

Aula 3

Primeiros passos com Javascript

Unidade

**Lógica de programação:
desenvolvendo um jogo estilo
Pong**

O que vamos aprender?



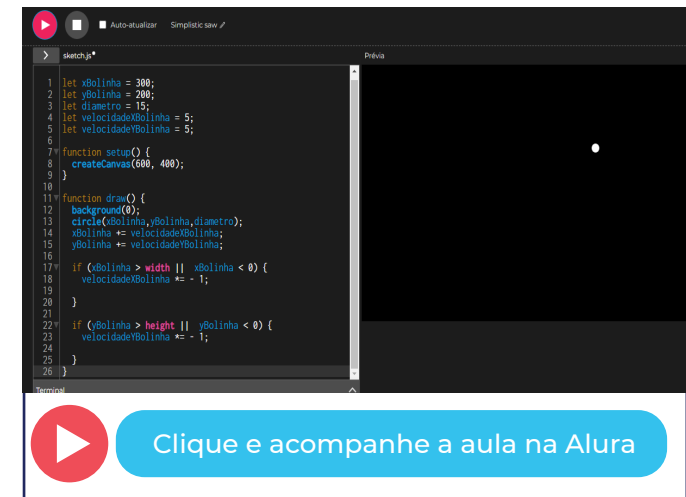
Iniciar a criação do jogo Pong utilizando Javascript.



Criar a bolinha do jogo utilizando funções.



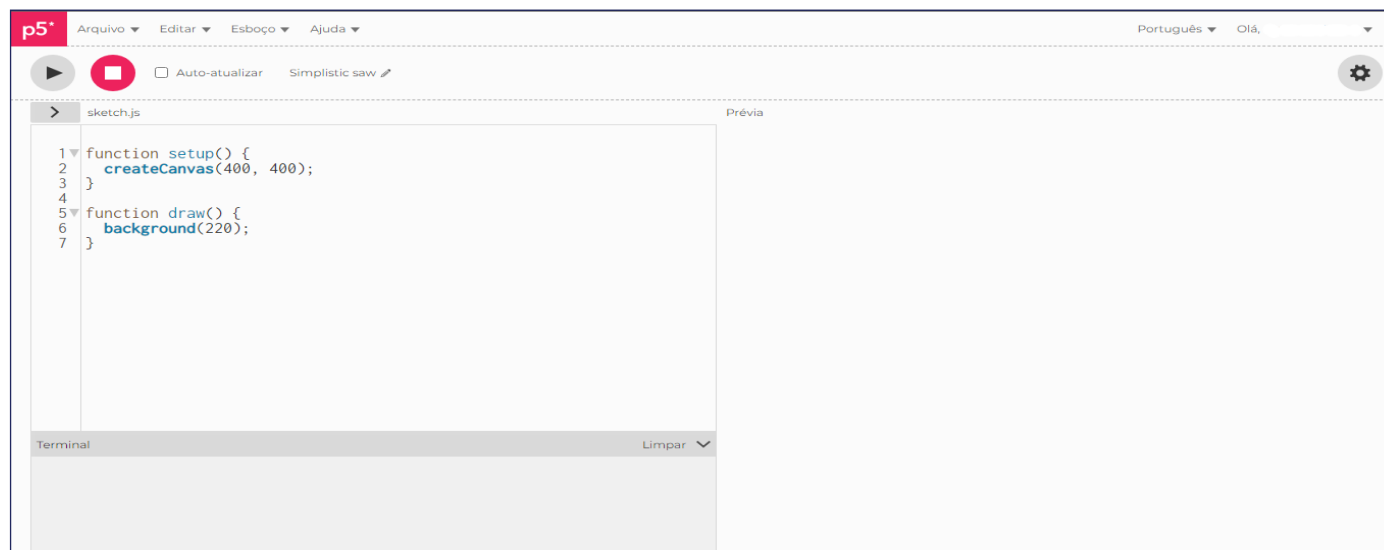
Implementar estruturas de verificação para movimentar a bolinha.



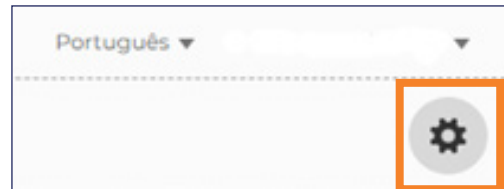
Clique e acompanhe a aula na Alura

Criando bolinha e o cenário

Na última aula, concluímos a programação do jogo Pong no Scratch, utilizando blocos de programação e atores criados por meio da página de desenho do próprio Scratch. A partir desta aula, vamos aprender a programar o mesmo jogo utilizando uma plataforma chamada p5, que utiliza a linguagem JavaScript. Vamos iniciar abrindo o *site* <https://editor.p5js.org/>. Devemos vincular um e-mail ou o conta do GitHub para programar nessa plataforma. Uma vez nela, a seguinte tela será exibida:



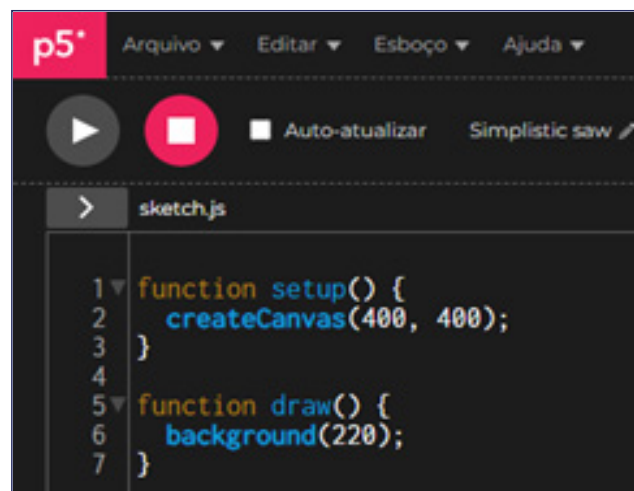
No canto superior direito da tela, há um ícone de engrenagem; clicaremos nele:







Poderemos configurar o ambiente de desenvolvimento da maneira como desejarmos, alterando a cor de fundo, o tamanho da letra etc.

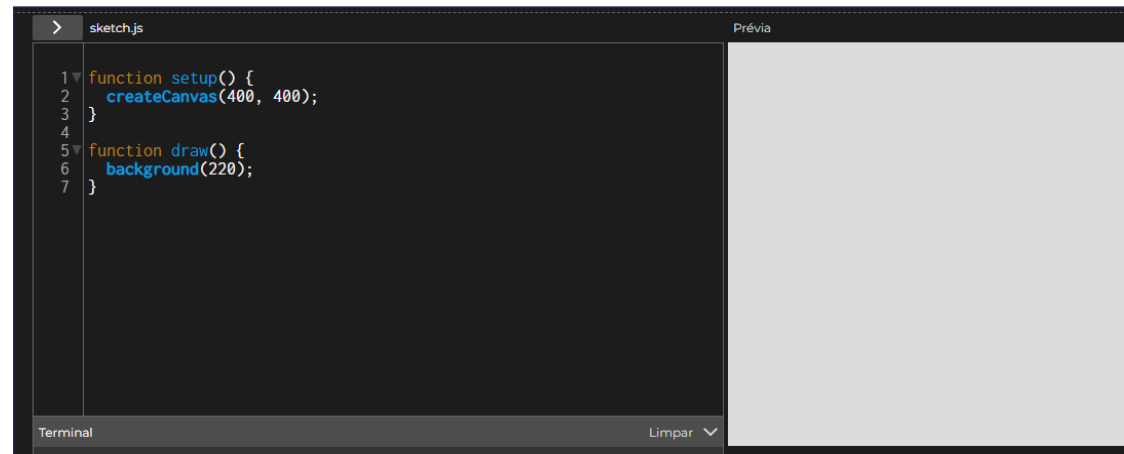


Logo que entramos na página, nos deparamos com algumas linhas de código e alguns botões. Vamos entender a funcionalidade desses elementos.



O botão cinza é o *Play*, ou seja, ele serve para executar ou iniciar o código: . O *Play* do p5 equivale ao botão *bandeira verde*  do Scratch. O botão ao lado do *Play*, na cor rosa, é o *Stop*: . Ele serve para parar o código, assim como o botão vermelho no Scratch . Clique no botão *Play* no p5 para ver o que acontece.

Ao clicar no botão *executar*, podemos observar que a seguinte tela cinza aparece.



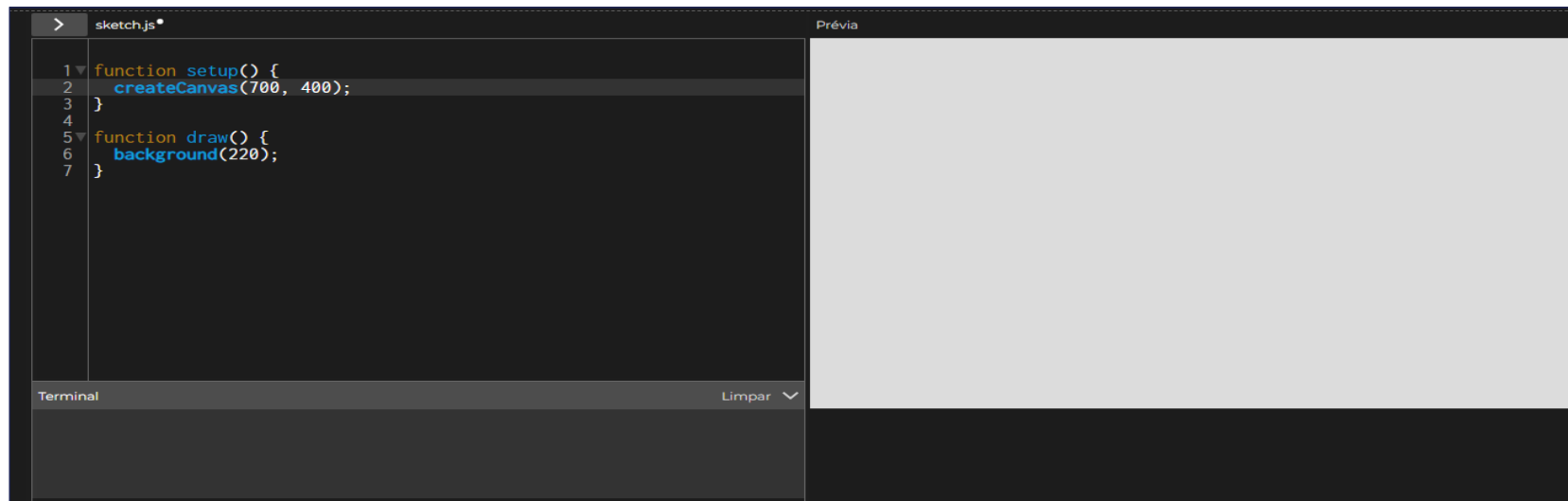
Essa tela é chamada de *Prévia*. É nela que todo o código é criado e, ao ser executado, mostrará os eventos e as ações programados.

O código responsável por criar esta tela quadrada é da linha 2, o **createCanvas**.

```
1 function setup() {  
2   createCanvas(400, 400);
```

Perceba que ele possui dois valores. O que acontece se alterarmos um dos valores? Substitua o primeiro valor 400 pelo valor 700 e, então, clique no botão *executar*.

Podemos observar que a largura da tela aumentou: isso significa que o primeiro valor corresponde à largura da tela.



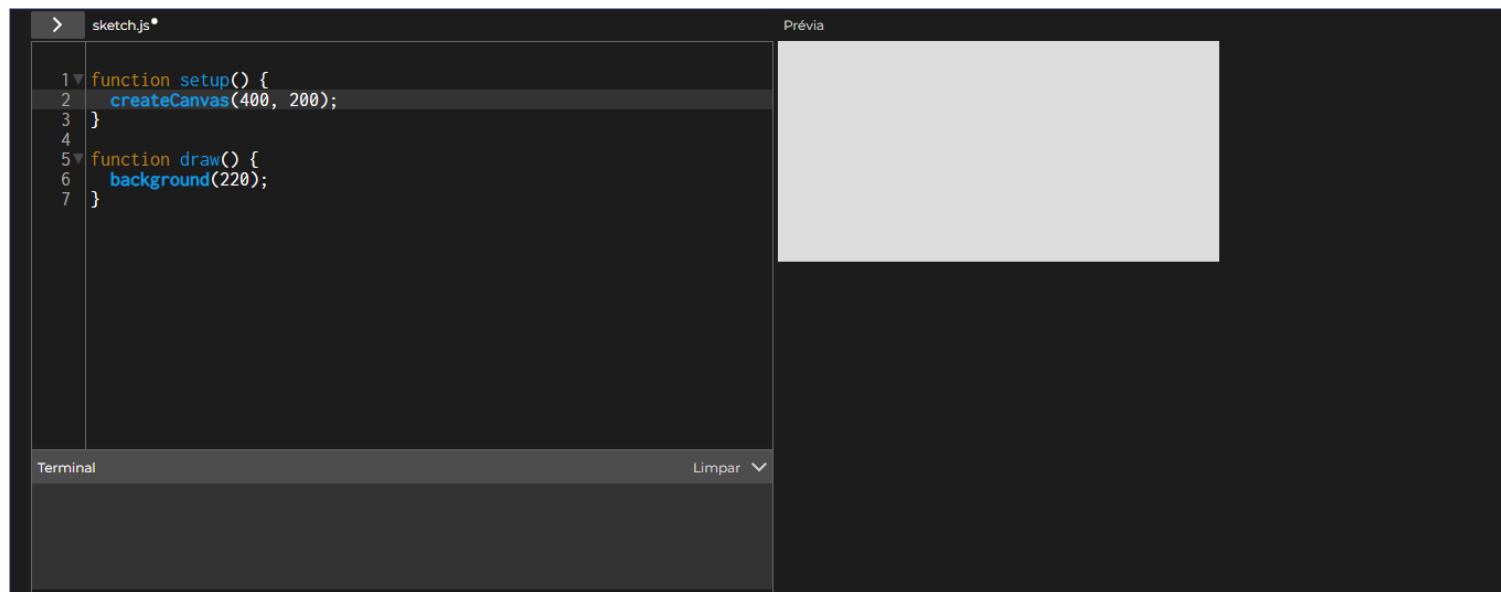
The screenshot shows a code editor interface with a dark theme. The main editor area displays the following JavaScript code:

```
1 function setup() {  
2   createCanvas(700, 400);  
3 }  
4  
5 function draw() {  
6   background(220);  
7 }
```

Below the code editor is a terminal window with the label "Terminal" and a "Limpar" button. To the right of the code editor is a preview window labeled "Prévia" showing a light gray rectangular canvas.

Faça mais um teste, retornando o primeiro valor para 400 e alterando o segundo valor para 200.

O resultado da alteração do segundo valor faz com que o retângulo cinza diminua sua altura. Portanto, o segundo valor corresponde à altura da tela.



The image shows a screenshot of a P5.js sketch editor interface. The editor is divided into three main sections: a code editor on the left, a preview window on the right, and a terminal at the bottom.

The code editor shows the following JavaScript code:

```
1 function setup() {  
2   createCanvas(400, 200);  
3 }  
4  
5 function draw() {  
6   background(220);  
7 }
```

The preview window, labeled "Prévia", shows a light gray rectangle on a black background, representing the canvas. The terminal at the bottom is labeled "Terminal" and has a "Limpar" (Clear) button.

O outro código que já existe ao carregarmos a página é o **background**.

```
4  
5▼ function draw() {  
6   background(220);  
7 }
```

Esse código possui uma função, representada por **function**, chamada **draw** (desenhar). Dentro dessa função, há a propriedade **background** (fundo de tela) e o número **220** entre parênteses. Essa propriedade define qual será a cor de fundo da tela. O valor **220** corresponde à cor cinza. Podemos testar o resultado de alguns valores diferentes.

```
5▼ function draw() {  
6   background(100);
```

```
5▼ function draw() {  
6   background(300);
```

```
5▼ function draw() {  
6   background(0);
```

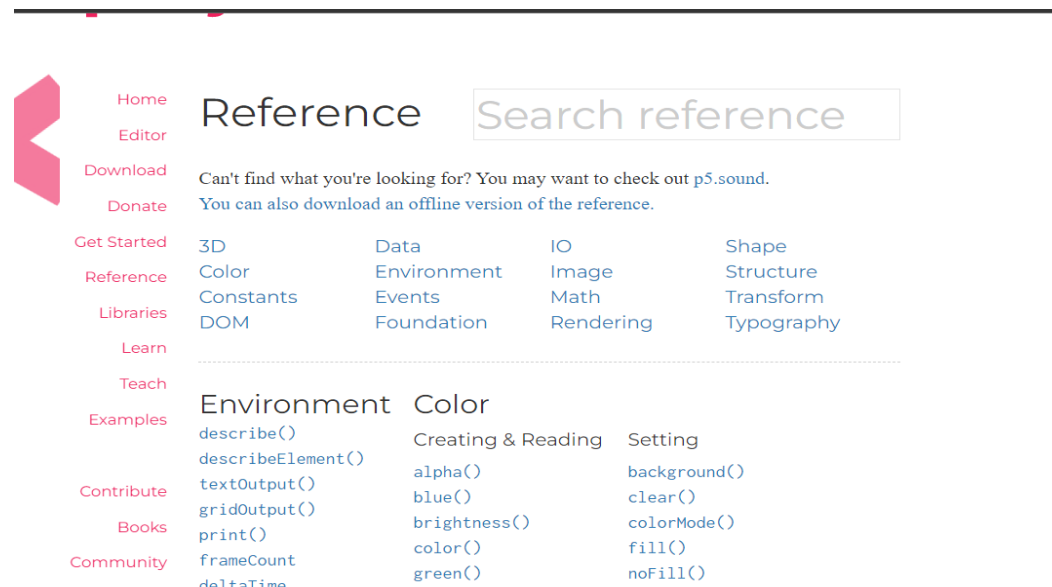
Após algumas tentativas é possível descobrir que o valor correspondente à cor preta, por exemplo, é **0**.

O próximo passo é criar as personagens do jogo Pong. No entanto, diferentemente do Scratch, no p5 não há menus coloridos que se dividem em funções. Então, como saber quais comandos e funções podemos criar no p5? Por meio de documentações.

A documentação é uma referência que nos ajuda a conhecer os elementos que podem ser programados na plataforma. Criar a documentação de um software é uma boa prática de programação, que encontramos nos principais ambientes de desenvolvimento. Para acessar a documentação do p5, vá até o menu *Ajuda* e clique em *Referência*.



Ao carregar a página, encontramos muitas dicas e informações sobre os métodos e propriedades que podem ser programados no p5.



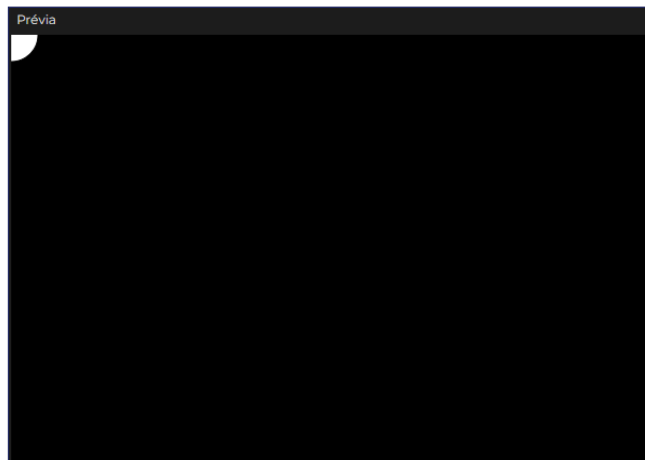
Procure pela função **Circle**.

A função **circle** será utilizada para criarmos a bolinha do jogo Pong. Sua sintaxe é **circle(x, y, d)**, ou seja, são três parâmetros. O **x** e **y** correspondem às coordenadas do objeto e o **d** corresponde ao diâmetro.

Abaixo da linha com o código 'background (0);' digite 'circle (0,0,50);' para criar um círculo no centro da tela com parâmetros (0, 0, 50).

```
4  
5 function draw() {  
6   background(0);  
7   circle(0,0,50);  
8 }
```

Ao executarmos o código, podemos observar que o círculo foi criado no canto superior esquerdo da tela.



Isso acontece porque o plano cartesiano do p5 tem o centro do sistema de referência posicionado ao canto superior esquerdo da tela.

Além de adicionar os valores numéricos diretamente na função, é possível criar variáveis para armazenar estes valores e, então, usar estas variáveis como parâmetros. Para criar uma variável no p5, utilizamos a palavra reservada **let**. Uma palavra reservada significa que ela possui uma funcionalidade e só poderá ser utilizada para este feito. Nas três primeiras linhas de código, crie três variáveis: **xBolinha**, **yBolinha** e **diametro**.

```
1 let xBolinha
2 let yBolinha
3 let diametro
```

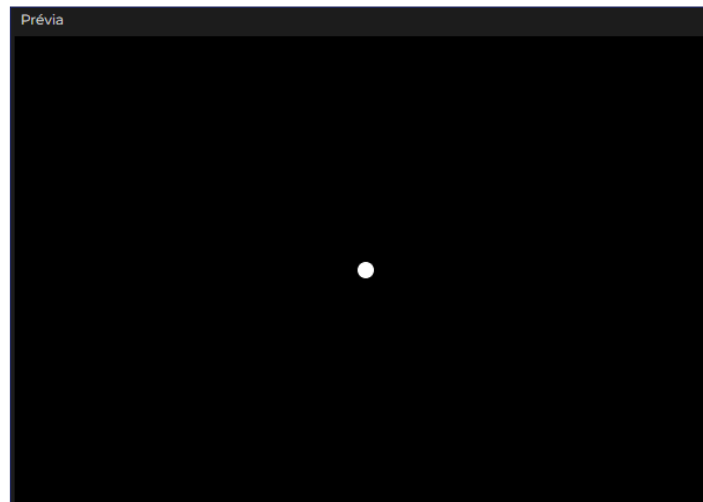
Cada uma dessas variáveis terá um valor. Teste alguns e, depois atribua os valores: **300**, **200** e **15**.

```
1 let xBolinha = 300;
2 let yBolinha = 200;
3 let diametro = 15;
```

Agora, substitua os valores antigos da função **circle** pelas variáveis criadas.

```
function draw() {  
  background(0);  
  circle(xBolinha,yBolinha,diametro);  
}
```

Ao carregarmos o jogo, podemos observar que a bolinha foi criada e está no centro da tela.



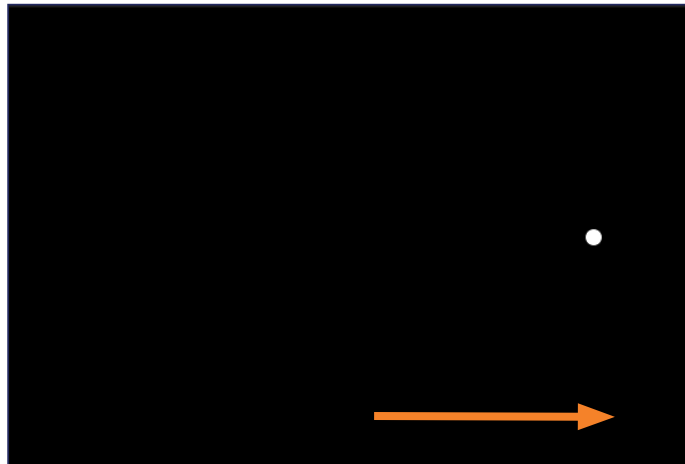
Já possuímos a bolinha do jogo. Vamos tentar fazê-la se movimentar. Para isso, criaremos uma variável chamada **velocidadeXBolinha**, que irá armazenar a movimentação da bolinha no eixo x da tela. Como sugestão para o valor inicial, adicione 5.

```
4 let velocidadeXBolinha = 5;
```

Devemos adicionar o novo valor à posição atual da bolinha no eixo x, e quem armazena esta informação é a variável **xBolinha**; portanto, vamos até a função **draw**, abaixo da linha de código 'circle', e adicionamos o seguinte código:

```
12 | xBolinha = xBolinha + velocidadeXBolinha;
```

Ao executarmos o jogo, podemos observar que a bolinha se movimenta para a direita até desaparecer da tela.



O código está executando corretamente, mas ainda podemos melhorá-lo. No trecho:

```
12  xBolinha = xBolinha + velocidadeXBolinha;
```

Exclua uma das menções **xBolinha** e acrescente o sinal positivo **+** antes do sinal de igualdade **=**. O resultado será o mesmo, mas haverá menos linhas de código.

```
10▼ function draw() {  
11    background(0);  
12    circle(xBolinha,yBolinha,diametro);  
13    xBolinha += velocidadeXBolinha;
```

A bolinha já se movimenta para o lado, mas é necessário que ela se movimente para cima e para baixo. Crie uma variável chamada **velocidadeYBolinha** e defina o valor para 5.

```
5 let velocidadeYBolinha = 5;
```

Depois, acrescente à posição atual da bolinha a nova variável.

```
15 yBolinha += velocidadeYBolinha;
```

Ao executarmos o jogo, podemos observar que a bolinha está se movimentando na diagonal, pois as velocidades nos dois eixos estão sendo aplicadas ao mesmo tempo.



Colisão com as bordas

A bolinha do jogo Pong já se movimenta e ultrapassa as bordas da tela. Quando criamos esta barreira no Scratch, utilizamos o bloco **se tocando em**; porém, como você já deve ter notado, estes blocos não existem no p5. Felizmente, temos códigos equivalentes. Vamos começar adicionando a estrutura condicional de verificação **se** (em inglês, **if**). Então, se a bolinha ultrapassar a largura da tela (representada pela palavra **width**), ela deverá retornar, invertendo a direção de seu movimento.

```
17▼  if (xBolinha > width) {  
18  
19  }
```

Para inverter a direção do movimento, vamos multiplicar a velocidade atual da bolinha no eixo x pelo **-1**. O código ficará assim:

```
17▼  if (xBolinha > width) {  
18      velocidadeXBolinha *= - 1;  
19  }
```

Para testarmos o eixo x, vamos desabilitar a instrução que adiciona velocidade ao eixo y da bolinha. Para isso, acrescentaremos duas vezes o caractere `/` antes da instrução:

```
15  //yBolinha += velocidadeYBolinha;
```

Perceba que a instrução fica cinza: isso significa que ela não será interpretada pelo p5.

Ao carregarmos o jogo, podemos observar que a bolinha bate na extremidade à direita da tela, inverte sua direção e se movimenta até a outra extremidade. No entanto, ao chegar na borda à esquerda, some novamente.

Isso acontece porque a instrução está verificando somente a extremidade que se refere ao fim da tela, do lado direito. Para que seja possível identificar a borda à esquerda, é necessário acrescentar uma nova instrução à **xBolinha < 0**. Então, quando a bolinha tocar nesta extremidade, o seu valor será menor que 0, pois ela está em constante movimento.

```
if (xBolinha > width || xBolinha < 0) {  
    velocidadeXBolinha *= - 1;  
}
```

Utilizaremos um conector entre as instruções, que é a palavra **ou**, representado pelo símbolo **||**.

```
if (xBolinha > width || xBolinha < 0) {  
    velocidadeXBolinha *= - 1;  
}
```

Ao carregarmos o jogo, podemos observar que a bolinha está tocando nas bordas e voltando, sem sair da tela.

Agora, é necessário realizar os mesmos passos para que a bolinha identifique as bordas superior e inferior da tela e, ao tocar nelas, inverta sua direção de movimento. Devemos desabilitar a instrução que adiciona velocidade no eixo x da bolinha e habilitar novamente a instrução que adiciona velocidade no eixo y da bolinha.

```
14 //xBolinha += velocidadeXBolinha;  
15 yBolinha += velocidadeYBolinha;
```

Em seguida, copie o código da estrutura condicional que criamos e cole logo abaixo, utilizando os atalhos **CTRL + C** (copiar) e **CTRL + V** (colar).

```
17▼ if (xBolinha > width || xBolinha < 0) {  
18     velocidadeXBolinha *= - 1;  
19 }  
20  
21▼ if (xBolinha > width || xBolinha < 0) {  
22     velocidadeXBolinha *= - 1;  
23 }
```

Em seguida, modifique a instrução do trecho copiado, mudando de **> width** para **> height**, que significa altura

```
if (xBolinha > width || xBolinha < 0) {  
    velocidadeXBolinha *= - 1;  
}  
  
if (xBolinha > height || xBolinha < 0) {  
    velocidadeXBolinha *= - 1;  
}
```

Substitua a variável **xBolinha** pela variável **yBolinha**.

```
if (yBolinha > height || yBolinha < 0) {  
    velocidadeXBolinha *= - 1;  
}
```

Então, substitua a variável **velocidadeXBolinha** para **velocidadeYBolinha**.

```
if (yBolinha > height || yBolinha < 0) {  
    velocidadeYBolinha *= - 1;  
}
```


Habilite novamente a instrução que adiciona velocidade à bolinha no eixo x, e você terá um código muito semelhante a este:

```
sketch.js
1 let xBolinha = 300;
2 let yBolinha = 200;
3 let diametro = 15;
4 let velocidadeXBolinha = 5;
5 let velocidadeYBolinha = 5;
6
7 function setup() {
8   createCanvas(600, 400);
9 }
10
11 function draw() {
12   background(0);
13   circle(xBolinha, yBolinha, diametro);
14   xBolinha += velocidadeXBolinha;
15   yBolinha += velocidadeYBolinha;
16
17   if (xBolinha > width || xBolinha < 0) {
18     velocidadeXBolinha *= -1;
19   }
20
21   if (yBolinha > height || yBolinha < 0) {
22     velocidadeYBolinha *= -1;
23   }
24 }
25
26 }
```

Ao executarmos o jogo, a bolinha toca em todas as bordas e é rebatida logo em seguida devido à alteração de direção.

Desafio

Durante esta aula, você iniciou a criação do jogo Pong utilizando uma das linguagens mais populares do mundo, a Javascript. Com ela, podemos criar elementos, manipular funções e alterar propriedades visuais no jogo. Falando em propriedades, os aspectos visuais de um jogo contribuem para maior imersão e entusiasmo de quem joga.

Sabendo disso, busque na documentação do p5 a funcionalidade que permite alterar a cor de fundo do jogo para qualquer uma existente, e não apenas tons de cinza e preto como tratado em aula.

Avalie este material acessando o link: <https://forms.gle/EcEZdj59zGS9QTWY9>