SUMÔ

Um jogo criado no Scratch

Introdução — Explorando o mundo da programação com o Scratch

A tecnologia faz parte da nossa vida todos os dias — nos celulares, computadores, videogames, redes sociais e até nos eletrodomésticos! Mas você já parou pra pensar em **como tudo isso funciona**?
Por trás de cada aplicativo e jogo existe um conjunto de **instruções escritas em linguagem de programação**, que dizem ao computador o que ele deve fazer.

Aprender a programar é como **aprender uma nova forma de pensar**: a gente aprende a resolver problemas, criar soluções e transformar ideias em realidade.

Foi pensando nisso que surgiu o **Scratch**, uma ferramenta criada pelo **MIT (Instituto de Tecnologia de Massachusetts)**, um dos centros de pesquisa mais importantes do mundo.
O Scratch foi desenvolvido para ajudar **crianças, adolescentes e iniciantes** a aprender programação de maneira **fácil, divertida e visual**, usando blocos coloridos que se encaixam como peças de um quebra-cabeça.

O que é o Scratch?

O Scratch é uma **plataforma gratuita** que permite criar **histórias interativas, animações, jogos e músicas**.
Em vez de escrever códigos complicados, o usuário **arrasta e encaixa blocos de comandos**, o que torna a programação mais acessível e intuitiva.

Cada bloco representa uma **ação**: mover um personagem, tocar um som, repetir uma tarefa, detectar colisões, entre muitas outras.
Ao unir esses blocos, o aluno **dá vida aos personagens**, criando projetos cheios de criatividade e lógica.

O Scratch pode ser usado **online** (no site scratch.mit.edu) ou **offline**, com o programa **Scratch Desktop**, que funciona sem internet.

Por que aprender com o Scratch?

Aprender a programar com o Scratch é mais do que aprender tecnologia — é **aprender a pensar!**
Enquanto criam seus próprios jogos e animações, os alunos desenvolvem habilidades importantes, como:

Raciocínio lógico: entender como organizar ideias e resolver problemas.

Criatividade: transformar imaginação em projetos reais.

Trabalho em equipe: o Scratch permite compartilhar e remixar projetos com outras pessoas.

Paciência e persistência: nem sempre o código dá certo de primeira, e isso faz parte do aprendizado.

Além disso, o Scratch ajuda a aproximar os alunos do **mundo da programação** de forma divertida, despertando o interesse por **ciência, tecnologia, engenharia e matemática (STEM)**.

Criando e aprendendo brincando

Neste livrinho, você vai aprender a criar **dois jogos incríveis** passo a passo, usando apenas o Scratch e sua criatividade!

Cada jogo foi pensado para ensinar novos conceitos de maneira **simples e divertida** — como mover personagens, criar pontuação, usar variáveis e controlar inimigos.

Prepare-se para aprender brincando!
Vamos começar com o jogo **"Barata Louca"**, onde você vai controlar uma barata corajosa que precisa escapar de um morcego e coletar maçãs pelo caminho.

Como usar este livro

Este livrinho foi feito para **te guiar passo a passo** no mundo da programação com o Scratch.
Aqui, você vai aprender **enquanto cria**, colocando a mão na massa e vendo suas ideias ganharem vida na tela! ✨

Para aproveitar ao máximo, siga estas dicas:

Leia cada etapa com atenção:
Cada parte do tutorial explica o que fazer e por que fazer. As instruções foram escritas de forma simples, para que você entenda o que cada bloco faz e como ele ajuda o jogo a funcionar.

Observe as imagens:
Ao lado de cada explicação, você encontrará **imagens dos blocos de programação** usados. Elas vão te ajudar a montar o código corretamente.

Entenda antes de copiar:
Mais importante do que repetir os blocos, é **entender o que eles fazem**. Assim, você vai poder criar seus próprios jogos no futuro!

Teste e explore:
Clique na bandeira verde e veja o que acontece! Se algo não funcionar, observe, pense e tente de novo. A programação é feita de **tentativas e descobertas**.

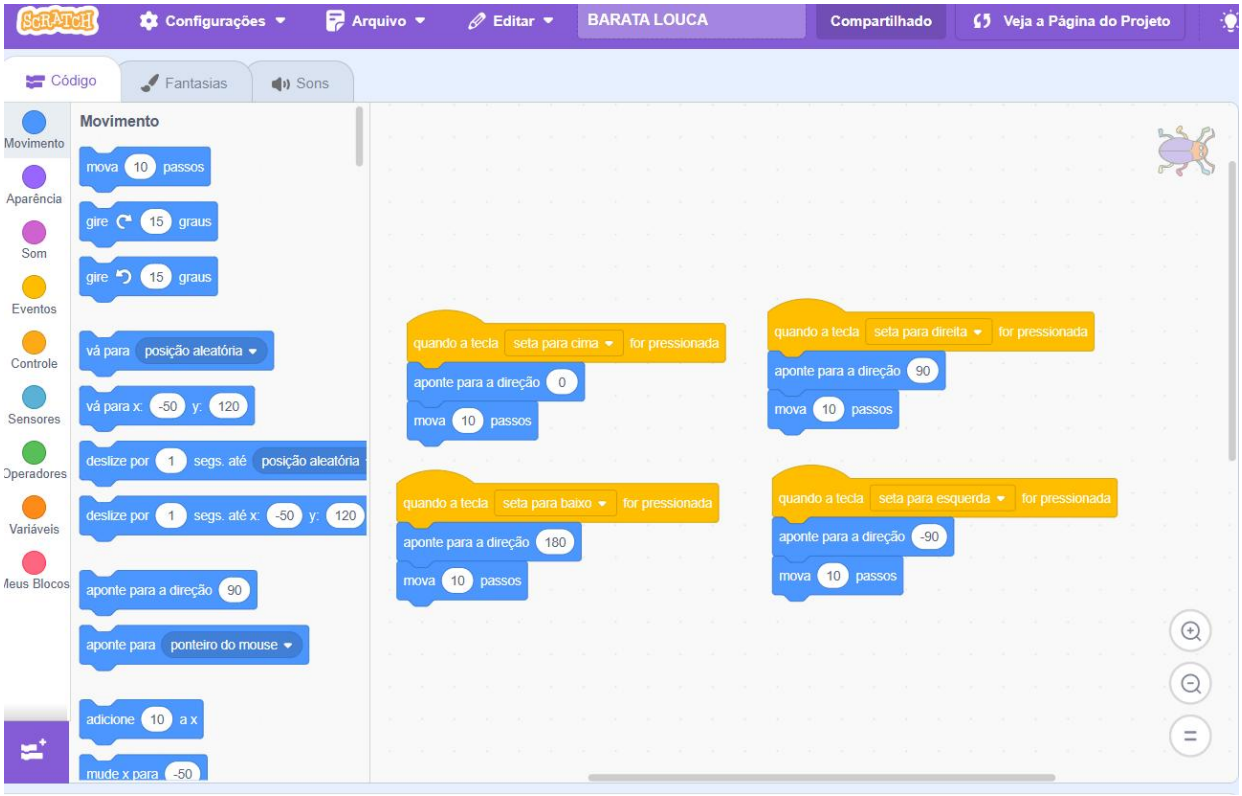
Use a criatividade:
Você pode mudar o fundo, trocar os personagens, inventar novas regras e até criar fases diferentes. O Scratch é um espaço para **imaginar e inovar**!

Compartilhe seus projetos:
Depois de terminar, publique seu jogo na comunidade do Scratch e mostre para amigos e colegas. Quem sabe alguém se inspire com a sua criação?

Com essas dicas, você já está pronto para começar sua jornada no mundo da programação!
Pegue seu computador, acesse o Scratch e prepare-se para se divertir aprendendo.
Nos próximos capítulos, você vai descobrir como criar jogos incríveis — começando com o divertido **"Barata Louca"**!

Movimento da Barata — Fazendo o personagem se mexer

Agora chegou a hora de dar **vida à nossa baratinha**!
Com a programação mostrada na imagem ao lado, vamos fazer com que ela **se mova em todas as direções** usando as **setas do teclado**.
Assim, o jogador poderá controlar a barata e fazê-la explorar o cenário do jogo.



“Programação da movimentação da barata no Scratch: cada bloco faz o personagem se mover em uma direção quando o jogador pressiona a tecla correspondente.”

Passo a passo da movimentação

No Scratch, cada **personagem (sprite)** pode ser programado para reagir a teclas do teclado.
Para isso, usamos o bloco **“quando a tecla ___ for pressionada”** (do grupo **Eventos**).
Esse bloco funciona como um **sensor**: ele detecta quando o jogador pressiona uma tecla e, então, executa os comandos que vêm logo abaixo dele.

Na imagem, temos **quatro blocos de evento**, um para cada direção:

Seta para cima

```
quando a tecla seta para cima for pressionada
aponte para a direção 0
mova 10 passos
```

➡ Faz a barata andar **para cima**.
O número **0** representa a direção “norte” no Scratch.

Seta para baixo

```
quando a tecla seta para baixo for pressionada
aponte para a direção 180
mova 10 passos
```

➡ Faz a barata andar **para baixo**.
O número **180** representa a direção “sul”.

Seta para a direita

```
quando a tecla seta para direita for pressionada
aponte para a direção 90
mova 10 passos
```

➡ Faz a barata andar **para a direita**.
O número **90** representa a direção “leste”.

Seta para a esquerda

```
quando a tecla seta para esquerda for pressionada
aponte para a direção -90
mova 10 passos
```

➡ Faz a barata andar **para a esquerda**.
O número **-90** representa a direção “oeste”.

Entendendo o que está acontecendo

No Scratch, o palco onde o jogo acontece tem **um sistema de coordenadas**, parecido com o que usamos na matemática:

O eixo **X** controla o movimento para os lados (direita e esquerda).

O eixo **Y** controla o movimento para cima e para baixo.

Quando a barata “aponta para uma direção”, ela está se orientando com base nesses eixos — é como se ela seguisse uma **bússola virtual** dentro do jogo!

O bloco **“mova 10 passos”** indica **quantos passos** a barata deve andar nessa direção.
Se você quiser deixá-la **mais rápida**, aumente o número de passos.
Se quiser que ela se mova **mais devagar**, diminua o número.

Dica extra

Se ao se mover a barata **ficar de cabeça para baixo**, isso é fácil de resolver!
Basta adicionar o bloco:

“definir estilo de rotação para esquerda-direita”

Com esse comando, a barata vai apenas virar para os lados, sem ficar de ponta-cabeça.

Curiosidade educativa

Sabia que, ao programar esse movimento, você está aprendendo **conceitos de geometria e lógica de programação** ao mesmo tempo?
Quando escolhe direções como 0°, 90°, 180° e -90°, você está aplicando a ideia de **ângulos e orientação no plano cartesiano** — tudo de forma prática e divertida!

A Programação da Maçã — Desaparecendo ao Tocar na Barata

Agora que a barata está se mexendo, chegou a hora de **criar o objetivo do jogo!**
Vamos adicionar uma **maçã** que o jogador precisa coletar.
Quando a barata encostar nela, a maçã **vai desaparecer** e tocar um som de "ponto ganho". ✨

Passo 1: adicionando o ator "Maçã"

Clique no **ícone de escolher um ator** (aquele com uma carinha de gato no canto inferior direito).

Vá até a **biblioteca de atores** e procure por **"Apple"** ou **"Maçã"**.

Clique nela e depois em **OK**.

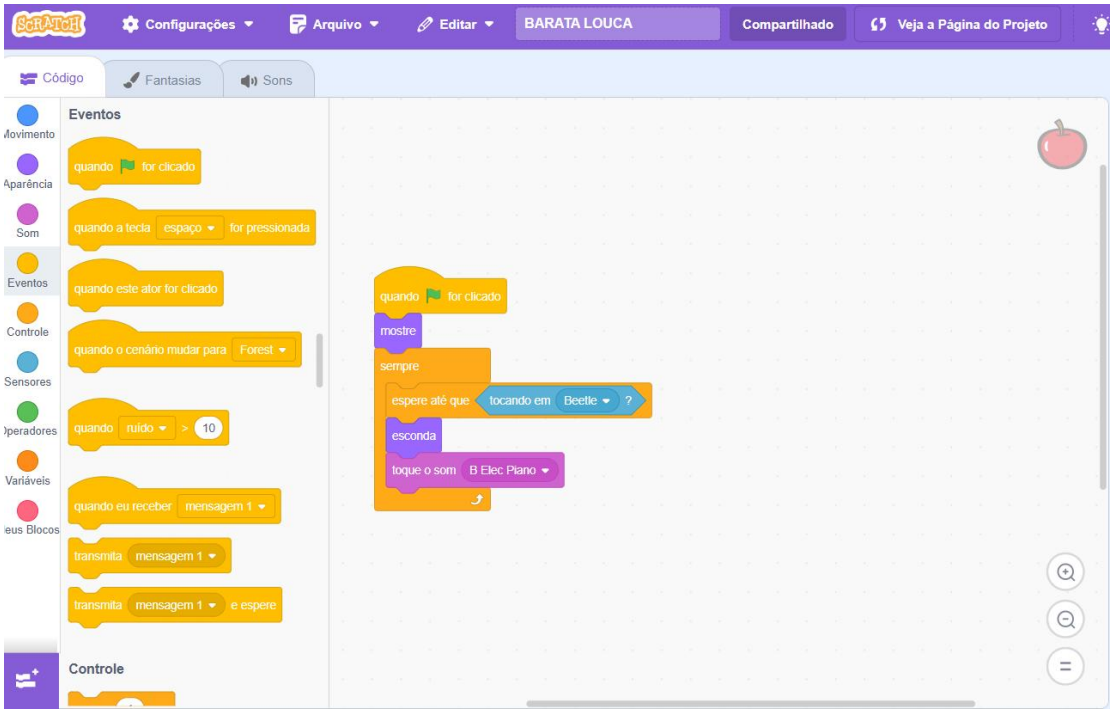
Pronto! Agora a maçã vai aparecer no palco ao lado da barata.

Dica: você pode redimensionar a maçã clicando nela e ajustando o tamanho na opção "Tamanho (%)" logo abaixo do palco. Assim, ela fica mais proporcional ao seu jogo.

Passo 2: criando a programação da maçã

Agora vamos fazer com que a maçã **desapareça** sempre que a barata encostar nela.

Use os seguintes blocos (como mostra a imagem):



"Código da maçã: ela desaparece quando a barata encosta e toca um som de ponto ganho."

🧠 Entendendo o que cada bloco faz:

Bloco	Função
🟧 **"quando a bandeira verde for clicada"**	É o comando que **inicia o código** quando o jogador começa o jogo. Tudo que estiver abaixo dele só será executado após clicar na bandeira verde.
🟣 **"mostre"**	Faz a maçã **aparecer na tela** no início do jogo. Isso é importante, pois ela pode estar escondida da rodada anterior.
🟥 **"sempre"**	Cria um **loop infinito**, ou seja, o código dentro dele vai rodar o tempo todo durante o jogo.
🔵 **"espere até que tocando em Beetle"**	Esse bloco faz o Scratch **esperar até que a maçã encoste na barata** (no caso, o ator chamado "Beetle"). Assim que elas se tocam, o programa executa os comandos seguintes.
🟣 **"esconda"**	Faz a maçã **sumir do cenário** — como se tivesse sido coletada!
🟣 **"toque o som (B Elec Piano)"**	Toca um som curto, marcando que o jogador **ganhou um ponto** ou **pegou a maçã**. Você pode escolher outro som clicando na seta ao lado do nome.

Dica do programador

Você pode fazer a maçã **reaparecer** em outro lugar depois que for coletada.
Para isso, adicione dentro do loop o bloco:
"vá para posição aleatória"
logo depois de "esconda" e **antes de "mostre"** novamente.

Assim, cada vez que a barata encostar nela, a maçã **desaparece, toca o som e surge em outro ponto do cenário** — deixando o jogo ainda mais divertido!

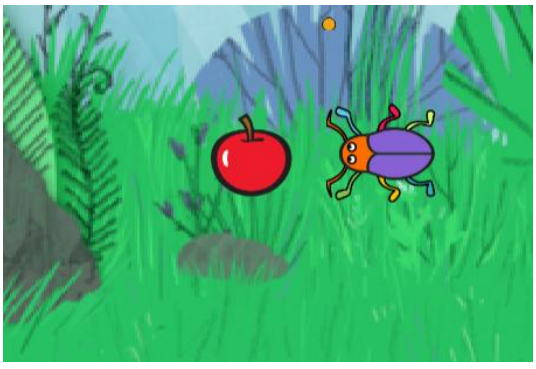
Curiosidade educativa

Neste código, você usou um **sensor**, que é um tipo de bloco que "detecta" o que acontece no jogo — como se a maçã tivesse **olhos e ouvidos**!
Na programação, esse tipo de lógica é chamada de **condição**, pois o comando só acontece **quando uma condição é verdadeira** (neste caso, *"tocando em Beetle"*).

Isso ensina o conceito de **causa e efeito**, uma ideia fundamental na programação e na matemática!

✅ Conclusão da etapa

Agora o jogo está ganhando forma!
Você já tem:
✅ Uma barata que se move;
✅ Uma maçã que desaparece quando tocada;
✅ Um som que marca o ponto conquistado.



Criando o Sistema de Pontos

Chegou o momento de dar mais emoção ao jogo!
Agora, sempre que a barata encostar em uma maçã, **o jogador vai ganhar um ponto**.
Vamos aprender como fazer isso usando as **variáveis**, que são como "caixinhas" onde o Scratch guarda informações durante o jogo.

O que é uma variável?

Imagine uma **caixa com uma etiqueta** escrita "PONTOS".
Toda vez que a barata pega uma maçã, colocamos **+1** dentro dessa caixa.
No final, podemos abrir e ver quantas maçãs foram coletadas! ✨

Essa é a função de uma variável: **guardar valores que mudam** durante o jogo.

Passo 1: criando a variável "PONTOS"

Vá até o grupo de blocos **"Variáveis"** (na cor laranja).

Clique em **"Criar uma variável"**.

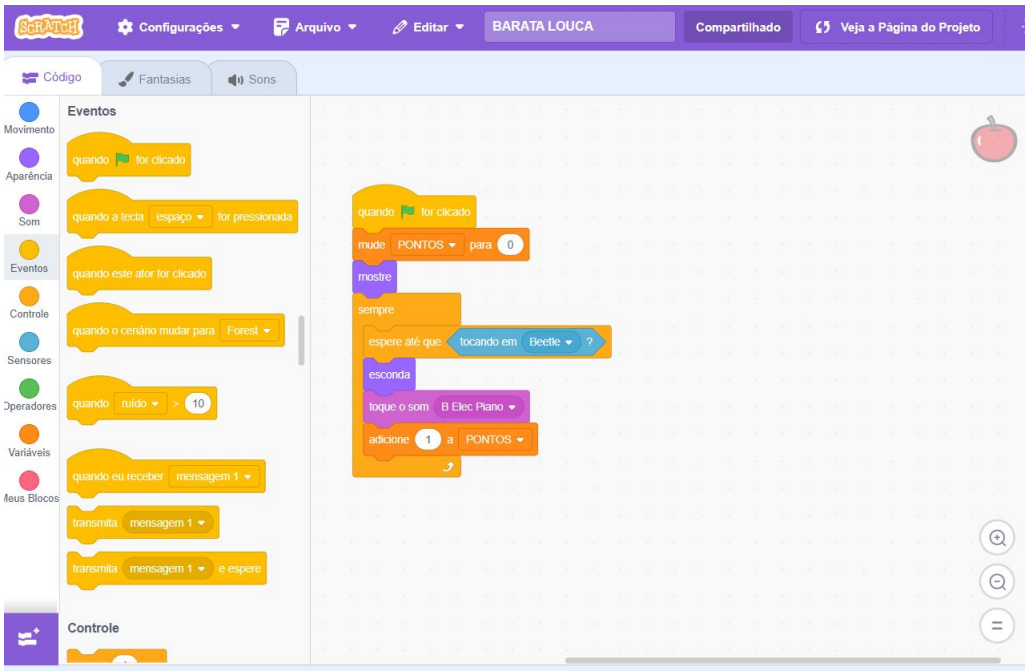
Na janela que abrir, escreva o nome **PONTOS** e clique em **OK**.

Agora, no canto da tela, você verá um número aparecendo — é o **contador de pontos** do jogo!

Passo 2: programando a contagem de pontos

Com a variável criada, vamos ajustar a programação da maçã para que ela adicione 1 ponto sempre que for tocada pela barata.

Seu código deve ficar assim:



🧠 Entendendo cada bloco:

Bloco	Função
🟧 "quando a bandeira verde for clicada"	Inicia o código quando o jogo começa.
🟥 "mude PONTOS para 0"	Zera a pontuação toda vez que o jogo é iniciado — assim, o placar começa do zero.
🟣 "mostre"	Faz a maçã aparecer no início do jogo.
🟥 "sempre"	Cria um loop que repete o código durante todo o jogo.
🔵 "espere até que tocando em Beetle"	Espera até a barata encostar na maçã.
🟣 "esconda"	Faz a maçã desaparecer, como se tivesse sido coletada.
🟣 "toque o som (B Elec Piano)"	Toca um som para indicar que o jogador ganhou um ponto.
🟥 "adicione 1 a PONTOS"	Aumenta a pontuação em 1 toda vez que a barata pega uma maçã.

Dica do programador

Se quiser que a maçã **reapareça** em outro ponto após ser coletada, adicione o bloco:
"vá para posição aleatória" logo depois de "adicione 1 a PONTOS" e **antes de "mostre"** novamente.

Assim, o jogo nunca para — a barata pega uma maçã, ganha um ponto, e outra aparece em outro lugar do cenário! ✨

Curiosidade educativa

Ao usar variáveis, você está aprendendo conceitos de **matemática e lógica computacional**.
Na prática, o Scratch está **somando 1 ao valor anterior da variável PONTOS** toda vez que a condição é verdadeira.
Ou seja, você está aplicando **adição e controle de variáveis**, dois fundamentos importantes na programação e na matemática!

✅ Conclusão da etapa

Agora sim, o seu jogo **"Barata Louca"** está ganhando forma de verdade:
✅ A barata se move;
✅ A maçã desaparece quando é tocada;
✅ O jogador ganha pontos por cada maçã coletada!

Programando o Morcego

Agora que já temos a **barata** e a **maçã** funcionando, é hora de dar um toque de desafio ao jogo: vamos adicionar o **morcego**, um personagem que se move de forma imprevisível e pode acabar com a diversão da barata!

1. Adicionando o ator "Morcego"

Clique no ícone de **"Escolher um ator"** (o símbolo de gato com um "+" embaixo da tela).

Procure por **“Bat” (Morcego)** na biblioteca do Scratch e clique em **OK**.

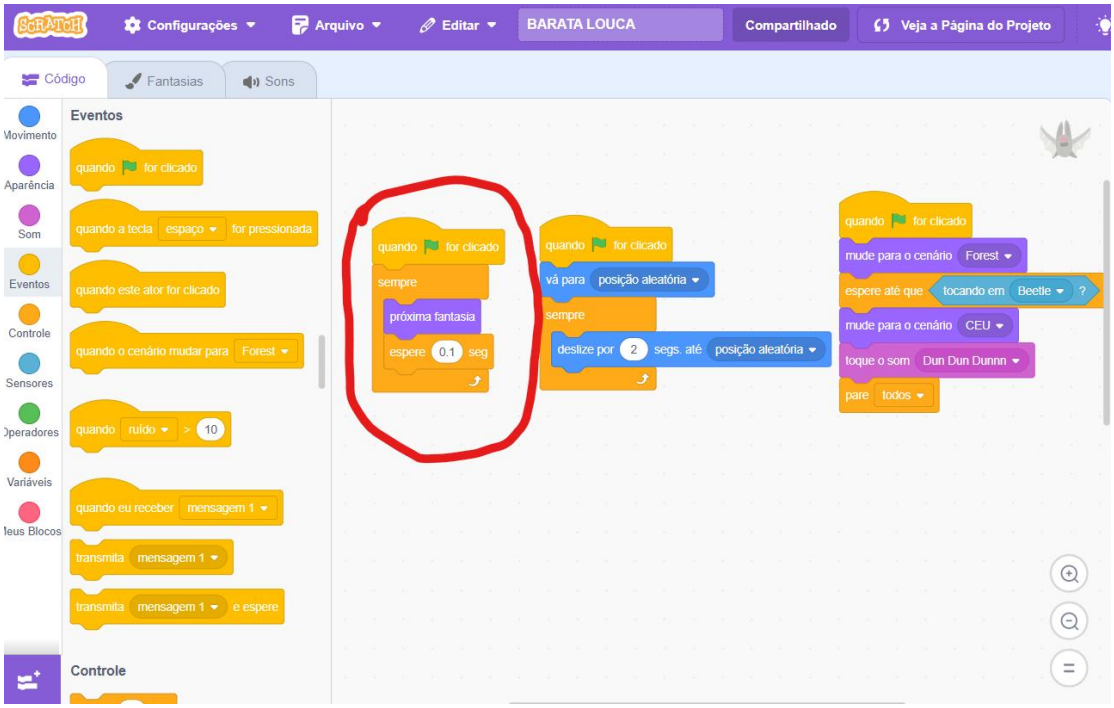
Agora o morcego aparecerá no palco, pronto para ser programado!

Dica: você pode ajustar o tamanho do morcego clicando no campo “Tamanho” abaixo do palco e colocando, por exemplo, **50**. Assim ele fica mais proporcional à barata.

2. Fazendo o morcego se mover e bater as asas

O morcego precisa parecer que está voando, então vamos fazer com que ele troque de fantasia rapidamente (as fantasias do morcego mostram as asas abertas e fechadas).

Blocos usados:

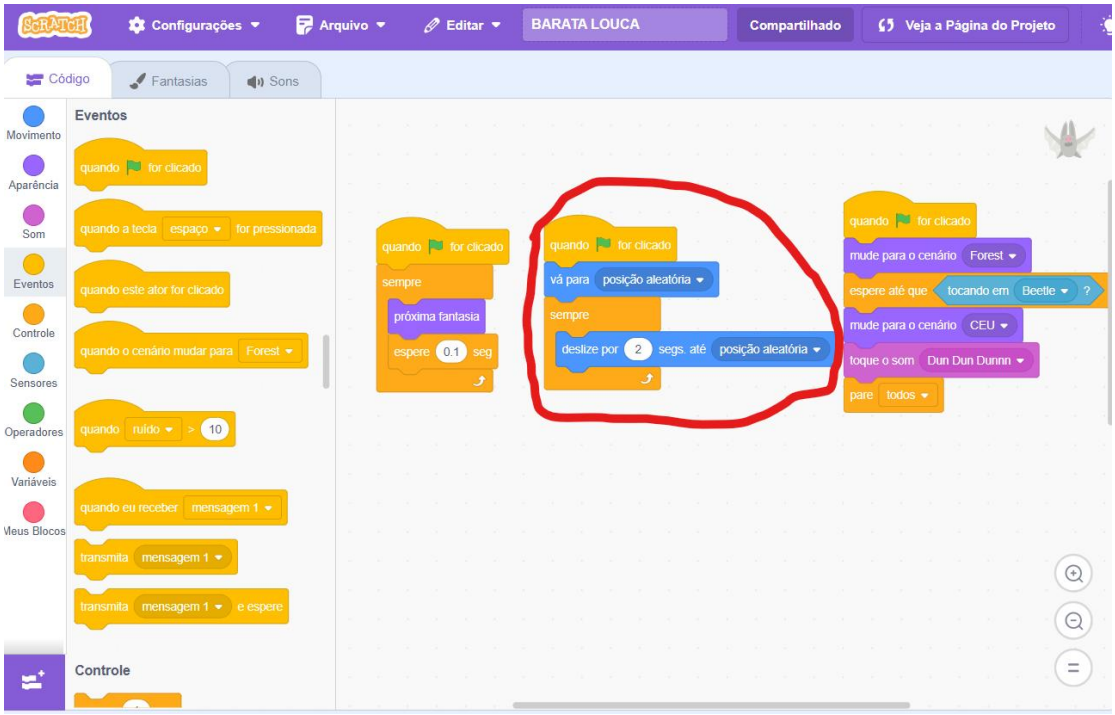


O que esse código faz:
Quando o jogo começar, o morcego vai trocar de fantasia sem parar, dando a impressão de que ele está batendo as asas o tempo todo.

3. Fazendo o morcego voar aleatoriamente

Agora, queremos que o morcego se mova de forma imprevisível pelo cenário.

Blocos usados:

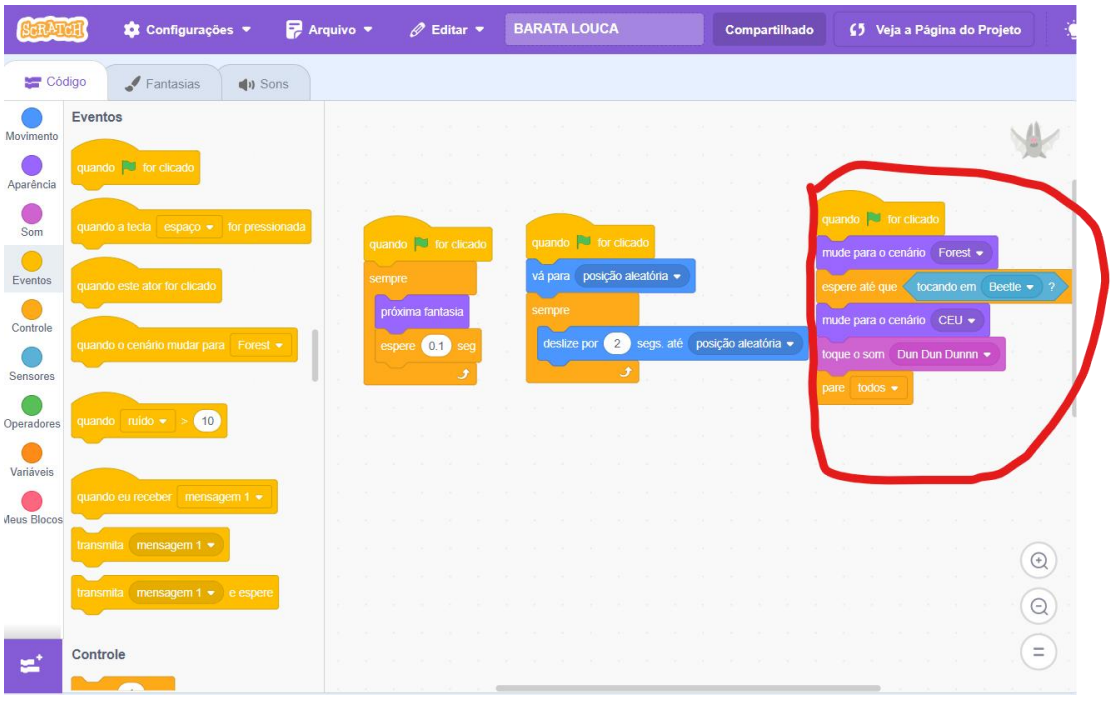


O que esse código faz:
O morcego começa o jogo em um ponto aleatório e, continuamente, desliza para diferentes partes da tela, como se estivesse voando por todo o ambiente.

4. O morcego “pega” a barata

Agora vem a parte mais divertida (e perigosa). Vamos programar o que acontece quando o morcego encostar na barata.

Blocos usados:



O que esse código faz:

O cenário começa como “Floresta” (Forest).

O jogo fica esperando até o morcego encostar na barata.

Quando isso acontece, o cenário muda para o “Céu” (Ceu), indicando o fim do jogo.

Um som dramático é tocado, e tudo para — o jogo termina!

Resultado Final

Agora, o jogo “Barata Louca” está ainda mais empolgante!

A **barata** corre aleatoriamente pelo cenário,

As **maçãs** somem quando encostadas e somam pontos,

E o **morcego** voa por todo lado tentando pegar a barata!

Programando o morcego — adicionando a vida e o dano da laranja

Agora que a barata já se movimenta e a maçã dá pontos, vamos adicionar um **desafio extra**: um morcego que perde vida sempre que é atingido pela laranja lançada pela barata!

Passo 1: Adicionando o ator Morcego

Clique no ícone **“Escolher um ator”** (aquele rostinho no canto inferior direito).

Procure o ator **“Bat” (morcego)** na biblioteca e clique nele.

Pronto! O morcego foi adicionado à sua área de trabalho.

Esse será o inimigo da barata!

Passo 2: Criando a variável de vida

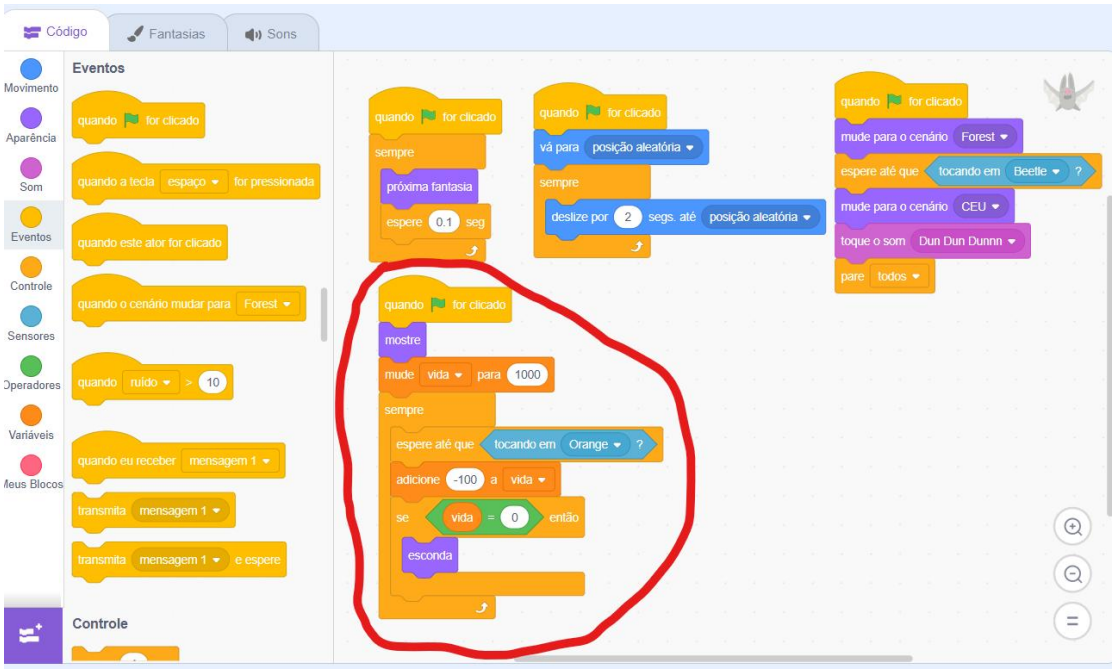
Antes de programar, precisamos criar uma **variável** para guardar o número de vidas do morcego.

Vá na categoria **“Variáveis”** (na cor laranja).

Clique em **“Criar uma variável”** e digite o nome **vida**.

Clique em **OK**.

Essa variável vai controlar a quantidade de energia que o morcego ainda tem.



⚙️ Passo 3: Programando o morcego

Agora, observe os blocos destacados na imagem. Vamos entender o que cada um faz:

🟧 "Quando bandeira verde for clicada"

Esse bloco inicia a programação do morcego assim que o jogo começa.

🟥 "Mostre"

Garante que o morcego apareça na tela no início do jogo.

🟥 "Mude [vida] para 1000"

Aqui definimos que o morcego começa com **1000 pontos de vida**.

Você pode mudar esse número se quiser deixá-lo mais forte ou mais fraco! 💪

🔁 "Sempre"

Cria um laço (repetição infinita), fazendo o Scratch verificar o tempo todo se o morcego foi atingido por uma laranja.

🔵 "Espere até que tocando em [Orange]"

Esse bloco de **Sensores** detecta o momento em que o morcego é atingido por uma **laranja**.

🟥 "Adicione -100 à [vida]"

Toda vez que a laranja toca o morcego, **100 pontos são subtraídos da vida dele**.

🟩 "Se [vida = 0] então"

Aqui entra uma condição: se a vida chegar a **zero**, o morcego será derrotado.

🟣 "Esconda"

Esse comando faz o morcego desaparecer da tela quando sua vida acabar. 💥

Programando a Laranja — o ataque da Barata

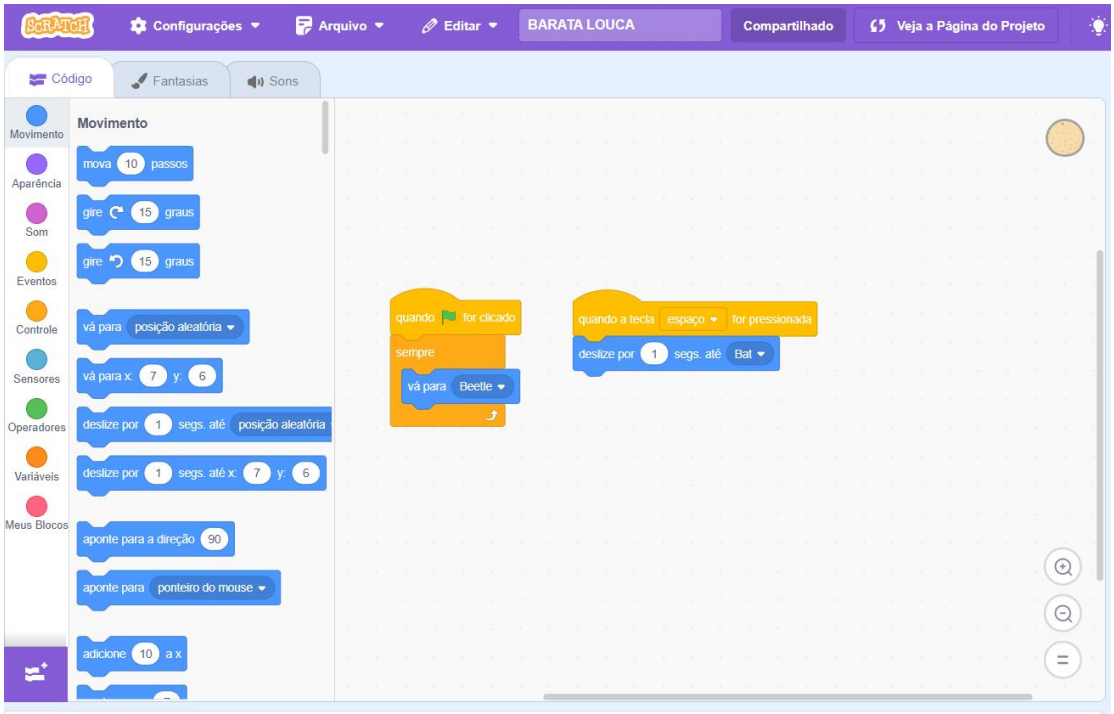
Agora que o morcego já tem **vida**, está na hora de dar uma arma para a nossa heroína! ⚡
A laranja será o **projétil** que a barata atira para tentar derrotar o morcego.

Passo 1: Adicionando o ator Laranja

Clique em **"Escolher um ator"** (ícone do rostinho no canto inferior direito).

Na biblioteca, procure **"Orange" (laranja)** e clique nela.

Pronto! Agora temos a arma da barata!



Passo 2: Entendendo os blocos da programação

"Quando bandeira verde for clicada"

Esse comando inicia o comportamento da laranja quando o jogo começa.

"Sempre" + "Vá para Beetle"

Esses blocos fazem com que a laranja **siga a barata** o tempo todo, ficando "guardada" até o momento do disparo.
Assim, onde a barata for, a laranja vai junto — como se estivesse preparada para o ataque!

"Quando a tecla espaço for pressionada"

Esse é o momento do ataque!
Toda vez que o jogador aperta a tecla **espaço**, a laranja é lançada em direção ao morcego.

“Deslize por 1 seg. até Bat”

Esse bloco faz a laranja **viajar pelo ar**, indo até a posição onde o morcego está.

“1 segundo” é o tempo que ela demora para chegar — você pode ajustar para deixar o disparo mais rápido ou mais lento.

0 que aprendemos?

A laranja é o **projetil do jogador**.

Ela **segue a barata** até o momento do ataque.

Ao apertar **espaço**, ela se move em direção ao **morcego**, tirando a vida dele quando o acerta.

Chegamos ao fim da criação do nosso primeiro jogo no **Scratch**!

Neste projeto, aprendemos passo a passo como **programar a movimentação de um personagem**, **criar interações entre atores**, **contar pontos**, **controlar a vida de um inimigo** e até **definir o fim do jogo**.

Com blocos coloridos e comandos simples, vimos que **programar pode ser divertido e acessível para todos** — basta explorar, testar e deixar a imaginação voar!

Agora que você dominou o jogo **“Barata Louca”**, que tal criar o seu próprio jogo?
Mude os personagens, os sons, os cenários e invente novas regras.
No Scratch, **você é o criador** — e o limite é a sua criatividade!

- Sumô

Bem-vindo ao jogo **Sumô**, uma aventura divertida e desafiadora criada no Scratch!

Neste jogo, você vai conhecer **Tatiana**, uma pequena e corajosa lutadora que adora comida!
O objetivo dela é **coletar o máximo de alimentos possível** — como **banana, maçã, bolo e taco** — para ganhar pontos e crescer cada vez mais.

Mas cuidado!
Nem tudo é tão simples quanto parece. Enquanto Tatiana tenta ficar forte e gigantesca, **animais perigosos** como a **cobra (Snake)**, o **ouriço (Hedgehog)** e o **tubarão (Shark)** estão à solta pelo cenário, prontos para atrapalhar sua missão.

Cada vez que Tatiana encosta em um desses inimigos, ela **perde um ponto de vida**.
Ela começa com **3 vidas**, então é importante **desviar bem rápido** para continuar jogando!

Além disso, existe outro personagem misterioso: o **Frank**.
Ele é como um tipo de "Ogro" que aparece conforme Tatiana ganha mais pontos.
Mas cuidado! Ao encostar nele, **Tatiana não perde vida**, mas é **empurrada para longe**, o que deixa o jogo ainda mais desafiador!

O grande objetivo é **pegar o maior número possível de comidas**, **ficar cada vez maior** e **sobreviver o máximo de tempo** sem perder todas as vidas!

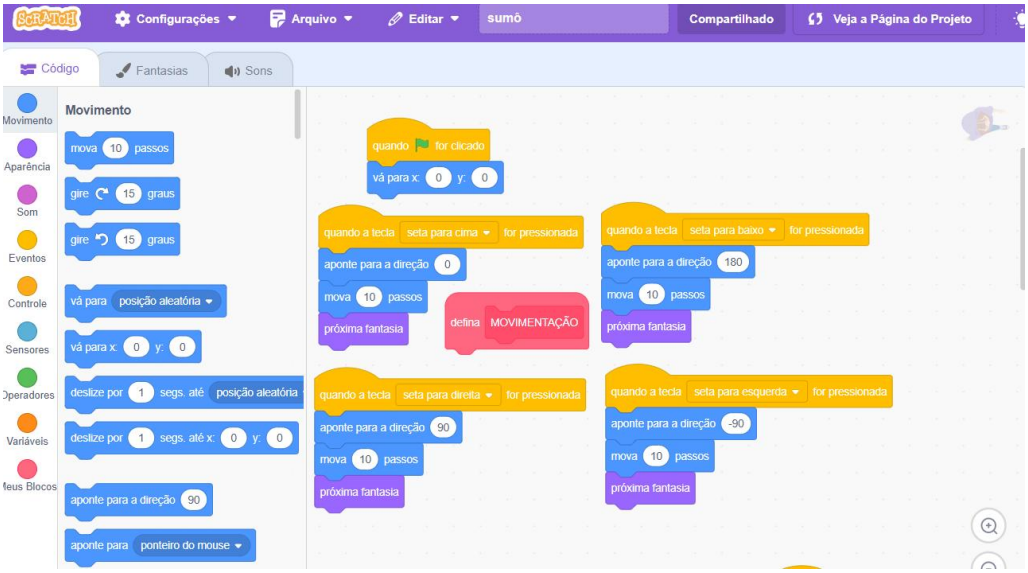
✨ O que você vai aprender com este jogo

Neste capítulo, você vai aprender:
✅ Como **criar um personagem que cresce** ao ganhar pontos.
✅ Como **programar inimigos** que causam dano ao jogador.
✅ Como **controlar vidas e pontuação**.
✅ Como **criar desafios dinâmicos**, como o empurrão do Frank.

Programação da Tatiana — Movimentação

Para começar, vamos dar vida à nossa protagonista, **Tatiana**!
Ela precisa se mover pelo cenário para coletar os alimentos e desviar dos inimigos.
Para isso, vamos programar suas **movimentações** usando as setas do teclado .

Blocos utilizados:



Eventos → “quando a tecla (seta) for pressionada”

Movimento → “aponte para a direção” e “mova 10 passos”

Aparência → “próxima fantasia”

Meus Blocos → “defina MOVIMENTAÇÃO”

Como funciona:

Quando o jogo começa (“quando a bandeira for clicada”), a Tatiana vai para a posição inicial — coordenadas **x: 0, y: 0**.

Cada vez que o jogador pressiona uma **seta do teclado**, ela se move 10 passos na direção correspondente:

Seta para cima: direção **0°**

Seta para baixo: direção **180°**

Seta para a direita: direção **90°**

Seta para a esquerda: direção **-90°**

A cada movimento, o bloco “próxima fantasia” faz a Tatiana **mudar de aparência**, criando um efeito de **animação de caminhada**.

✨ **Resultado:**
Com isso, a Tatiana já pode andar livremente pelo cenário!
Ela está pronta para começar sua jornada e coletar os alimentos que vão deixá-la cada vez maior!

Programando os alimentos — Pontos e crescimento da Tatiana

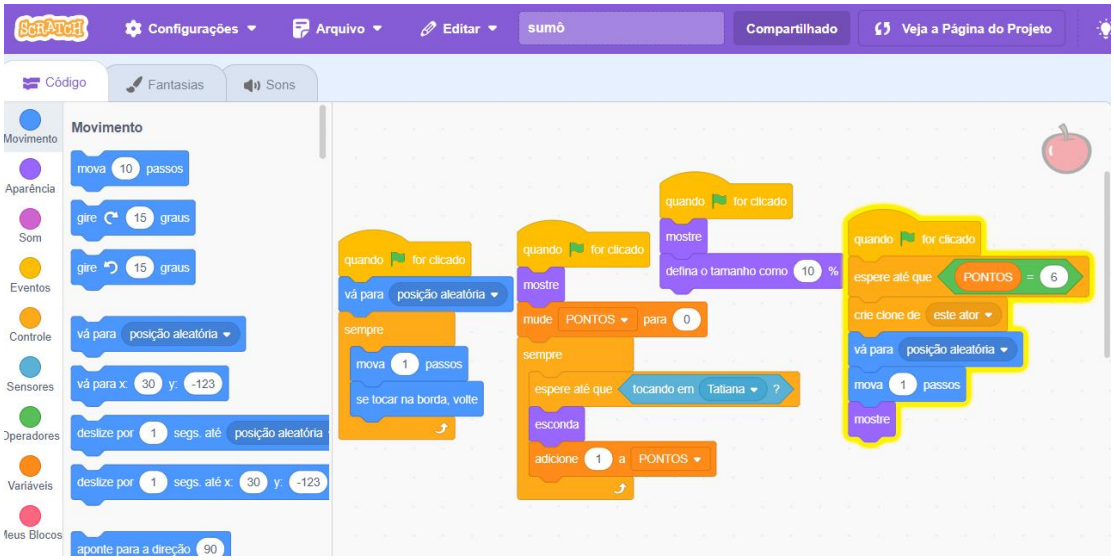
Agora que a personagem **Tatiana** já pode se movimentar pelo cenário, chegou o momento de dar a ela um objetivo: **coletar alimentos e ganhar pontos!**
Os alimentos são os atores **banana, maçã, taco, bolo**, entre outros. Vamos usar a **maçã** como exemplo, mas a mesma programação vale para os demais.

Passo 1: Criando o ator Maçã

Clique no ícone de **“Escolher um ator”** (embaixo à direita).

Escolha a **Maçã** na biblioteca de personagens do Scratch.

Posicione-a em qualquer lugar do cenário.



Passo 2: Fazendo a maçã aparecer e se mover

No código da maçã, adicione os seguintes blocos:

“Quando bandeira verde for clicada” → serve para iniciar o código junto com o jogo.

“Vá para posição aleatória” → faz com que a maçã apareça em diferentes lugares toda vez que o jogo começa.

Bloco “sempre” → cria um laço que não para de repetir.

Dentro do “sempre”, adicione:

“mova 1 passo” → faz a maçã deslizar devagar.

“se tocar na borda, volte” → impede que ela saia do cenário.

Com isso, a maçã se movimenta suavemente, parecendo “viva” dentro do jogo.

Passo 3: Criando a pontuação

Agora precisamos fazer a Tatiana **ganhar pontos** ao encostar na maçã.

No mesmo ator (Maçã), adicione:

“quando bandeira verde for clicada” → para iniciar a contagem toda vez que o jogo começar.

“mostre” → faz a maçã aparecer na tela.

“mude PONTOS para 0” → define que a pontuação começa em zero.

Bloco “sempre” → repete continuamente a verificação.

Dentro dele, coloque o bloco **“espere até que tocando em Tatiana”**.

Depois, adicione **“esconda”** (para sumir quando for coletada).

E, por fim, **“adicione 1 a PONTOS”** — isso faz a pontuação aumentar a cada coleta.

✨ Dica: faça isso com os outros alimentos (banana, taco, bolo etc.), mudando apenas o nome do ator.

Passo 4: Reaparecimento dos alimentos

Para deixar o jogo mais desafiador, os alimentos reaparecem depois de um tempo ou quando o jogador atinge certa pontuação.

Na imagem, podemos ver o bloco:

“espere até que PONTOS = 6”

“crie clone de (este ator)”

Isso significa que, ao alcançar **6 pontos**, o Scratch cria **outra maçã no cenário**, gerando mais movimento e aumentando a diversão!

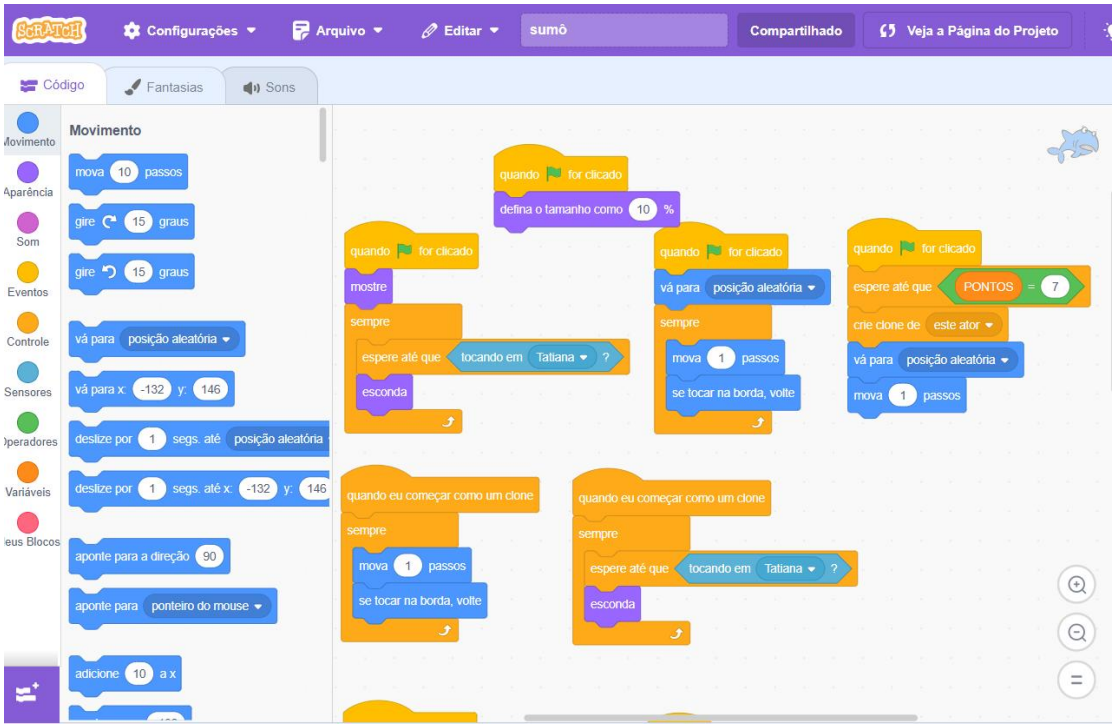
Passo 5: O crescimento da Tatiana

A cada ponto ganho, a Tatiana cresce um pouquinho.
Essa parte é feita com a variável **tamanho** da personagem principal — e explicaremos com detalhes quando formos para o código dela relacionado ao crescimento.
Mas, em resumo, **quanto mais alimentos ela pega, maior ela fica!**

Clonagem e aumento da dificuldade

Conforme a Tatiana vai coletando os alimentos, o jogo fica mais emocionante!
Isso acontece graças a um recurso muito poderoso do Scratch: os **clones**. Eles permitem que novos alimentos apareçam automaticamente, sem que o jogador precise reiniciar o jogo.

Veja como isso funciona:



Passo 1: Criando clones automaticamente

No código da maçã (ou outro alimento), há um bloco que diz:

“espere até que PONTOS = 7”
“crie clone de (este ator)”

Esses blocos fazem com que, ao atingir **7 pontos**, o Scratch crie **um novo clone** do alimento. Assim, quanto mais pontos a Tatiana ganha, **mais frutas aparecem**, aumentando o desafio.

Passo 2: Fazendo os clones se moverem

Cada clone também precisa se comportar como o alimento original.
Por isso, usamos o evento:

“quando eu começar como um clone”

Dentro desse bloco, há:

“mova 1 passo”

“se tocar na borda, volte”

Esses comandos garantem que o clone se mova e **permaneça ativo** no cenário, sem sair da tela.

Passo 3: Fazendo o clone desaparecer ao ser coletado

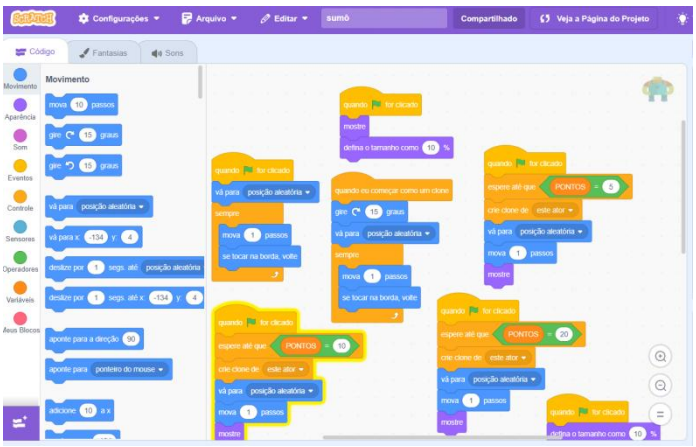
Ainda dentro do mesmo evento (“quando eu começar como um clone”), colocamos:

“espere até que tocando em Tatiana”
“esconda”

Assim, sempre que a Tatiana encosta no clone, ele **desaparece** e pode dar **pontos extras** (ou outro efeito, dependendo da programação).

Aumento progressivo da dificuldade

O jogo **Sumô** foi planejado para não ficar fácil demais.
Conforme a Tatiana coleta alimentos e ganha pontos, o Scratch cria **novos desafios automáticos**: mais alimentos começam a se mover pelo cenário, exigindo mais habilidade e rapidez do jogador.



Na programação, isso acontece por meio de blocos como:

“espere até que PONTOS = 5”
“espere até que PONTOS = 10”
“espere até que PONTOS = 20”

Cada vez que uma dessas condições é atingida, o Scratch **cria clones de alimentos**.
Esses novos clones aparecem em **posições aleatórias**, movendo-se continuamente e **aumentando o nível de dificuldade** da partida.

Dessa forma:

No início, há poucos alimentos, e o jogador pode se acostumar aos comandos.

Depois dos **5 pontos**, surgem mais clones, deixando o jogo um pouco mais agitado.

Ao chegar aos **10 e 20 pontos**, a tela fica cheia de movimento, o que exige mais atenção e reflexos rápidos. ⚡

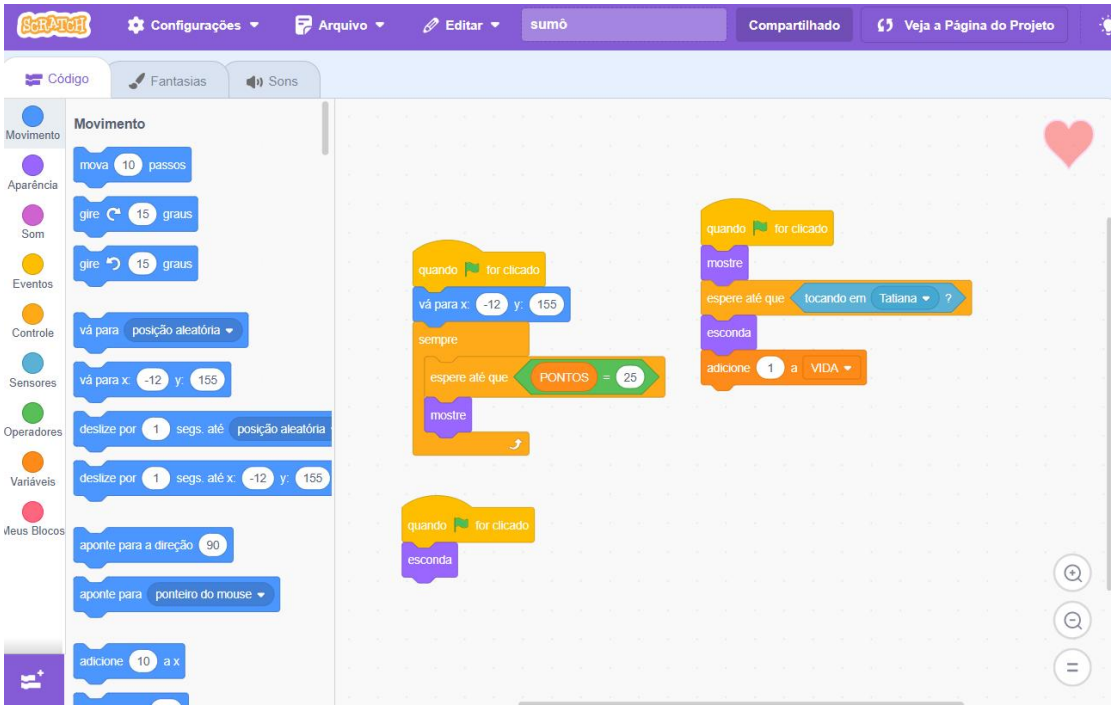
Além disso, todos os clones usam o comando:

“mova 1 passo”
“se tocar na borda, volte”

Isso garante que cada alimento — seja clone ou não — continue circulando no cenário sem sair dele.

A vida extra: o coração que salva o jogo

Para deixar o jogo *Sumô* ainda mais interessante, foi adicionada uma **vida extra** em forma de **coração**.
Esse coração aparece **automaticamente** quando a Tatiana atinge **25 pontos**, funcionando como uma espécie de “bônus” pela boa performance do jogador.



No Scratch, isso é feito com o comando:

“espere até que PONTOS = 25”

Quando essa condição é verdadeira, o coração aparece na tela (com o bloco **“mostre”**) e fica visível até ser coletado.

Ao tocar na Tatiana, o coração:

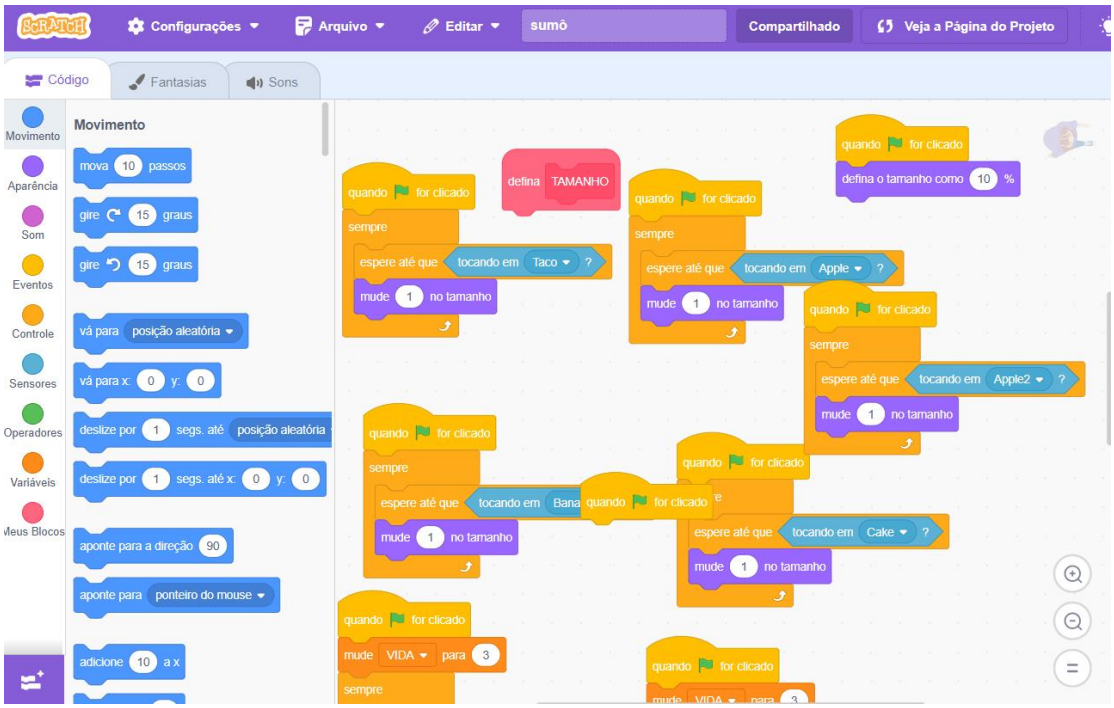
Se esconde novamente, usando o bloco **“esconda”**;

Adiciona 1 ponto à variável VIDA, com o comando **“adicione 1 a VIDA”**.

Essa mecânica traz um elemento estratégico ao jogo:
o jogador precisa sobreviver tempo suficiente e capturar pontos para que o coração apareça, o que incentiva a **persistência e o cuidado** durante as rodadas.

O crescimento da Tatiana

No jogo *Sumô*, a personagem principal — **Tatiana** — começa bem pequena, com apenas **10% do tamanho original**. Isso é feito através do bloco:



“defina o tamanho como 10%”

A cada alimento que Tatiana coleta (como **banana**, **maçã**, **taco**, **bolo** e outros), o jogo verifica o toque entre os personagens. Essa verificação é feita com o bloco:

“espere até que tocando em [nome do alimento]”

Assim que o toque é detectado, o programa usa o comando:

“mude 1 no tamanho”

Isso faz com que Tatiana cresça gradualmente a cada alimento consumido. Dessa forma, o jogo simula o aumento de massa da personagem — quanto mais pontos ela faz, **maior ela fica**, tornando o jogo mais desafiador e divertido.

Além disso, no início do jogo, a variável **VIDA** é configurada com o valor **3**, representando as três chances que a Tatiana tem antes de perder o jogo:

“mude VIDA para 3”

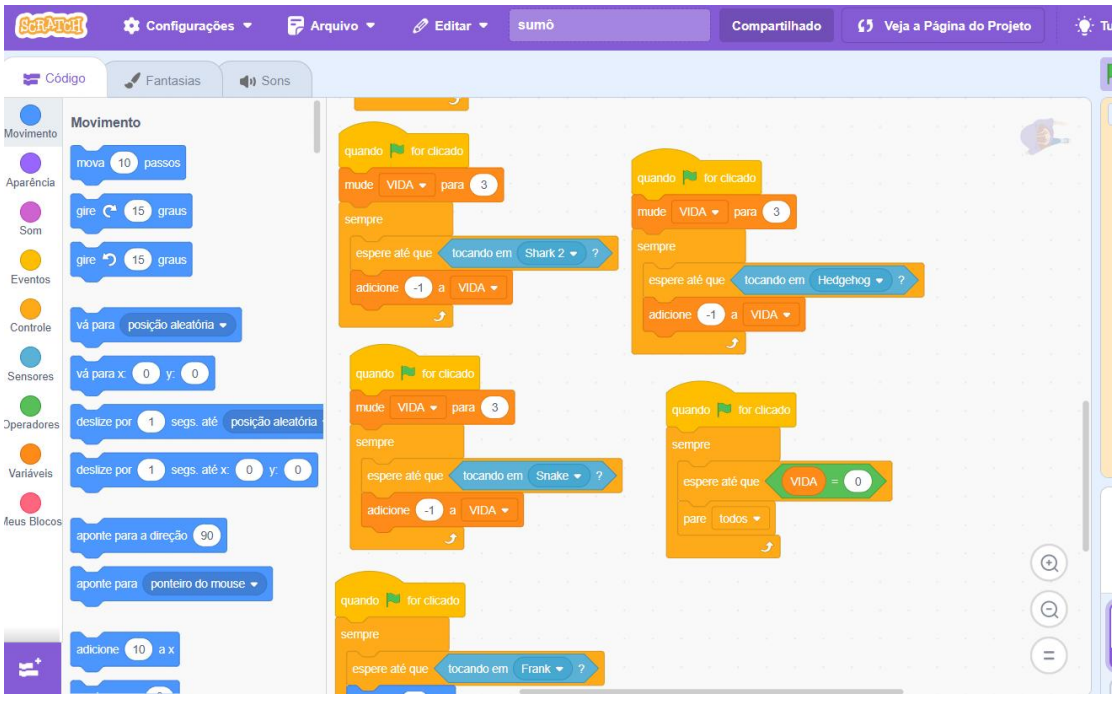
Esse sistema equilibra o desafio do jogo, pois enquanto Tatiana aumenta de tamanho e ganha mais pontos, ela também fica **mais exposta aos inimigos** (cobra, tubarão e ouriço), precisando ser mais ágil para desviar deles.

Sistema de Dano e Vidas

No jogo *Sumô*, a personagem **Tatiana** precisa evitar diversos inimigos — como o **tubarão (Shark 2)**, a **cobra (Snake)**, o **ouriço (Hedgehog)** e o personagem **Frank**. Cada vez que Tatiana encosta em um desses inimigos, ela **perde uma vida**.

Esse sistema é implementado através da variável **VIDA**, que representa a quantidade de chances que a personagem tem.

Logo no início do jogo, o comando:



“quando a bandeira verde for clicada → mude VIDA para 3”

define que Tatiana começa com **três vidas**.

Em seguida, há blocos de repetição que verificam constantemente se ela está tocando em algum inimigo:

> “espere até que tocando em [nome do inimigo]”
> “adicione -1 a VIDA”

Esses blocos aparecem repetidos para cada inimigo (Shark 2, Snake, Hedgehog e Frank). Assim, sempre que ocorre uma colisão, **1 é subtraído da variável VIDA**, simulando o dano sofrido.

O jogo também contém um sistema de **fim de partida automático**. Um dos blocos monitora continuamente o valor da variável VIDA e, quando ela chega a zero, o comando:

> “pare todos”

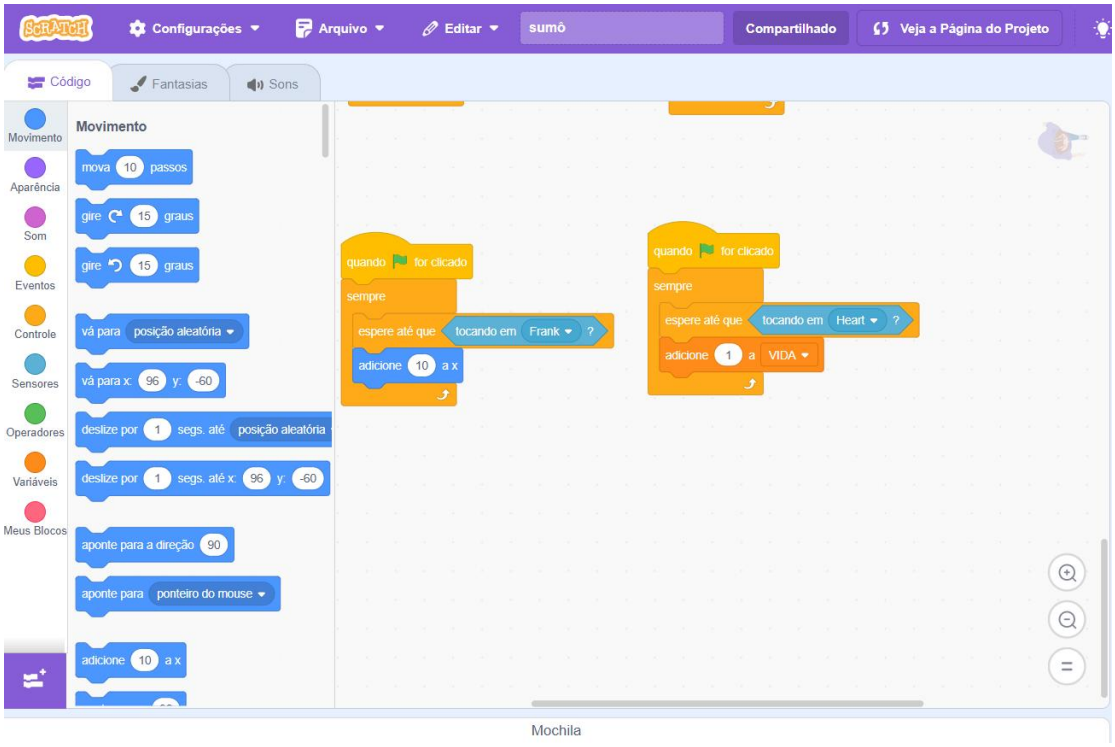
é executado, encerrando imediatamente o jogo.

Esse conjunto de comandos cria um **mecanismo de risco e recompensa** — o jogador precisa se arriscar para coletar alimentos e aumentar de tamanho, mas deve evitar os inimigos para não perder todas as vidas.

Mais adiante, quando Tatiana atingir **25 pontos**, aparecerá um **item especial em forma de coração**. Ao tocá-lo, ela recupera **+1 vida**, fortalecendo ainda mais a jogabilidade dinâmica e equilibrada.

Interações Especiais de Tatiana

Além de coletar alimentos e evitar inimigos, Tatiana também interage com outros personagens que influenciam sua jornada no jogo — como **Frank** e o **Heart (coração)**.



Essas interações foram programadas para ocorrer **automaticamente**, usando a estrutura de repetição **“sempre”**, garantindo que o jogo monitore continuamente se Tatiana está tocando nesses elementos.

Interação com Frank

Quando a bandeira verde é clicada, o jogo entra em um laço de repetição que verifica constantemente se Tatiana está **tocando em Frank**.
Assim que isso acontece, o comando:

```
adicione 10 a x
```

é acionado.

Isso faz com que Tatiana **avance 10 unidades para a direita**, representando um impulso, como se Frank a empurrasse ou ajudasse a seguir em frente. Essa mecânica adiciona **dinamismo e movimento** à jogabilidade, tornando o ambiente mais interativo.

Interação com o Coração (Heart)

Em paralelo, outro bloco verifica se Tatiana **toca no coração**, símbolo da vida extra.
Quando a colisão ocorre, o jogo executa:

```
adicione 1 à variável VIDA
```

Isso significa que **Tatiana ganha uma vida adicional**, o que reforça a sensação de progresso e recompensa por suas ações.

Essa lógica complementa a já existente no ator “Heart” (vista anteriormente), garantindo que a recuperação de vida funcione mesmo se o jogador tocar no coração mais de uma vez durante o ciclo de jogo.

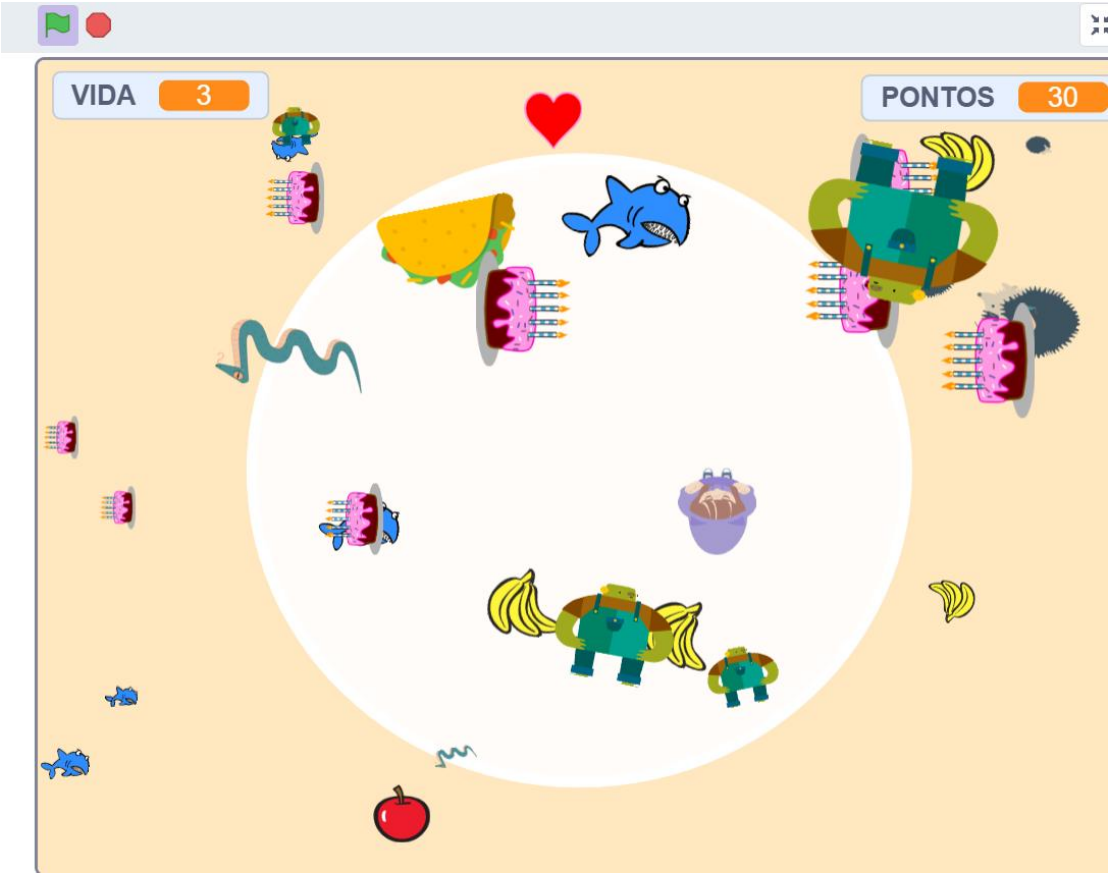
Com isso, a programação de Tatiana fica ainda mais rica — ela não apenas coleta, cresce e enfrenta inimigos, mas **interage com o ambiente de forma viva**, reagindo a estímulos e se fortalecendo.

Programação dos Outros Alimentos e Animais

É importante destacar que, embora tenhamos mostrado detalhadamente a programação de **alguns alimentos e inimigos**, **as demais programações seguem exatamente o mesmo padrão**.
Cada alimento — como o **taco, maçã, banana, bolo**, entre outros — utiliza a **mesma estrutura lógica**: quando Tatiana encosta neles, a variável associada ao tamanho ou aos pontos é alterada, representando o crescimento ou ganho de energia da personagem.

Da mesma forma, os **animais inimigos** — como o **tubarão, a cobra e o ouriço** — também compartilham o mesmo tipo de código, mudando apenas o nome do ator e a ação executada. Essa padronização evita que o projeto fique excessivamente longo e complexo, mantendo a organização e o entendimento do funcionamento geral do jogo.

Com isso, mesmo mostrando apenas uma parte, já é possível compreender **como todas as outras partes se conectam de forma lógica e harmoniosa**, tornando o jogo completo e funcional.



Conclusão

O jogo **Sumô** é um excelente exemplo de como a **lógica de programação** pode ser usada para criar experiências interativas e divertidas.
Através de blocos simples, mas bem pensados, o projeto combina **movimento, colisão, pontuação, vidas extras e crescimento do personagem**, mostrando como o *Scratch* pode ser uma poderosa ferramenta de aprendizado.

A personagem **Tatiana** representa o progresso do jogador: quanto mais alimentos ela coleta, mais forte fica; quanto mais inimigos enfrenta, mais aprende a se proteger.
Esse ciclo de tentativa, erro e recompensa ensina, de forma lúdica, **conceitos fundamentais de programação, matemática e raciocínio lógico**.

Assim, o *Sumô* não é apenas um jogo — é uma **aventura educativa**, criada passo a passo com criatividade, paciência e muita dedicação.
Ele mostra que, quando unimos **imaginação e lógica**, podemos transformar simples blocos coloridos em histórias incríveis e cheias de vida.

Créditos

Desenvolvedores

Barata Louca

João Henrique Lira Silva Veras

Paulo Levi Azevedo Carvalho

Wellyton Herique Pereira da Silva

Leandro da Costa Ribeiro

Sumô

Marcos Isaque Oliveira Pinheiro Amaral

Alfredo José Bezerra Braga

João Henrique Lira Silva Veras

Jefferson dos Santos

Professor: Thayson Gabriel Ferreira Silva