start by alura

Aula 5

Novo projeto: quente e frio

▶ Unidade

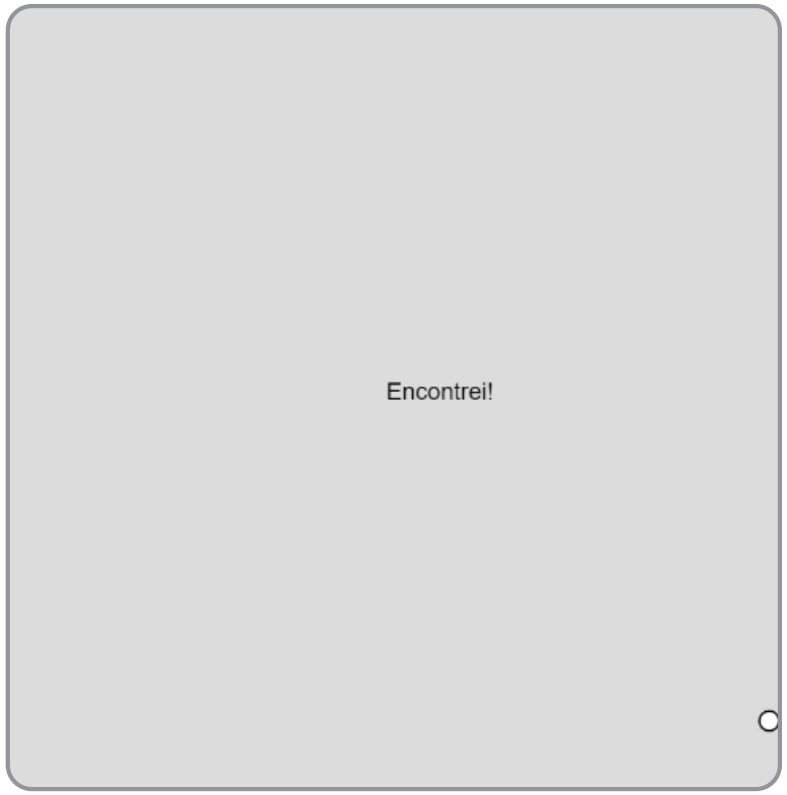
Lógica de programação: criando arte interativa com p5.js

O que vamos aprender?

- Aplicar conceitos de coordenadas aleatórias para posicionar elementos na tela de um projeto.
- Compreender a diferença entre números inteiros e decimais em operações de comparação.
- Testar e depurar código utilizando **`console.log()`**.

ACESSE A PLATAFORMA START

Encontrando um ponto



Na aula passada, finalizamos o projeto Monalisa adicionando o efeito de fazer os olhos da nossa obra de arte seguirem o ponteiro do mouse. Nesta aula, iniciaremos um novo projeto chamado Quente e frio. Vamos lá?

Iniciaremos um novo projeto no p5.js. Por isso, é importante que você faça login na sua conta e se certifique de que o projeto anterior, Monalisa, está devidamente salvo, como fizemos nas aulas anteriores. Em seguida, inicie um novo projeto.

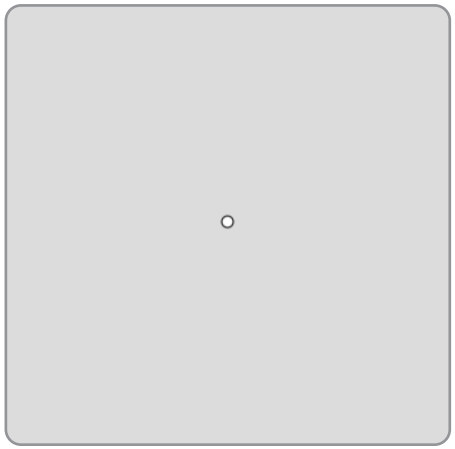
O projeto que criaremos será chamado Quente e frio. A ideia é a seguinte: imagine que temos um objetivo que, no nosso caso, será encontrar um ponto específico da tela. Quanto mais perto chegamos desse objetivo, o programa deverá nos sinalizar que estamos quentes; porém, ao nos distanciarmos, ele deve indicar que estamos frios. Vamos começar?

Com um novo projeto no p5.js iniciado, o primeiro passo será adicionar um ponto no centro da tela. Para isso, utilizaremos a função **`circle`** e adicionaremos um círculo de 10 cm de diâmetro. O código será o seguinte:

```
function setup() {
    createCanvas(400, 400);
  }

  function draw() {
    background(220);
    circle(200,200, 10);
}
```

O resultado será um círculo branco no centro da tela:



Agora, precisamos fazer com que esse círculo apareça sempre em um lugar aleatório da tela. Para isso, trabalharemos com a função **random** que, no JavaScript, é utilizada para gerar um número aleatório. Assim, geraremos um valor aleatório entre 0 a 400, que é o tamanho da nossa tela, tanto para a posição X quanto para a posição Y do círculo criado. O código ficará assim:

```
function draw() {
    background(220);
    circle(random(400),random(400), 10);
}
```

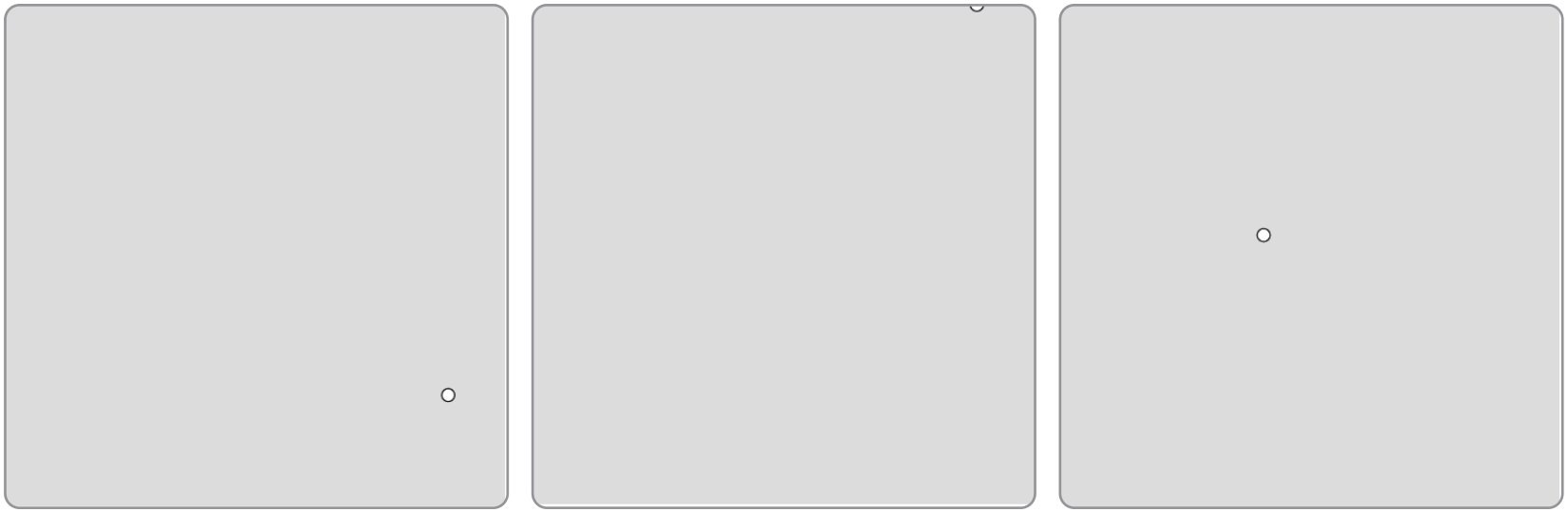
Ao executar o programa, você verá o círculo mudando de posição pelo cenário muito rapidamente de forma aleatória. Isso acontece porque estamos usando a função **draw**, que reinicializa o mesmo código várias vezes. Nos outros projetos, essa função nos ajudou no desenvolvimento, mas, nesse caso, queremos que a posição do círculo seja sorteada apenas uma vez e, para isso, utilizaremos a função **setup**.

Para trabalharmos com os mesmos valores que são utilizados na função **setup** e na função **draw**, precisamos criar variáveis. Assim, na função **setup**, criaremos uma variável **x** e uma variável **y**, que receberão o número aleatório, e aplicaremos essas variáveis na função **circle**. O código atualizado ficará assim:

```
function setup() {
    createCanvas(400, 400);
    x = random(400);
    y = random(400);
  }

function draw() {
    background(220);
    circle(x, y, 10);
}
```

Ao executarmos o projeto, veremos que o círculo aparece em uma posição diferente do cenário a cada interação. Observe:



Quando trabalhamos com variáveis, uma boa prática é declarar todas elas no início do código. Faremos isso utilizando a palavra reservada **`let`**. Assim, vamos declarar as nossas variáveis antes da função **`setup`**.

```
let x;
let y;

function setup() {
    //código omitido
```

Agora, precisamos pensar em um modo de programar para que o mouse exiba a mensagem “Encontrei!” toda vez que encostar no círculo que acabamos de configurar. Podemos fazer isso utilizando a estrutura condicional **`if`**.

A condição que criaremos será a seguinte: quando a posição do ponteiro do mouse no eixo X for igual à posição X do círculo, exibiremos a mensagem “Encontrei!”. Portanto, o código será o seguinte:

```
function draw() {
    background(220);
    circle(x, y, 10);

    if(mouseX == x) {

    }
}
```

Perceba que estamos usando dois sinais de igualdade (==). Isso acontece pois utilizamos um único sinal de igualdade quando queremos guardar um valor em uma variável, e usamos dois sinais quando queremos comparar valores, que é o nosso caso atual.

Agora, vamos programar a ação de exibir o texto!

Para exibir o texto, utilizamos a função **text** com três parâmetros: o texto a ser exibido, escrito entre aspas simples, a posição X e a posição Y do texto. O código ficará da seguinte forma:

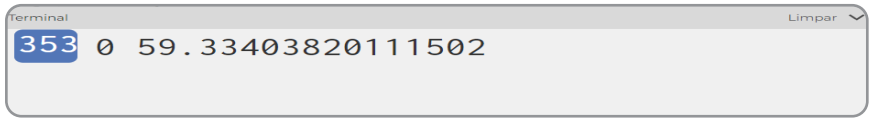
```
    if(mouseX == x) {
        text('Encontrei!', 200, 200);
    }
```

Feito isso, teste seu projeto. Você perceberá que a mensagem ainda não está sendo exibida. O problema está no formato dos valores de **mouseX** e da variável **x**. Assim, utilizaremos a função **console.log** para exibir esses valores no terminal. Observe:

```
//código omitido
    console.log(mouseX,x);

    if(mouseX == x) {
        text('Encontrei!', 200, 200);
    }
```

Agora, ao executarmos o projeto, teremos dois valores sendo exibidos no terminal:

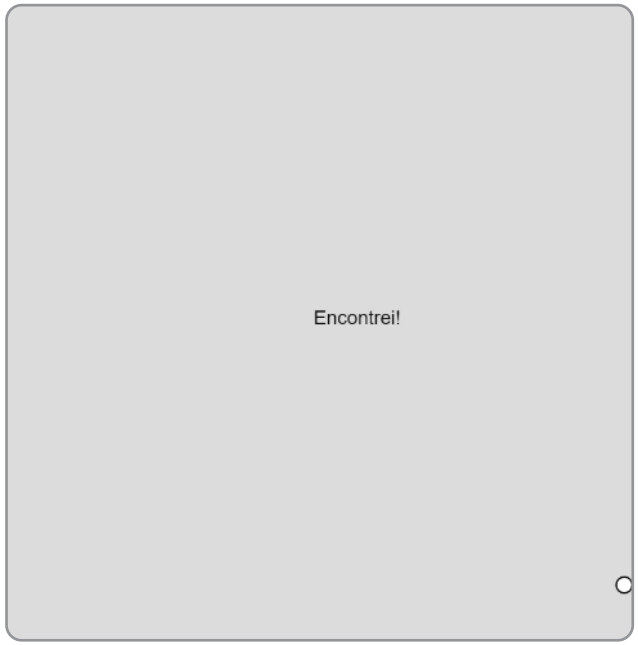


Perceba que *0*, que representa a posição X do ponteiro do mouse, é um número inteiro, enquanto o valor da variável **x** é um número real. Para que tudo funcione, precisamos que os dois valores tenham o mesmo formato.

Assim, na função **setup**, alteraremos o valor sorteado para a variável **x** para que ele se torne um número inteiro. Observe que o mesmo deve ser feito para a variável **y**. Faremos isso através da função **int**. Confira o código:

```
function setup() {
    createCanvas(400, 400);
    x = random(400);
    x = int(x);
    y = random(400);
    y = int(y);
}
```

Ao executar o projeto, você verá a mensagem sendo exibida corretamente quando o ponteiro do mouse e círculo se encontram, conforme a imagem ao lado:



Quase tudo está funcionando, porém, como estamos verificando apenas a posição no eixo X, a mensagem continua sendo exibida mesmo que o mouse e o círculo estejam em alturas diferentes da tela.

Esse é um desafio que iremos solucionar nas próximas aulas!

Até lá!

start by alura

Bons estudos!