start by alura

Habilidades trabalhadas nesta aula:

(EM13CO02) Explorar e construir a solução de problemas por meio de refinamentos, utilizando diversos níveis de abstração desde a especificação até a implementação.

(EM13CO18) Planejar e gerenciar projetos integrados às áreas de conhecimento de forma colaborativa, solucionando problemas, usando diversos artefatos computacionais.

(EMIFMAT04) Reconhecer produtos e/ou processos criativos por meio de fruição, vivências e reflexão crítica na produção do conhecimento matemático e sua aplicação no desenvolvimento de processos tecnológicos diversos.

Aula 7

Comandos em JavaScript

▶ Unidade

Lógica de programação: criando arte interativa com p5.js

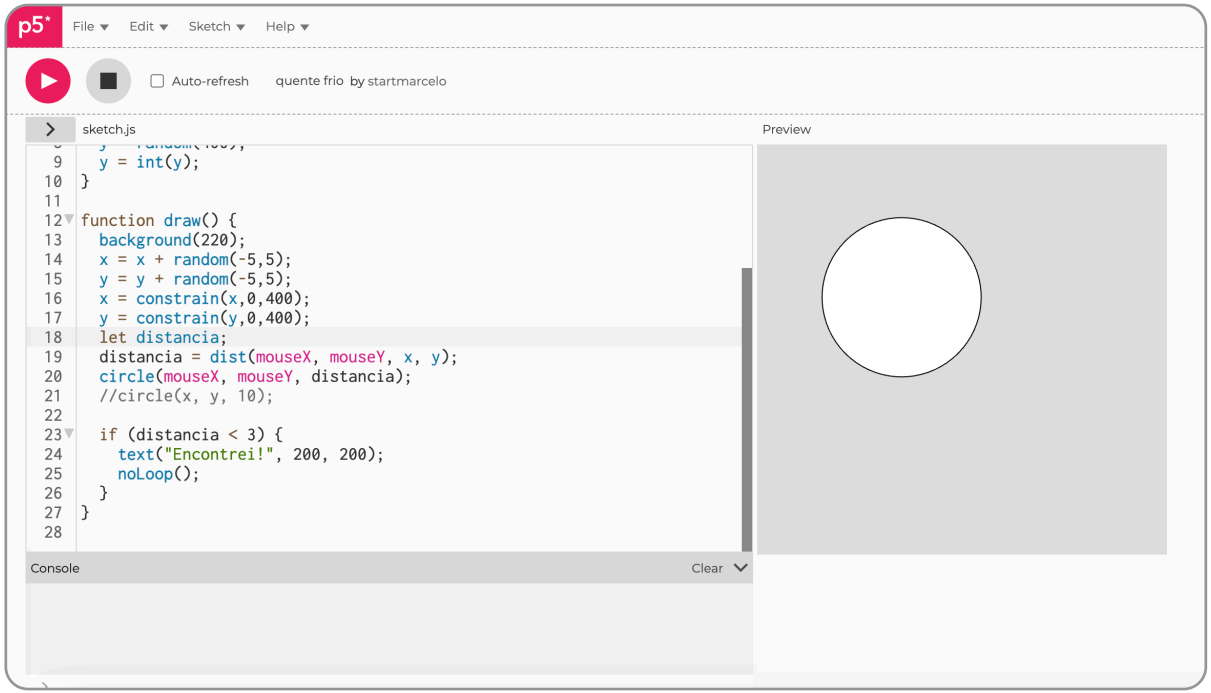
O que vamos aprender?

- Compreender o uso da função **`dist()`** do p5.js para calcular distâncias entre dois pontos.
- Aplicar o conceito de distância para ajustar o comportamento visual de elementos interativos.
- Associar diferentes abordagens para calcular distâncias.

ACESSE A PLATAFORMA START

Aplicando o comando `dist()`

Na aula passada, exploramos como usar nossos conhecimentos de trigonometria para resolver problemas, incluindo o teorema de Pitágoras. Nesta aula, aprenderemos a utilizar as funções próprias do p5.js para realizar cálculos de distância e inserir elementos que tornarão nosso jogo ainda mais divertido.



Nesta aula, os estudantes farão algumas alterações no código para otimizar o jogo. Estimule-os a refletir sobre o que falta e o que pode ser aprimorado. Por exemplo, uma experiência que depende do console pode não ser a mais adequada para quem está jogando, uma vez que o jogador precisa ler as mensagens enquanto move o cursor. Incentive-os a acessar a documentação do p5.js e a buscar alternativas para melhorar o código e a experiência do jogo.

O objetivo do jogo é encontrar um ponto oculto, mas, no momento, estamos sempre olhando para um lado da tela, onde está o terminal, enquanto o projeto está do outro lado. Portanto, o que podemos fazer? Seria muito útil se, em vez de verificarmos o resultado na linha 22, que é o console.log, pudéssemos visualizar os resultados diretamente na tela do jogo.

Que tal criarmos um círculo que estará posicionado nas coordenadas **`mouseX`**, **`mouseY`**? Com base no tamanho desse círculo, saberemos se estamos perto ou longe de encontrar o ponto oculto.

Já sabe como fazer isso?

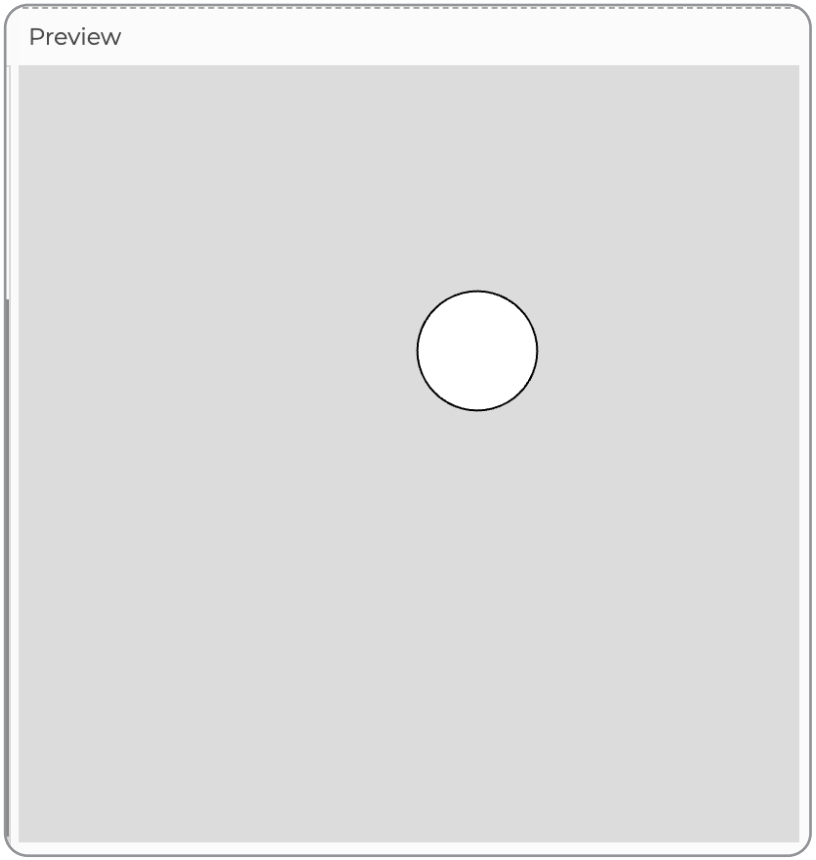
Professor, incentive os estudantes a tentarem implementar a alteração proposta antes de prosseguir. Lembre-os de que já fizemos isso em aulas anteriores.

Após calcularmos a distância, adicionaremos o comando **`circle(mouseX,mouseY,distancia)`**, que desenhará um círculo que segue o mouse. O tamanho (diâmetro) do círculo será alterado com base na distância ao ponto oculto. Nosso código ficará da seguinte forma:

```
distanciaX = mouseX - x;
distanciaY = mouseY - y;
distancia = sqrt(distanciaX*distanciaX + distanciaX*distanciaX);
circle(mouseX,mouseY,distancia);
```

Já consegue ver o resultado?

Por aqui está funcionando. Se estivermos muito longe, a distância será grande, e, portanto, o círculo será grande. Observe a imagem:



No entanto, conforme nos aproximamos e conseguimos localizar esse ponto, o círculo vai diminuindo. Quando o encontrarmos, o texto *Encontrei!* aparecerá na tela.

O que mais podemos fazer?

No bloco de **`if`** vocês verão que já estamos trabalhando com o conceito de comparação: se **`mouseX`** é igual a **`x`** e se **`mouseY`** é igual a **`y`**, ou seja, **`if (mouseX == x && mouseY == y)`**.

No entanto, já temos uma variável que faz essa verificação: a distância. Portanto, se a distância for, por exemplo, menor que 0, 1, 2, ou 3, dependendo do nível de dificuldade que queremos trabalhar, conseguiremos encontrar o nosso ponto. Consideremos o caso em que a distância é menor que 3.

Professor, essa é mais uma alteração que os estudantes podem tentar fazer sozinhos antes de prosseguir. O comando de comparação já está sendo utilizado; o que eles precisam fazer é alterar a variável que está sendo comparada.

Então, alteraremos a condição de comparação que temos no bloco **if**. O código ficará conforme o exemplo a seguir:

```
if (distancia < 3) {
    text('Encontrei!', 200, 200);
    noLoop();
}
```

Visualmente, o resultado será o mesmo que vimos anteriormente. No entanto, agora poderemos controlar o nível de dificuldade do jogo. Isso ocorre porque, quanto maior o valor com o qual comparamos a distância, mais fácil será achar o ponto oculto. Dessa forma, como já temos o projeto completo funcionando, podemos fazer melhorias no código que facilitarão a leitura, mas não alterarão o comportamento do jogo.

Começaremos utilizando um método mais simples para calcular a distância. Para isso, ocultaremos o comando onde fizemos o cálculo da distância, que já está funcionando. Em seguida, utilizaremos comandos da própria biblioteca do p5.js, nesse caso, o comando **`distancia = dist()`**. Dentro dele, vamos passar quatro parâmetros: o ponto do cursor do mouse, que é **`mouseX,mouseY`**, o terceiro parâmetro que é **`X`**, e o quarto parâmetro que é **`Y`**. Assim, estaremos calculando a distância entre os pontos **`(mouseX,mouseY)`** e **`(x,y)`**. Veja como ficará o código no exemplo a seguir.

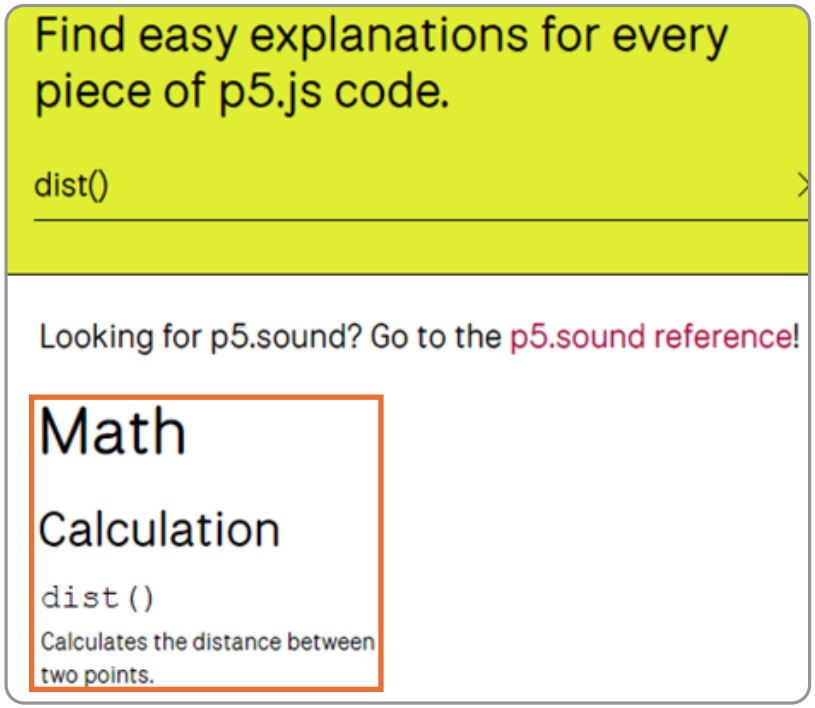
```
distanciaX = mouseX - x;
distanciaY = mouseY - y;
//distancia = sqrt(distanciaX*distanciaX + distanciaX*distanciaX);
distancia = dist(mouseX,mouseY,x,y);
circle(mouseX,mouseY,distancia);
```

Por aqui, tudo funcionou bem. O jogo segue funcionando!

O uso de comentários durante as melhorias no código serve principalmente para nos proteger de erros. Caso o novo código não funcione, não perdemos o antigo que estava funcionando. Após testar e verificar o funcionamento da alteração, podemos apagar as linhas comentadas e deixar o nosso código limpo.

Uma sugestão muito importante é acessar a opção *Ajuda* > *Referência*, localizada no menu superior do editor.

Em seguida, na tela de pesquisa que se abrirá, busque pelo comando *dist()*. Observe a imagem ao lado:



Ao clicar na primeira opção, *Math Calculation dist ( )* abrirá uma nova tela com a explicação sobre a distância entre dois pontos e o que isso significa na prática. Explore e amplie seu entendimento sobre esse comando!

Professor, sugere-se orientar os estudantes a traduzirem a página, pois ela possivelmente abrirá em inglês, conforme o exemplo.

Para finalizar, não se esqueça de alterar o nome do projeto. Para isso, clique na opção *Sore furniture*, localizada acima do código.

Em seguida, exclua esse título e o renomeie como: quente frio.

Por fim, salve o seu projeto clicando na opção *Arquivo > Salvar*. Até a próxima aula!

start by alura

CLIQUE AQUI PARA AVALIAR ESTE MATERIAL