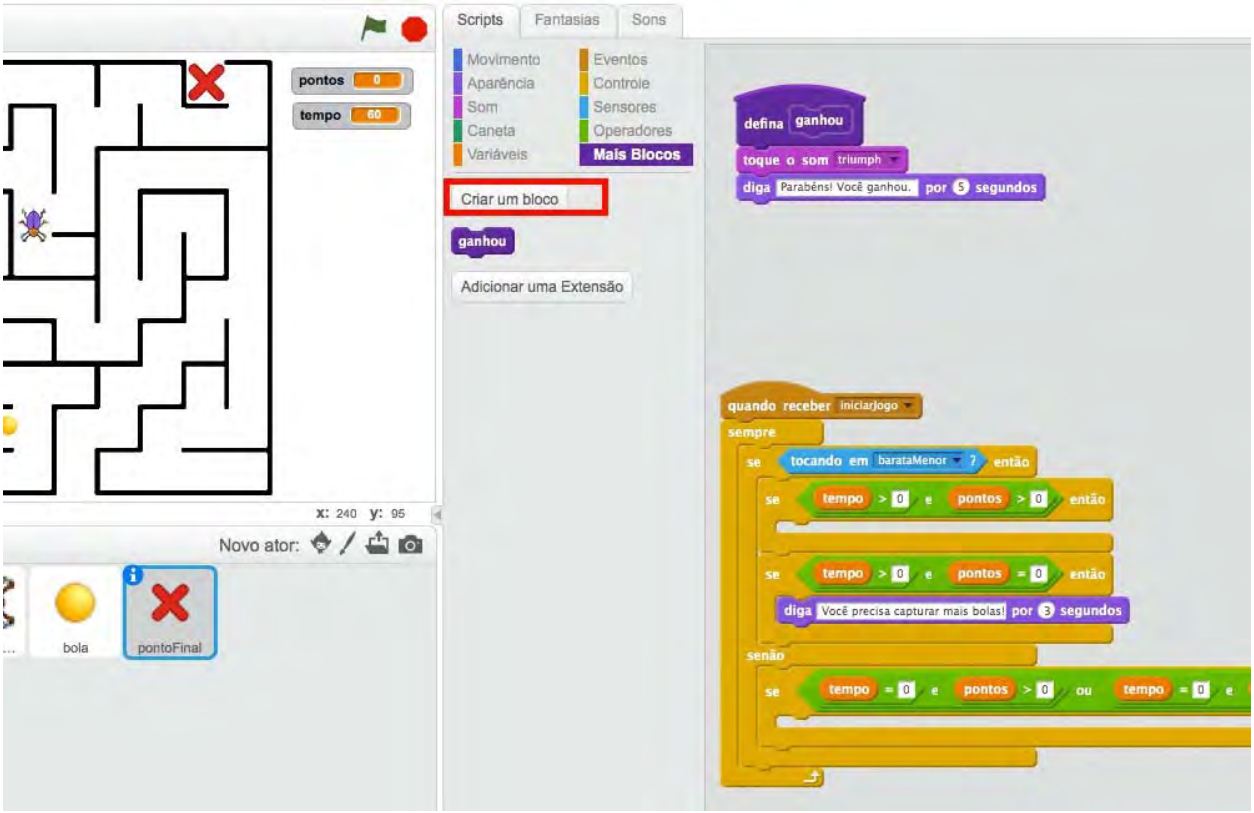
9. Dessa forma:



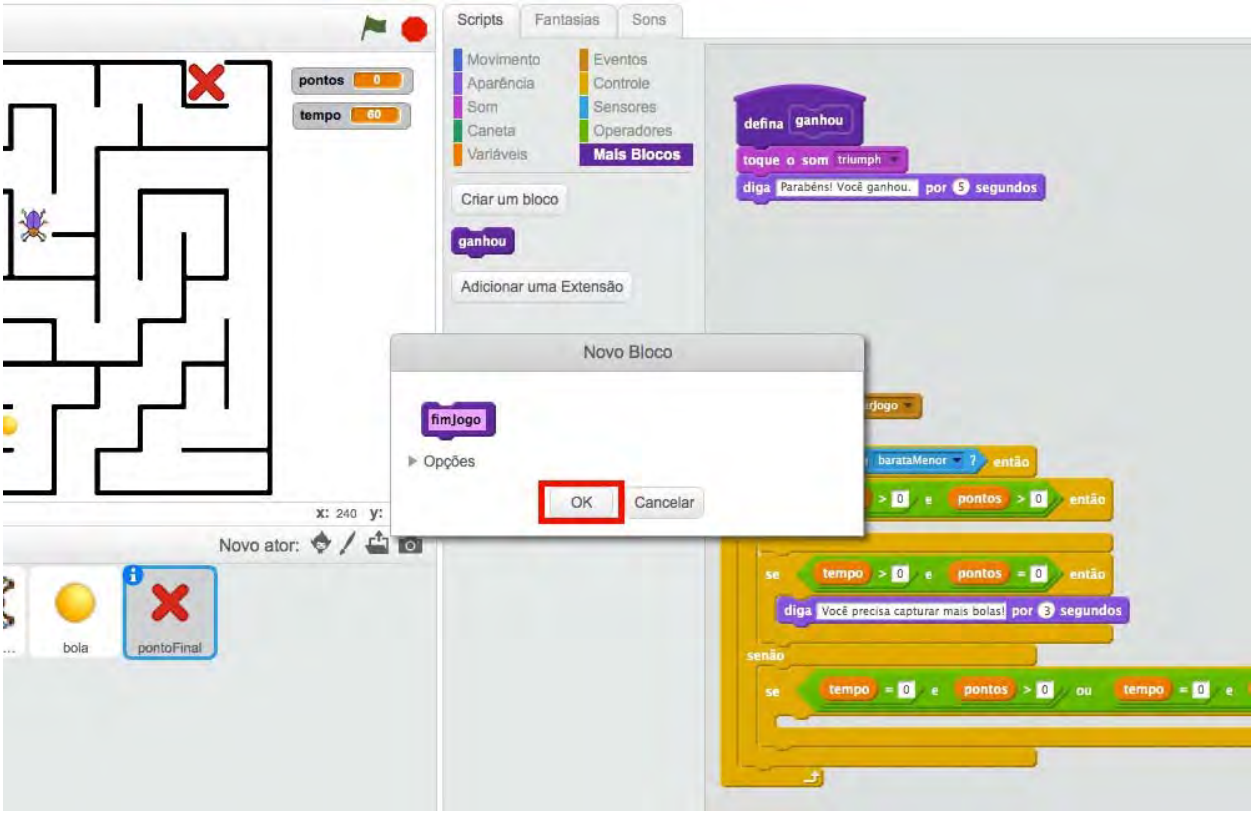
10. Vá agora em Aparência e arraste o bloco diga por x segundos. Altere a mensagem como está a seguir. Desta forma, fazemos com que, quando jogador ganhar o jogo, sejam executados o som e a mensagem! Legal, não? Vamos programar a seguir o bloco responsável pelo fim de jogo.



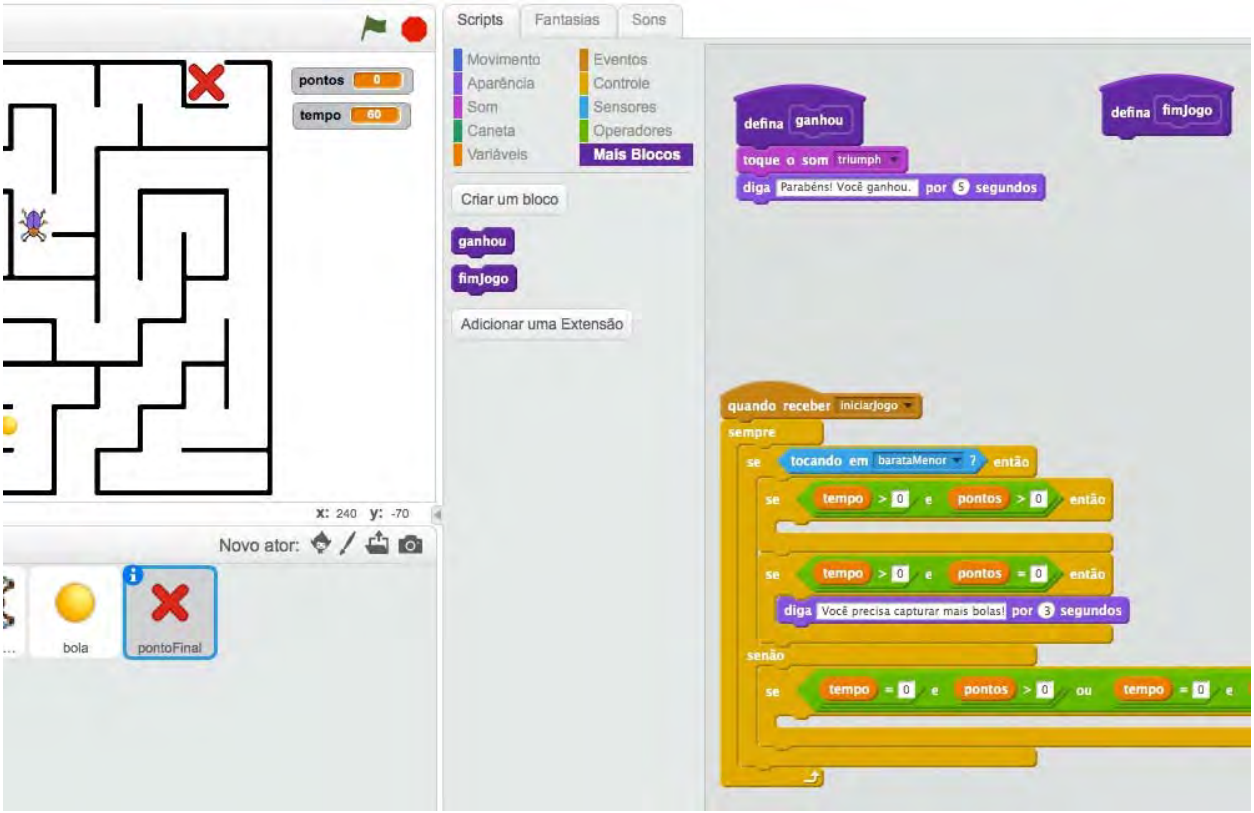
11. Vamos criar mais um bloco, dessa vez o procedimento fimJogo. Clique em Criar um bloco. Recomendo a você que tente fazer esta parte sozinho, apenas aplique o que você aprendeu de procedimentos e depois confira. Caso sinta dúvidas, continue seguindo o passo a passo.



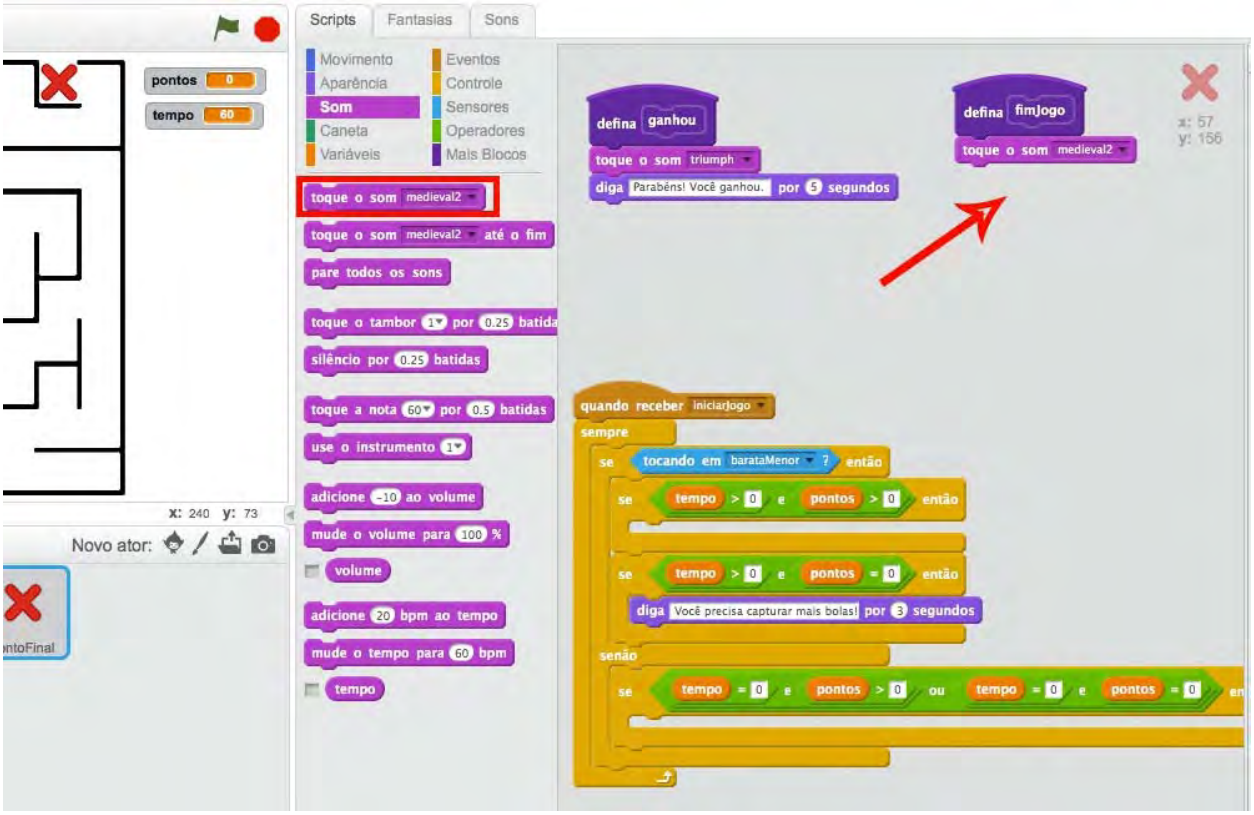
12. Digite fimJogo.



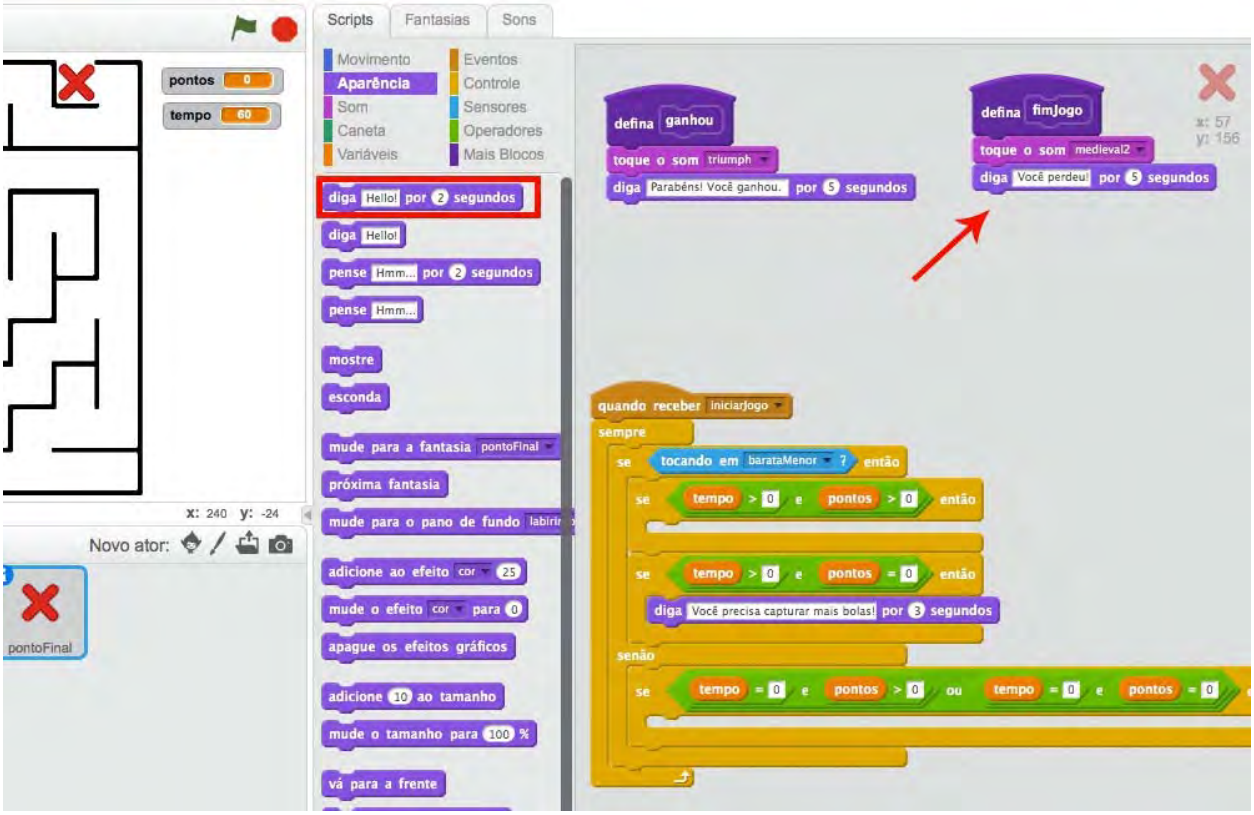
13. Ficará dessa forma. Muito parecido com o procedimento ganhou.



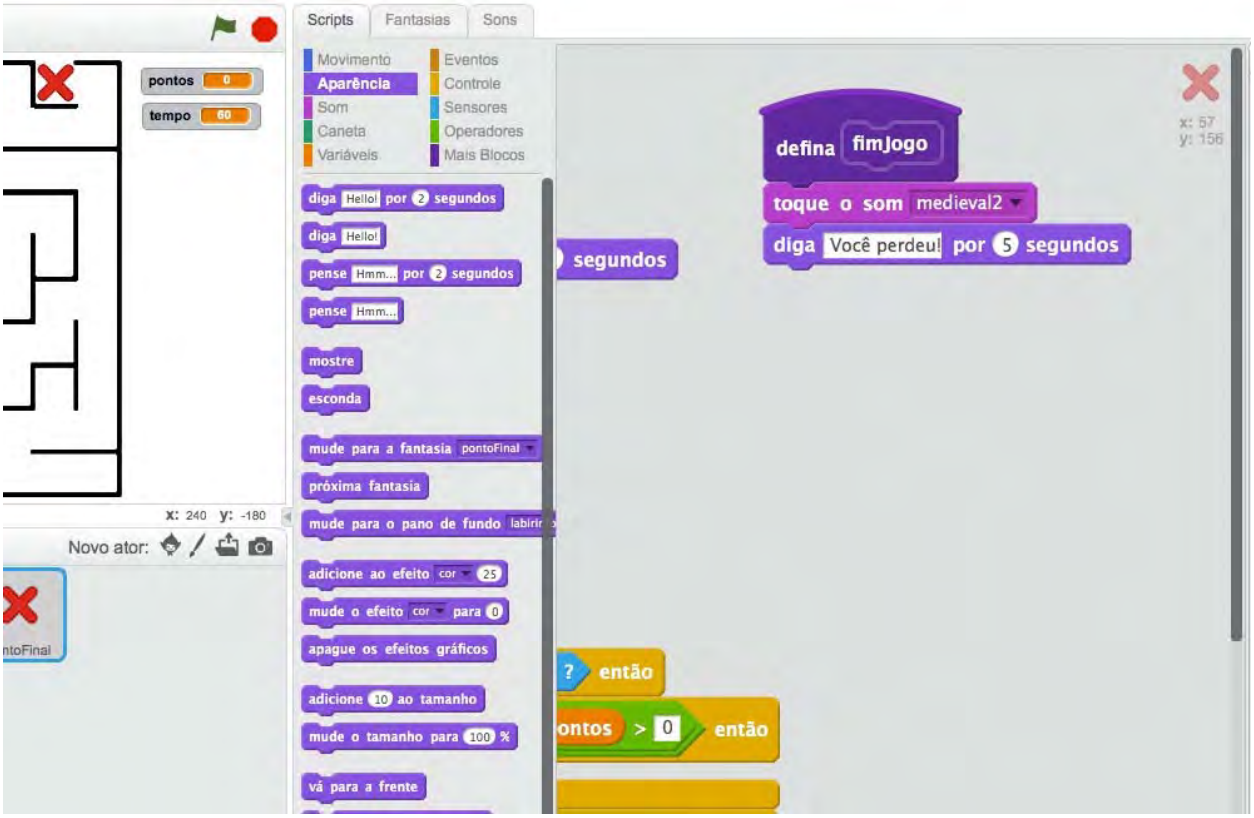
14. Novamente, vá em Som, adicione o bloco toque o som e altere se necessário para medieval2, ou para qualquer outro som que você desejar.



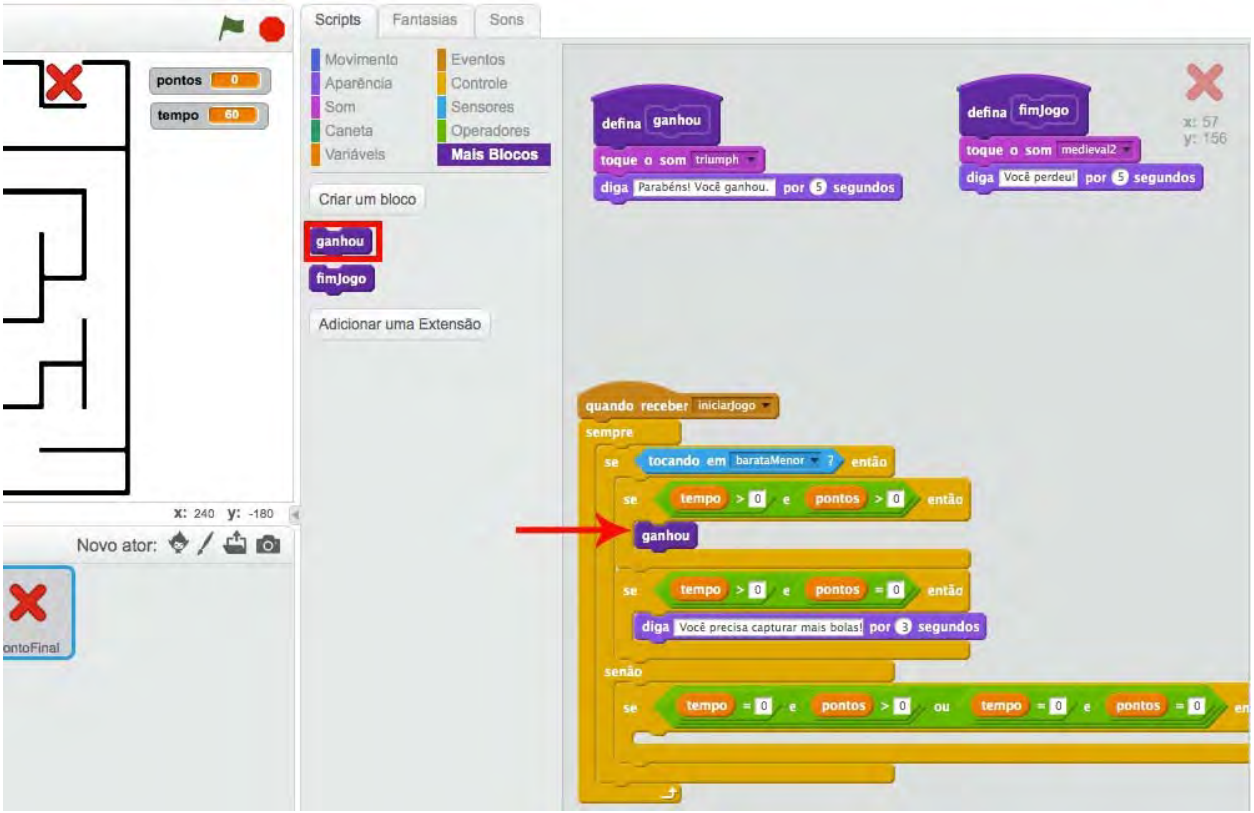
15. Altere a mensagem e o tempo como na figura a seguir.



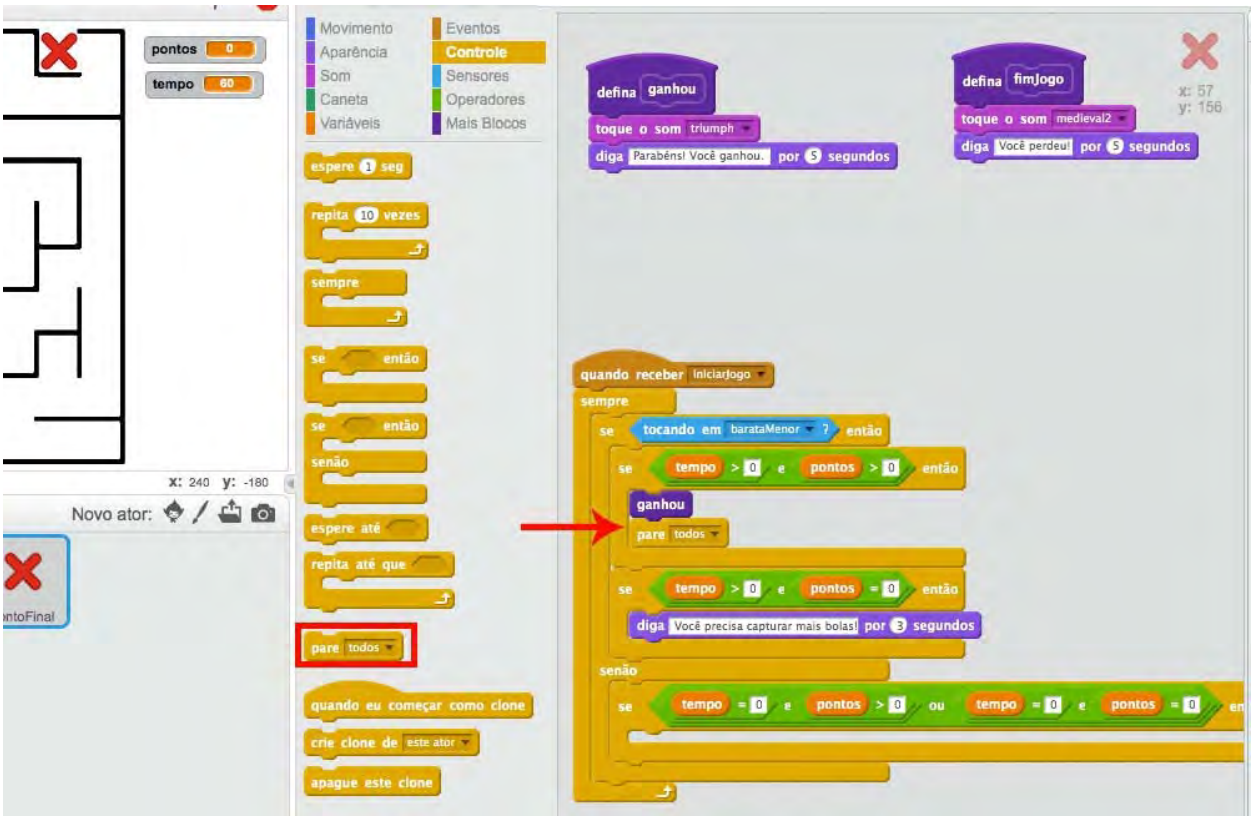
16. Ficará dessa forma:



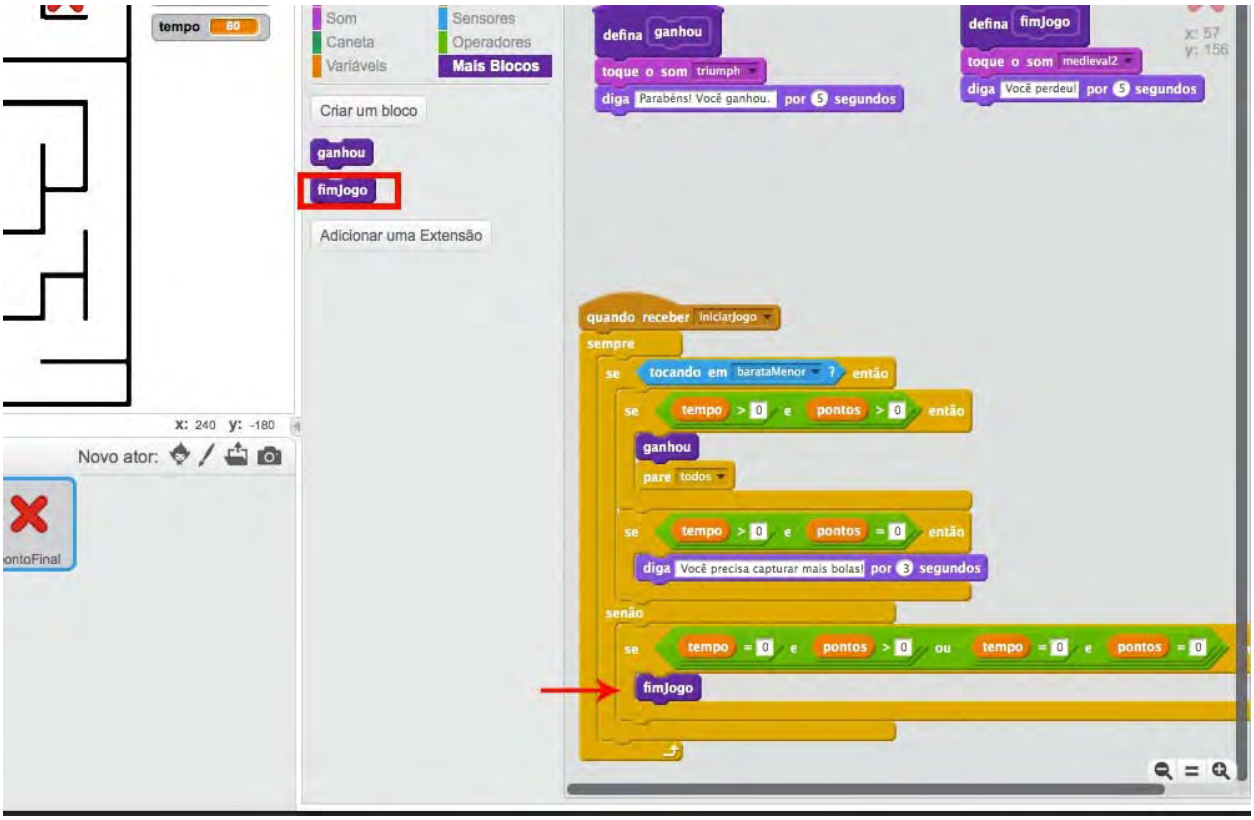
17. Com os procedimentos criados, precisamos colocar os blocos responsáveis por realizar a chamada do procedimento. Em Mais Blocos, arraste o bloco ganhou para dentro do se então conforme a seguir:



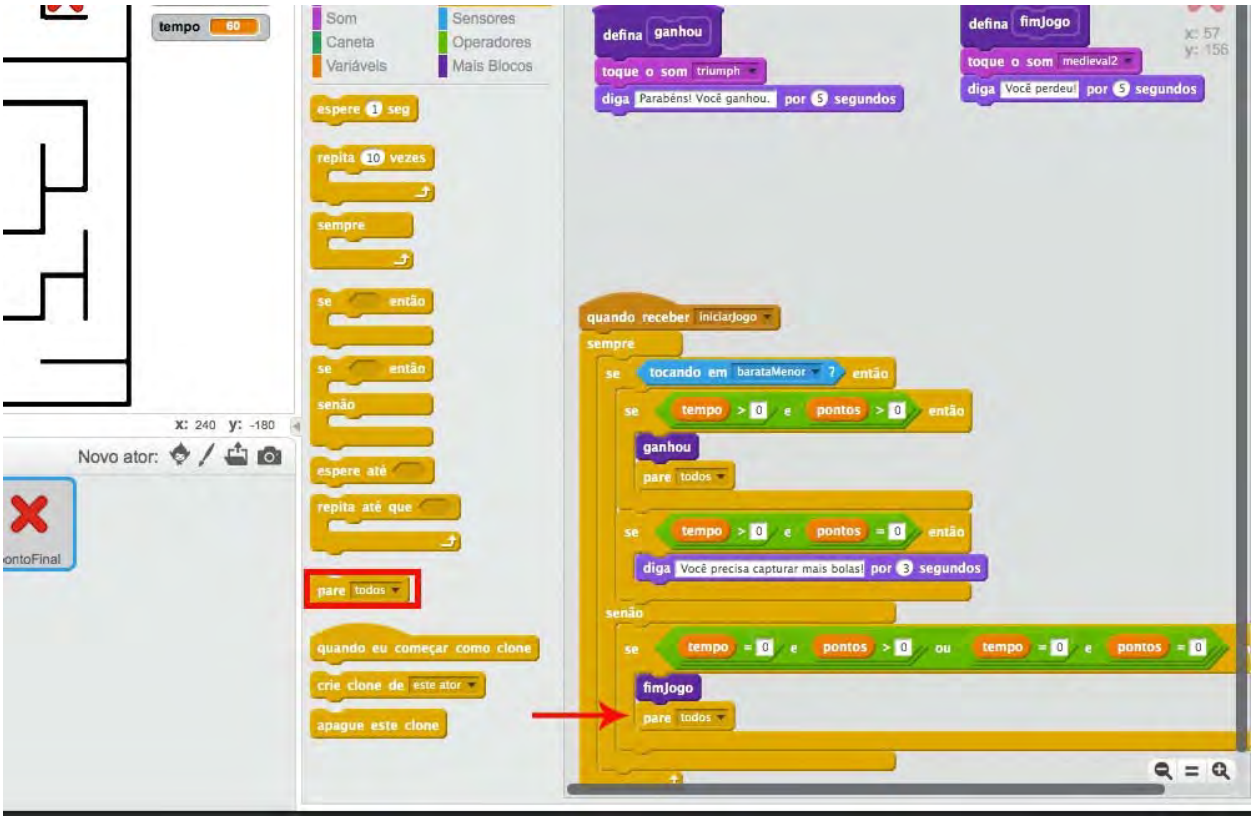
18. Realizar a chamada do procedimento ganhou significa que o jogo encerrou, logo precisamos parar a execução de todos os scripts. Para isso, vá em Controle e arraste o bloco pare todos para baixo da chamada do procedimento ganhou.



19. Vamos fazer o mesmo para o fimJogo. Em Mais Blocos, arraste a chamada fimJogo conforme na figura seguinte.



20. Arraste também o bloco pare todos. Este é responsável por parar a execução de todos os scripts, encerrando desta forma o jogo. Se você desejasse parar a execução de somente um script, bastaria clicar na setinha do bloco e selecionar a opção desejada.



Nos procedimentos ganhou e fimJogo, são executados um bloco de mensagem e um de som. Ou seja, quando o jogo encerrar (não importa se venceu ou perdeu), será exibido uma mensagem informando e executando um som conforme o resultado.

5.3 Conclusão

- Neste quinto capítulo, vimos como trabalhar com procedimentos e reaproveitar código, para deixar mais organizado e limpo.
- Para ficar mais divertido, colocamos o fundo musical.
- Encerramos o jogo de labirinto.

Diante de todas estas ferramentas que você aprendeu, a possibilidade de criações agora é enorme. Antes de iniciar os próximos capítulos, desafio você a criar o seu próprio jogo. Pesquise algumas ideias, esboce algo no papel e tente criar algo interessante. E que tal se você colocasse online para que todos possam ver e jogar o seu jogo?

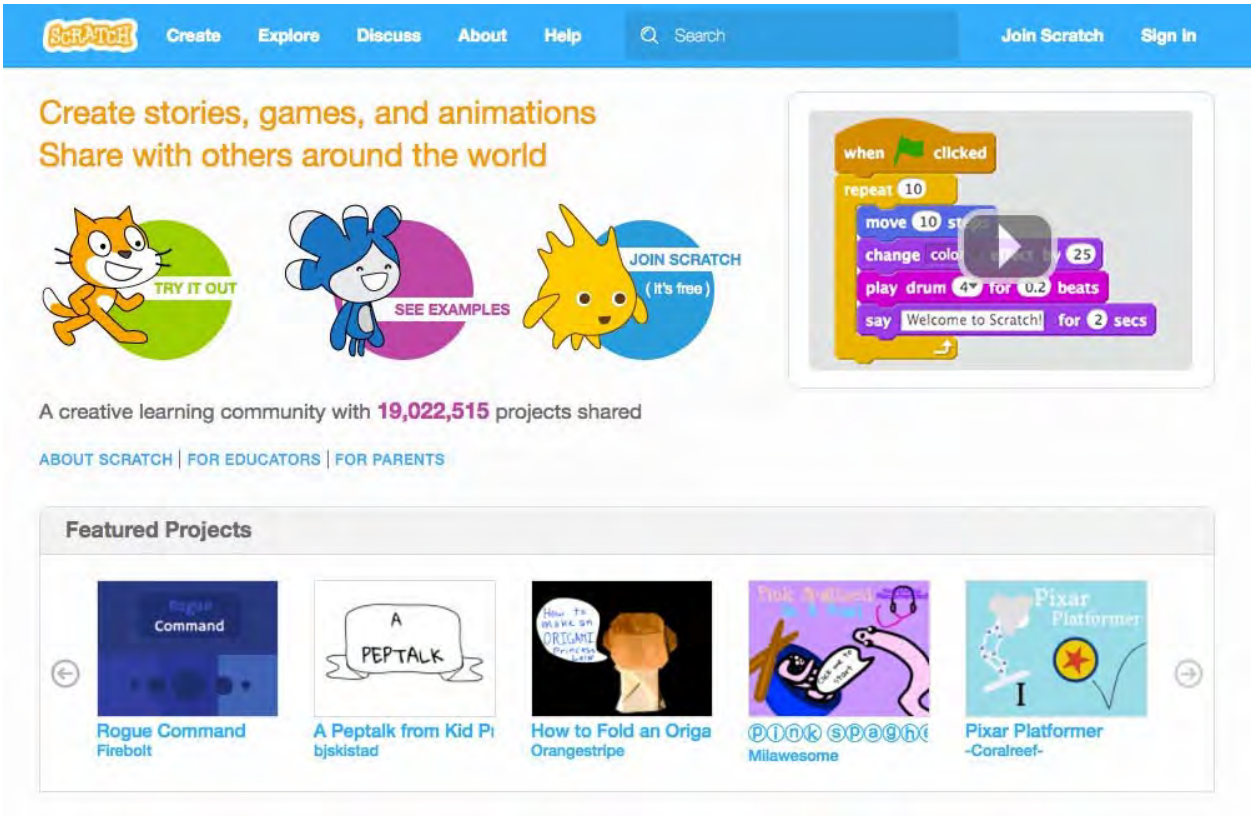
Imagine uma comunidade com milhões de pessoas e o seu jogo estará disponível para que elas joguem. Você poderá também divulgar o seu jogo para os seus amigos e família, e mostrar que você está programando e criando jogos próprios. Portanto, no próximo capítulo, veremos como criar uma conta no site do Scratch e colocar o jogo online.

CAPÍTULO 6

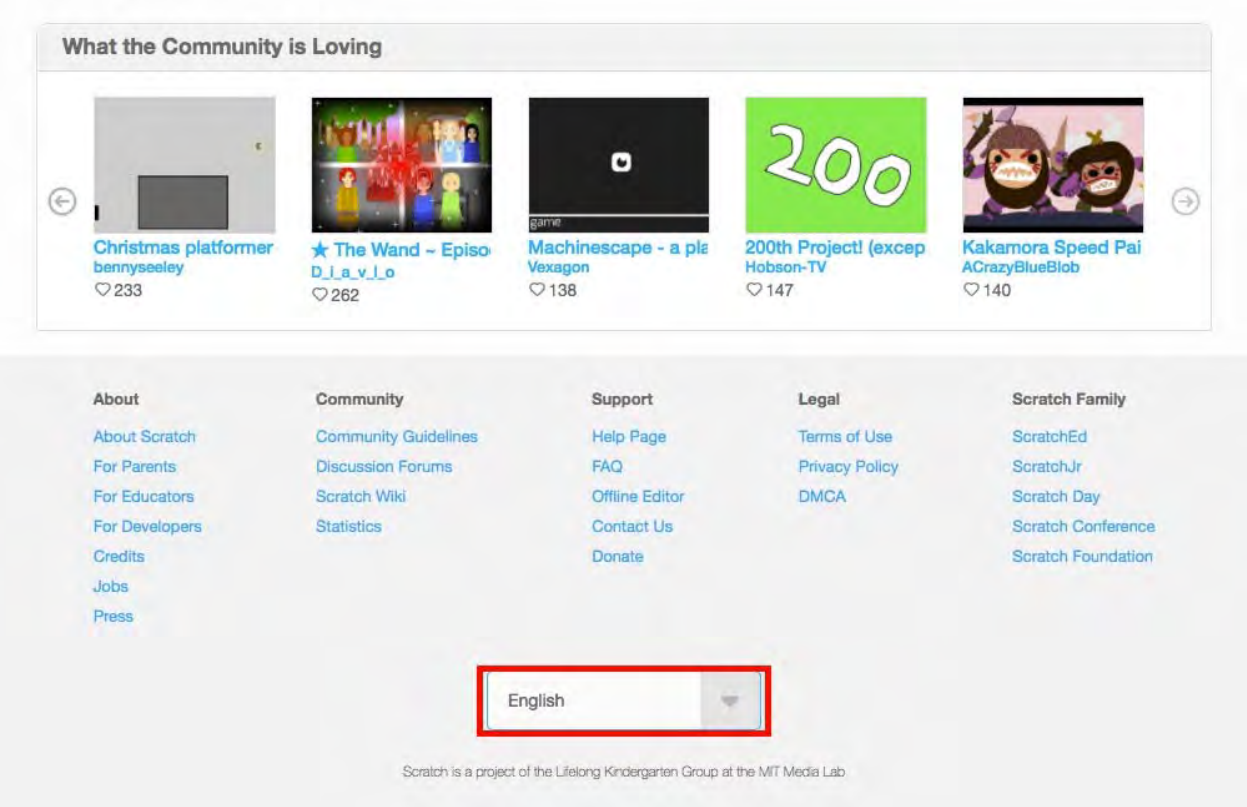
Compartilhando projetos no site

Com o jogo finalizado e tudo funcionando, vamos aprender agora a compartilhar o projeto no site, para que outros usuários possam jogar.

1. Acesse o site oficial do Scratch em: http://scratch.mit.edu.



2. Caso a página esteja em inglês, coloque em português rolando a página para baixo. Clique na caixa e selecione Português Brasileiro.



3. Para compartilhar os projetos, é necessário criar uma conta. Para isso, clique em Inscreva-se.

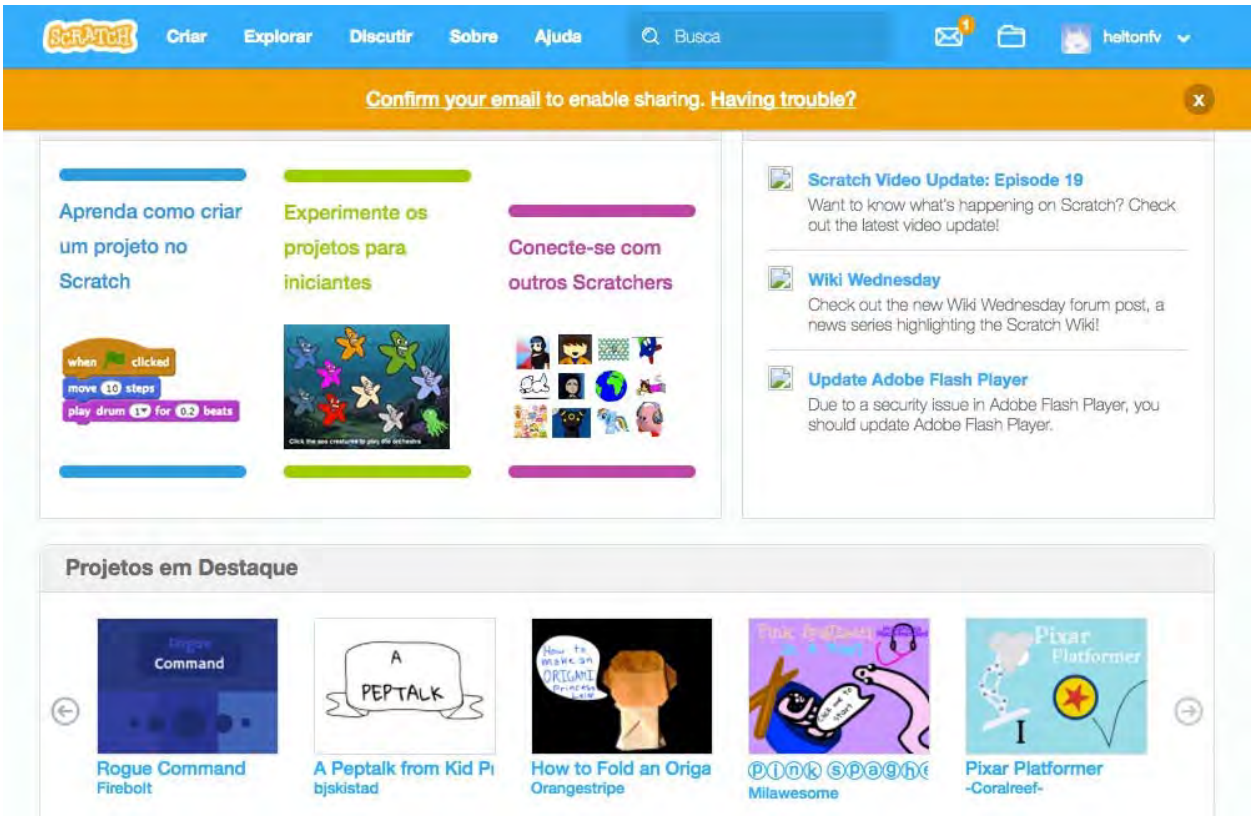
4. Escolha um nome de usuário e uma senha. Anote esta senha, pois ela é muito importante para você publicar os seus jogos online.

5. Novamente, preencha com seus dados: sua data de nascimento, sexo e o país onde você mora.

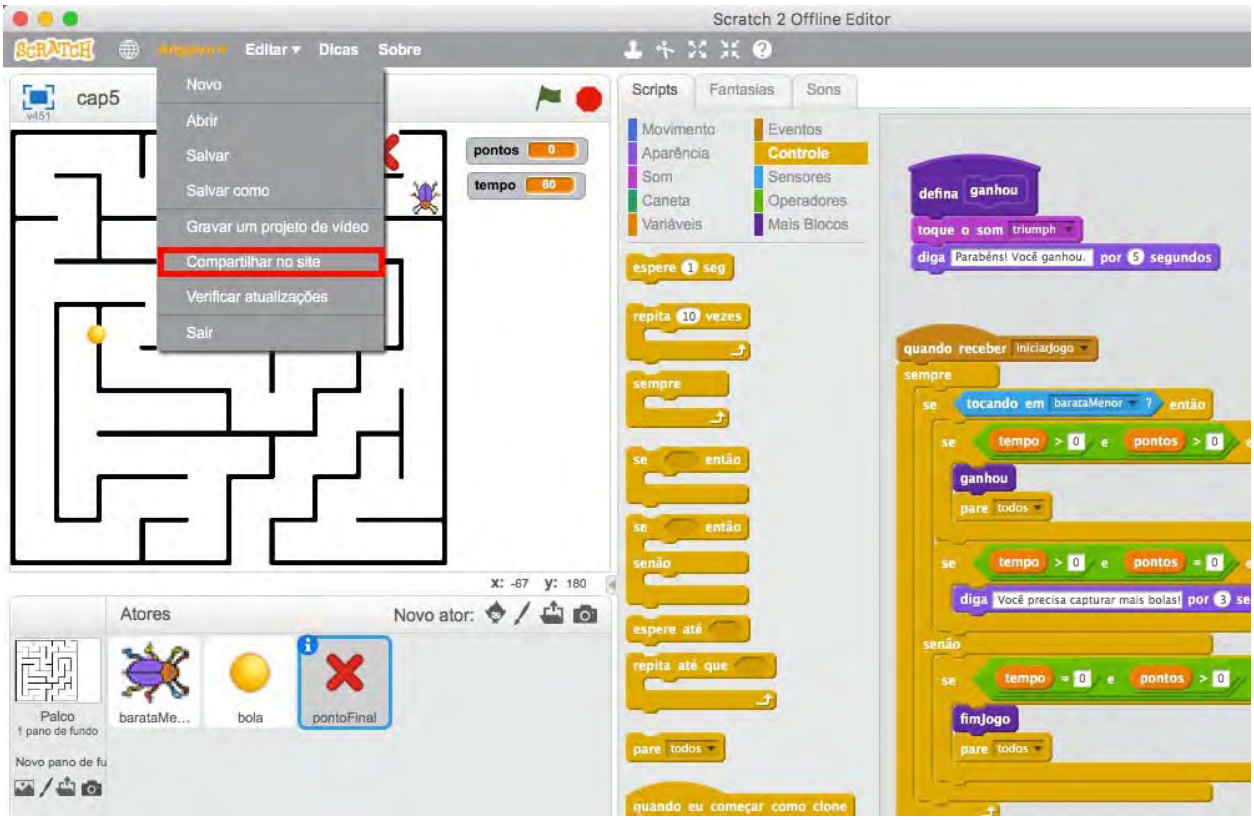
6. Digite seu endereço de e-mail e confirme.

7. Agora, basta clicar em Ok, vamos lá!.

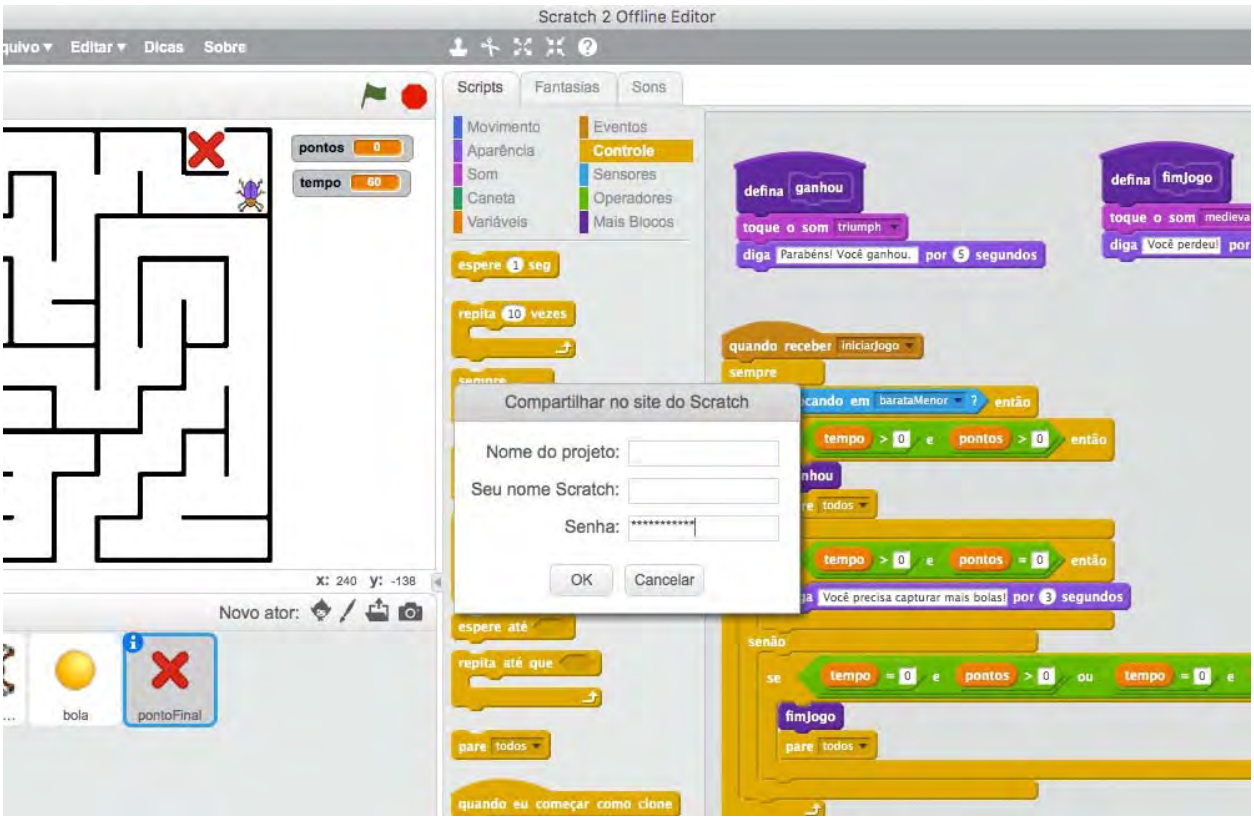
8. Será necessário confirmar seu e-mail. Para isso, entre na sua conta de e-mail, acesse sua caixa de entrada, e clique no link que você recebeu para confirmar.



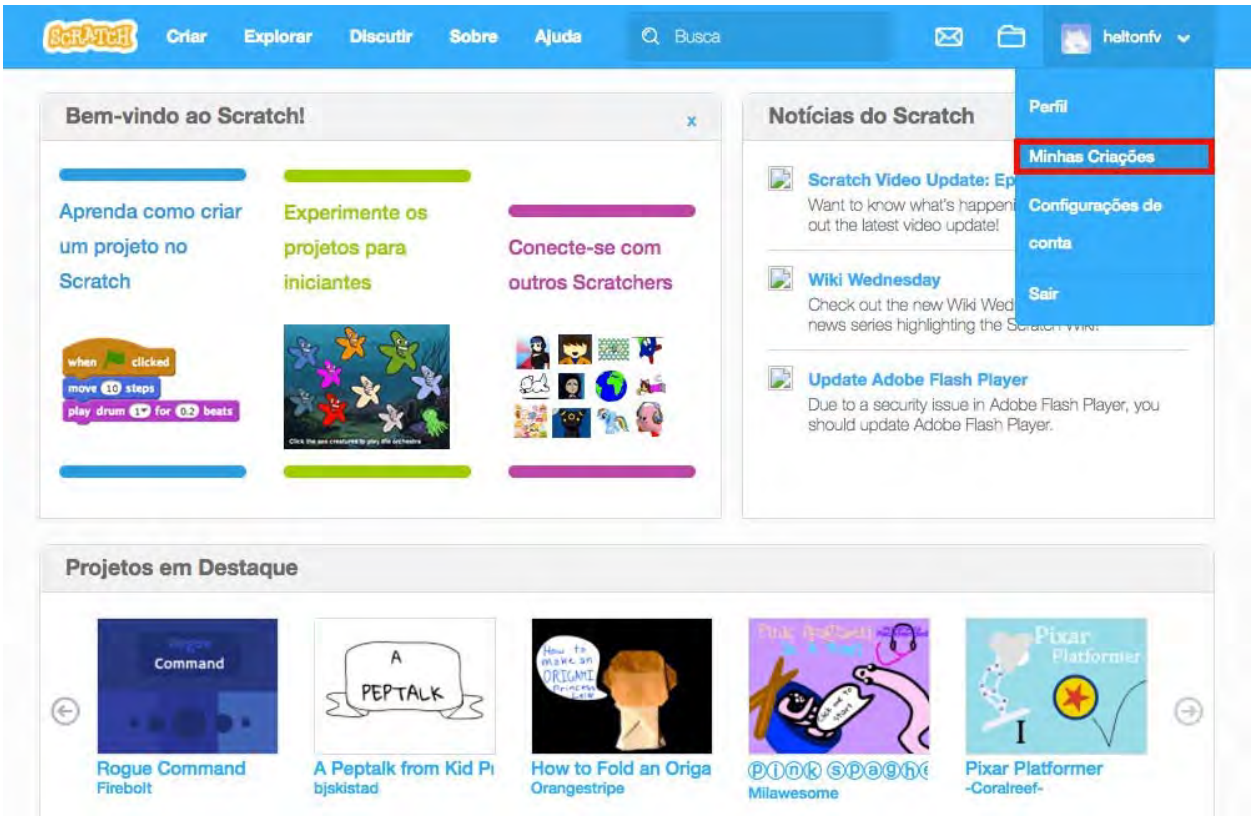
9. Agora com o arquivo do jogo aberto, clique em Arquivo e clique em Compartilhar no site.



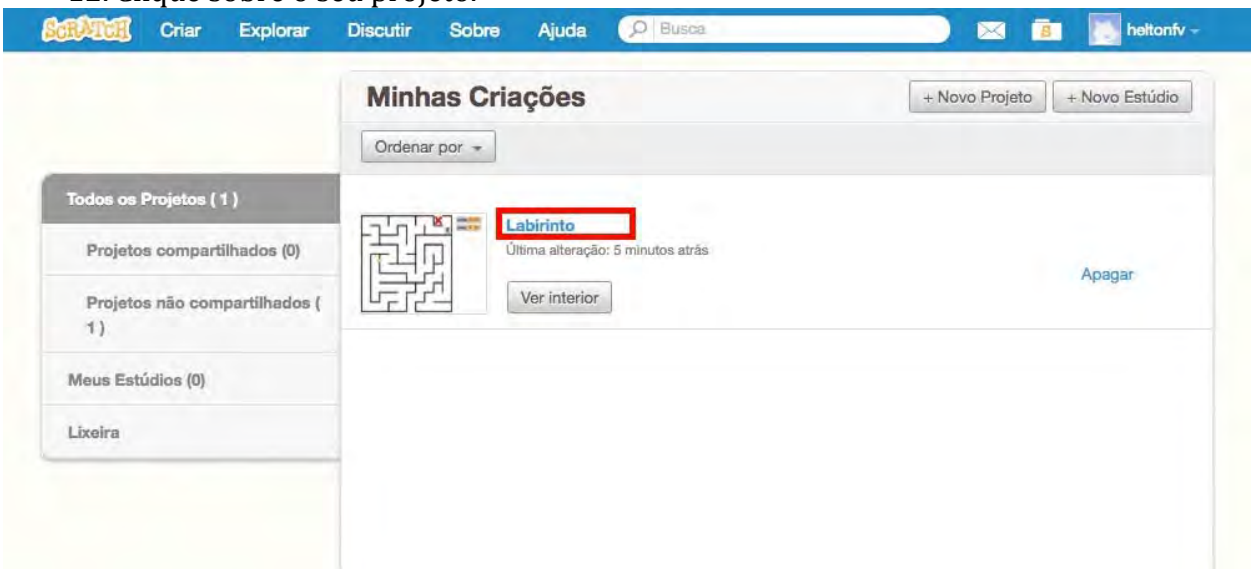
10. Escolha o nome do projeto e digite seus dados para confirmar. Clique em Ok e espere a mensagem de confirmação.



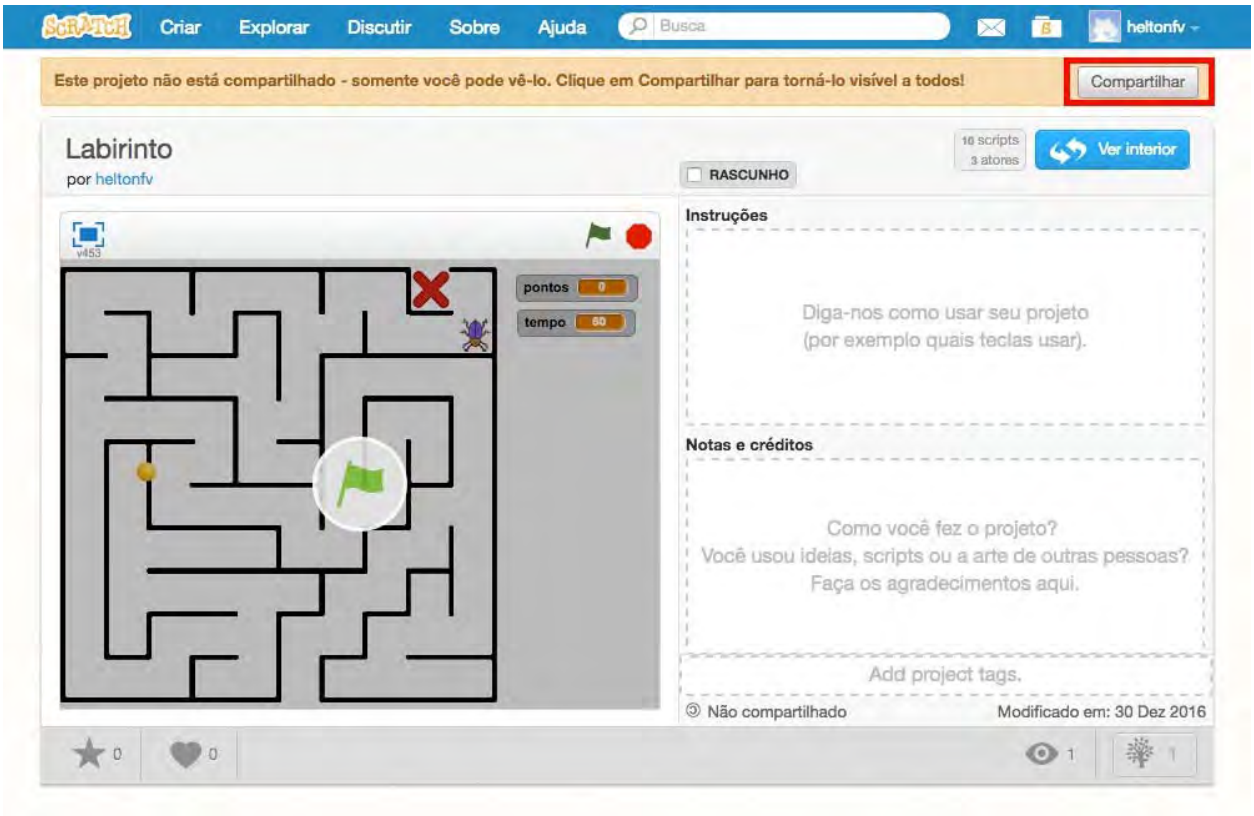
11. Voltando ao site, clique sobre seu nome de usuário e clique em Minhas criações.



12. Clique sobre o seu projeto.



13. Agora você pode editar algumas informações sobre o seu projeto, como Instruções e Notas. Após digitar todas as informações, clique em Compartilhar.



Dessa forma, o seu projeto está online e é possível compartilhar o link com seus amigos.

6.1 Conclusão

Muito fácil colocar o jogo online, não é mesmo? E se você estiver com alguma dúvida, problema ou sugestão, no próximo capítulo coloquei algumas informações de contato. Espero que tenha gostado deste conteúdo introdutório e que seja apenas o início de uma caminhada na área da programação.

CAPÍTULO 7

Links e contato

Se você desejar, acesse nosso blog e confira também a videoaula em que apresento o Scratch e suas ferramentas: http://scratchdicas.wordpress.com.

Ficou curioso e quer conferir como é o jogo pronto? Acesse http://bit.ly/2kw0ayW e baixe, jogue ou modifique.

Está começando o projeto e deseja utilizar as imagens do jogo? Acesse através do link e baixe: http://bit.ly/2kg0Z0o.

Você pode também entrar em contato comigo diretamente pelo meu e-mail: helton1@outlook.com. Estarei sempre disposto a lhe ajudar.