

💡 Habilidades trabalhadas nesta aula:

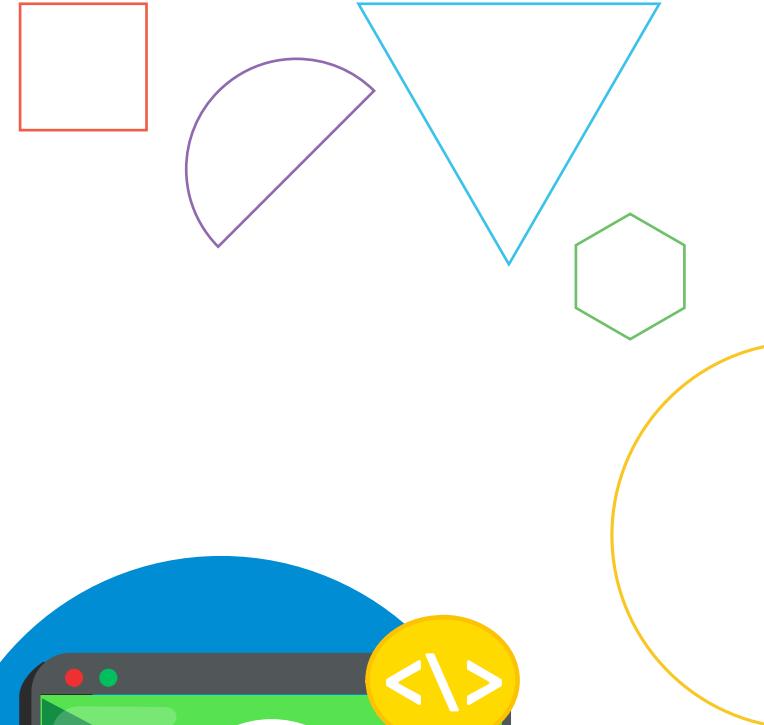
- (EM13CO02)** Explorar e construir a solução de problemas por meio de refinamentos, utilizando diversos níveis de abstração desde a especificação até a implementação.
- (EM13CO18)** Planejar e gerenciar projetos integrados às áreas de conhecimento de forma colaborativa, solucionando problemas, usando 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 2

Adicionando outras formas

► Unidade

Lógica de programação:
criando arte interativa com P5.js

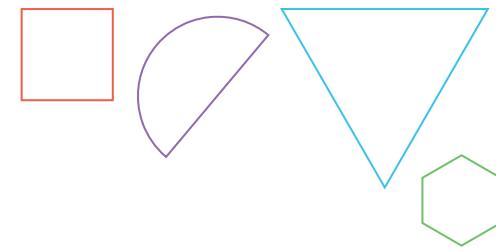


O que vamos aprender?

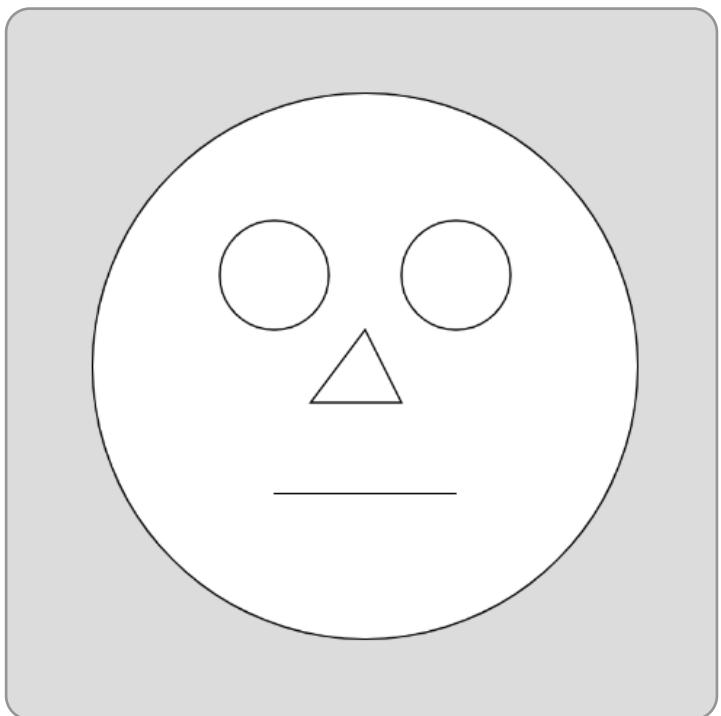
- Aplicar comandos de desenho básicos, como `line()` e `triangle()`, no P5.js.
- Criar novas formas e elementos visuais (ex.: boca, nariz, orelhas) na personagem utilizando as coordenadas do plano cartesiano.
- Relembrar o processo de login e recuperação de projetos anteriores no editor P5.js.



▶ ACESSE A PLATAFORMA START



Outras formas e plano cartesiano



Na aula anterior, iniciamos a construção do projeto Monalisa no P5.js adicionando três círculos diferentes. Nosso desafio agora será continuar esse projeto, adicionando outras formas geométricas por meio de novas funções, desenvolvendo a aparência de nossa personagem.

 Professor, nesta aula, os estudantes se aprofundarão no sistema de coordenadas do P5.js. Esse sistema utiliza um plano cartesiano de duas dimensões como base. Para facilitar esse momento, é importante iniciar a aula retomando com a turma o que é um plano cartesiano e quais os quadrantes que esse plano possui, indicando os valores de cada quadrante.



Para continuar o nosso projeto, precisamos acessar o site do P5.js (<https://editor.p5js.org/>) e logar em nossas respectivas contas, que criamos na aula passada. Para isso, clicaremos em *Log in*, no canto superior direito da tela:



Uma nova tela se abrirá e precisaremos digitar nossos dados de cadastro (e-mail e senha) para logar.

The image shows a login form titled "Log In". It has two input fields: one for "Email or Username" and one for "Password". The password field includes an eye icon to toggle visibility. Below the fields is a "Log In" button.

Professor, também podemos alterar o idioma para o português no menu de idiomas, como fizemos na aula passada. Oriente os estudantes nesse sentido, se houver necessidade.

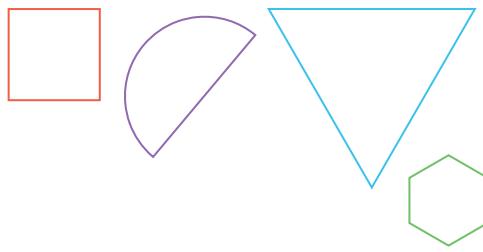


Feito isso, precisamos localizar o projeto que estávamos desenvolvendo. Para isso, acesse o menu no canto superior direito da tela, onde está seu nome de usuário, e selecione a opção *Meus Esboços*:



O esboço que criamos estará salvo com um nome aleatório. Vamos acessá-lo com um duplo clique:

Esboço	Data de Criação	Data Atualizado
Merciful wallaby	15 ago 2024, 10:24:19	15 ago 2024, 10:26:33



Com o projeto aberto, primeiro alteraremos seu nome no menu superior da página clicando no lápis ao lado do nome do arquivo, conforme mostrado na imagem:



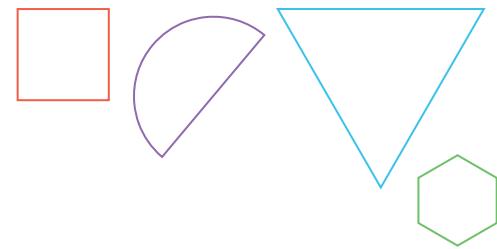
Nomeamos o projeto como Monalisa, porém, você pode adicionar outro nome, se desejar. Agora, começaremos a programar adicionando mais elementos visuais a nossa personagem.



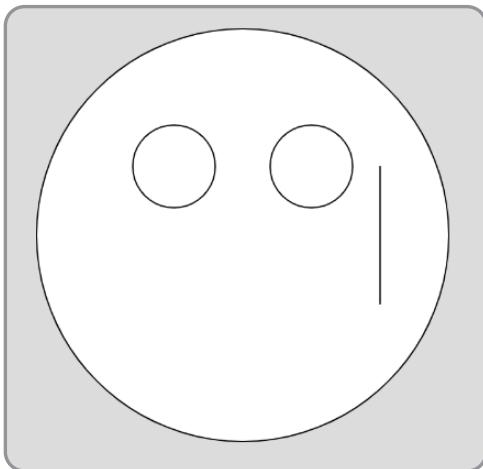
A próxima parte da nossa obra de arte será a boca. Para isso, vamos utilizar a função **line** (em português, linha). Essa função precisa de quatro parâmetros para funcionar. Primeiro colocaremos valores aleatórios e visualizaremos o resultado e, em seguida, faremos as adaptações necessárias. Confira no código:

```
function draw() {  
    background(220);  
    circle (200,200,300);  
    circle (150,150,60);  
    circle (250,150,60);  
    line(300,150,300,250);  
}
```

Você pode colocar valores diferentes se desejar, pois ajustaremos isso logo em seguida! Portanto, teste os valores que desejar.



O resultado, com os valores mostrados, será o seguinte:



Observe que a linha ficou na vertical e do lado direito do cenário. Como queremos que ela represente uma boca, tentaremos deixá-la na horizontal e logo abaixo dos olhos. Mas, como descobrimos quais valores temos que incluir para fazer isso?

Precisamos lembrar que o P5.js trabalha com o sistema de coordenadas de um plano cartesiano, em que os valores são posicionados nos eixos X (horizontal) e Y (vertical).

Portanto, os dois primeiros valores que colocamos na função **line** representam, respectivamente, o ponto de início da linha no eixo X e o ponto de início da linha no eixo Y.



Para facilitar esse entendimento, vamos criar um código que mostra as coordenadas de um ponto X e Y do cenário quando clicamos sobre ele!

Para isso, utilizaremos a função **if**, criando uma condicional que verifica se o mouse foi pressionado. O código ficará assim:

```
function draw() {  
    background(220);  
    circle (200,200,300);  
    circle (150,150,60);  
    circle (250,150,60);  
    line(300,150,300,250);  
  
    if(mousePressed) {  
    }  
}
```

Perceba que, após o **if**, adicionamos a abertura e o fechamento de chaves. Isso serve para indicar que o que colocarmos dentro desse espaço só acontecerá quando o mouse for pressionado.



Agora, dentro do **if**, vamos inserir a função **console.log()**. Essa função é utilizada para mostrar informações no terminal do projeto e pode ser muito útil quando estamos realizando testes. Dentro dessa função, passaremos os valores **mouseX** e **mouseY**. Dessa forma, estamos dizendo que, ao clicar com o mouse, será mostrada no console da página a posição no eixo X e Y onde esse clique ocorreu. Confira o código:

```
if(mouseIsPressed) {  
    console.log(mouseX, mouseY);  
}
```

Para testar, clique na posição onde você deseja que seja iniciado o desenho da linha e, em seguida, na posição onde deseja que a linha termine.



Ao clicar, seu console exibirá alguns valores, conforme a imagem abaixo:

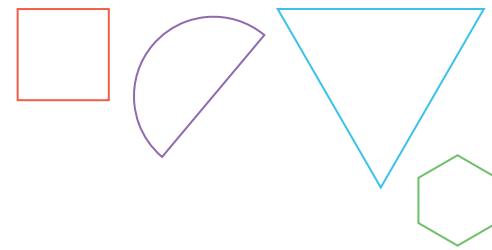


Os valores 147 e 239 indicam a posição no eixo X onde o clique ocorreu, enquanto os valores 274 e 272 indicam a posição no eixo Y. Vamos adicionar esses valores na nossa função `line`.

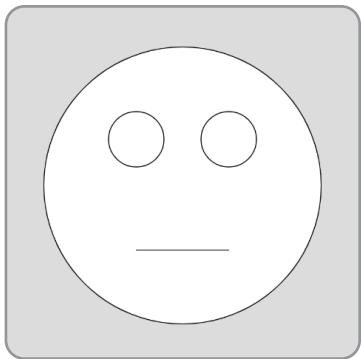
Podemos, inclusive, arredondar os valores. Confira o código:

```
line(150, 270, 250, 270);
```

💡 Professor, neste momento, você pode retomar o plano cartesiano apresentado no início da aula para conversar com os estudantes sobre qual o ponto 0,0 no P5.js. Esse ponto fica no canto superior esquerdo do canvas, o que significa que todo o nosso desenho está no terceiro quadrante de um plano cartesiano. Além disso, mostre que os valores do eixo X aumentam da esquerda para a direita e que os valores do eixo Y aumentam de cima para baixo.



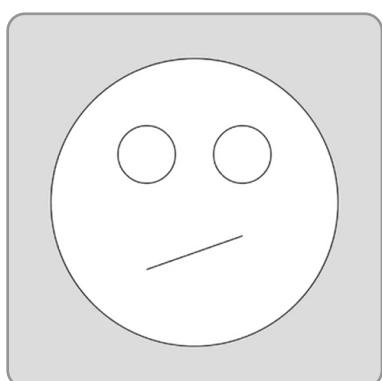
O resultado dessas modificações será o seguinte:



A linha agora está bem posicionada. Se mudarmos um pouco os valores, veremos que nossa personagem terá uma nova expressão. Vamos alterar o último valor do eixo Y para 235 e ver o que acontece:

```
line(150, 270, 250, 235);
```

O resultado será o seguinte:





Agora, adicionaremos um nariz ao nosso rosto. Para isso, vamos buscar no menu de referências a função que nos permite desenhar um triângulo.

2D Primitives

`arc()`

Draws an arc.

`circle()`

Draws a circle.

`ellipse()`

Draws an ellipse (oval).

`line()`

Draws a straight line between two points.

`point()`

Draws a single point in space.

`quad()`

Draws a quadrilateral (four-sided shape).

`rect()`

Draws a rectangle.

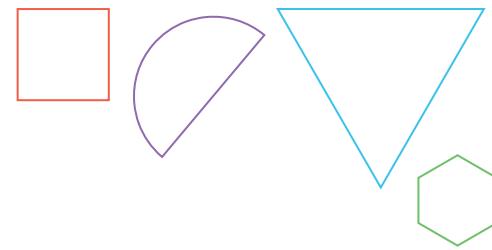
`square()`

Draws a square.

`triangle()`

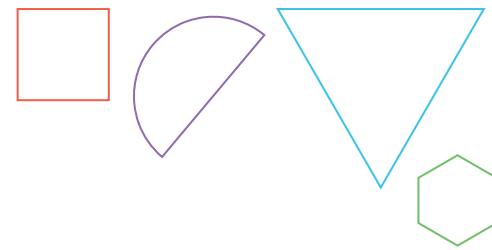
Draws a triangle.

Essa função precisa de seis parâmetros para funcionar. Nesse caso, indicamos dois valores (posição X e Y) para cada uma das pontas do nosso triângulo.

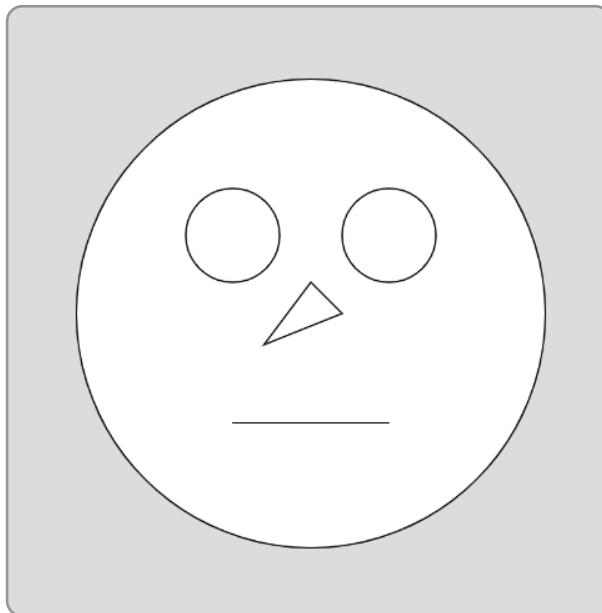


Utilize a função de clique do mouse para descobrir as coordenadas dos pontos onde você deseja que o triângulo seja desenhado. No nosso caso, trabalharemos com os seguintes valores iniciais:

```
function draw() {  
    background(220);  
    circle (200,200,300);  
    circle (150,150,60);  
    circle (250,150,60);  
    line(150,270,250,235);  
    triangle(200,180,170,220,220,200);
```

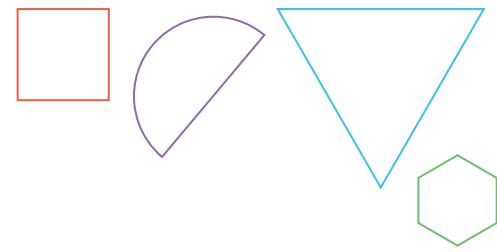


O resultado para os valores mostrados será o seguinte:

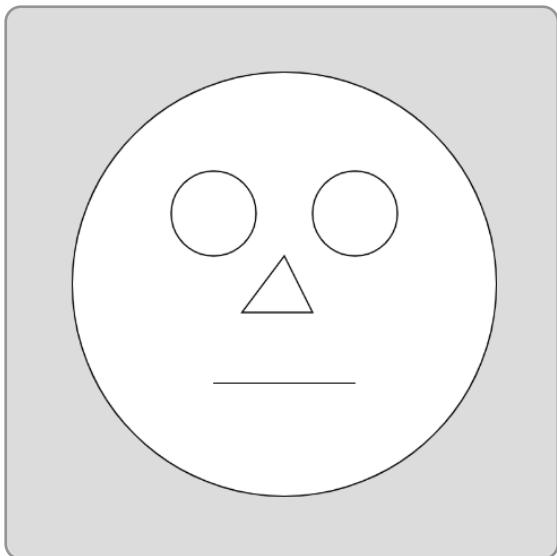


Se quisermos que a base do triângulo fique um pouco mais reta, podemos alterar o último valor no eixo Y para 220. Observe o código abaixo:

```
triangle(200, 180, 170, 220, 220, 220);
```



Ao final, o nosso rosto estará assim:



Explore novos valores para posicionar o nariz da sua obra de arte!

► CLIQUE AQUI PARA AVALIAR ESTE MATERIAL