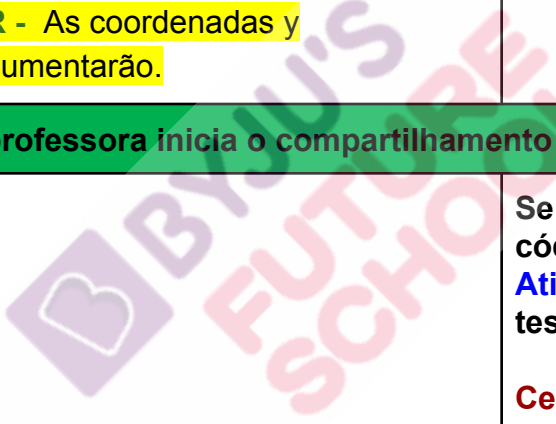



Tópico	JOGO COM IA	
Descrição da Aula	A criança revisa e reaplica todos os conceitos de <i>machine learning</i> (aprendizado de máquina) para aprimorar um jogo popular, fazendo com que funcione com redes neurais. Assim, aprenderá a aplicar IA no mundo real.	
Aula	ADV-C141	
Duração da Aula	55 min	
Objetivo	<ul style="list-style-type: none"> Adicionar código para acessar a webcam, inicializar e executar o modelo poseNet. Adicionar código para acessar noseX e noseY no arquivo characters_environment.js. Adicionar código para a função <code>startGame()</code>. 	
Recursos Necessários	<ul style="list-style-type: none"> Recursos da Professora <ul style="list-style-type: none"> Utilize suas credenciais de login do gmail Fone de ouvido com microfone Bloco de notas e caneta Webcam Recursos do Aluno <ul style="list-style-type: none"> Utilize suas credenciais de login do gmail Fone de ouvido com microfone (opcional) Bloco de notas e caneta Webcam 	
Estrutura da Aula	Introdução Atividade Dirigida pela Professora Atividade Dirigida pelo Aluno Encerramento	5 min 15 min 35 min 5 min
Passos da Aula	Dizer	Fazer
Passo 1: Introdução (5 min)	Na aula anterior, havíamos feito um pequeno protótipo de um jogo IA Mario. Em que aprendemos o	Peça para o aluno ativar o modo Tela cheia .

	<p>sistema de coordenadas da tela. E também aprendeu a lógica por trás do movimento do Mario.</p> <p>P - Se nos movermos da esquerda para a direita na tela, qual coordenada aumentará?</p> <p>R - As coordenadas x aumentarão.</p> <p>P - Se passarmos de cima para baixo na tela, qual coordenada aumentará?</p> <p>R - As coordenadas y aumentarão.</p>	
A professora inicia o compartilhamento de tela		
<p>Passo 2: Atividade Dirigida pela Professora (15 min)</p>		<p>Se você está baixando o código-fonte completo da Atividade da Professora 3, teste em seu sistema.</p> <p>Certifique-se de que ele faça o teste, clicando no botão GO LIVE do VS CODE. Isso resulta na execução do arquivo no servidor ao vivo do programa.</p>  <p>Como estamos usando arquivos de imagem e som, e o p5.js não nos permite executar esses tipos de arquivos a partir de um sistema local, ele precisa ser executado em um servidor.</p> <p>Por favor, siga o fluxo da</p>

		<p>aula.</p> <p>Explique como copiar e colar alguns códigos do protótipo do jogo IA Mario no arquivo main.js.</p> <p>Explique o CSS para visualização da webcam.</p> <p>Explique o que e onde adicionar código no arquivo characters_environment.js.</p> <p>Deixe o aluno copiar e colar o código do protótipo do jogo IA Mario no arquivo main.js.</p> <p>Deixe o aluno adicionar CSS para visualização da webcam.</p> <p>Deixe o aluno adicionar código no arquivo characters_environment.js.</p> <p>Continue com as Atividades Adicionais, se sobrar tempo.</p>
<p>Na aula de hoje começaremos a adicionar código JS para o jogo IA Mario. Antes disso, deixe-me dar um resumo do arquivo JS em que estaremos adicionando código:</p> <ol style="list-style-type: none"> 1. Arquivo main.js: este é o primeiro arquivo JS importante escrito por Lunik e possui códigos para carregar as imagens necessárias para o jogo IA Mario e o código para iniciar o jogo. Neste arquivo estaremos adicionando código para o modelo poseNet e para acessar a webcam. 2. Arquivo characters_environment: Este arquivo também é escrito por Lunik e tem código para controlar o jogo usando as teclas do teclado. Neste arquivo estaremos adicionando código para acusar as coordenadas x e y do nariz e a função <code>startGame()</code>. Também queremos controlar o Mario usando o movimento do nariz, então substituiremos o código para controlar Mario com teclas do teclado pelo nosso código de controle usando as coordenadas x e y do nariz. Mas isso será feito na próxima aula. <p>Em main.js, parte do código já foi feito para você.</p>		

OBSERVAÇÃO: Todas as imagens abaixo estão presentes na **Atividade da Professora 2** para representação visual. Por favor, abra a **Atividade da Professora 2** durante a explicação.

```
function preload() {  
  world_start = loadSound("world_start.wav");  
  setSprites();  
  MarioAnimation();  
}  
  
function setup() {  
  canvas = createCanvas(1240,336);  
  instializeInSetup(mario);  
}  
  
function draw() {  
  game()  
}
```

Que é:

1. O código para carregar o arquivo de áudio da música de início do jogo AI Mario.
2. A função **setSprites()**, usada para carregar todas as imagens usadas no jogo. Você não precisa se preocupar com essa função, pois ela já foi pré-escrita por Lunik.
3. A função **MarioAnimation()** carregará toda a animação necessária para o jogo AI Mario. Você não precisa se preocupar com essa função, pois ela já foi pré-escrita por Lunik.
4. Depois, há o código para criar a tela.
5. Também há a função **instializeInSetup()** que carregará todo o código necessário para o jogo AI Mario. Você não precisa se preocupar com essa função, pois ela já foi pré-escrita por Lunik.
6. Depois, há a função **game()**, que iniciará o jogo AI Mario. Você não precisa se preocupar com essa função, pois ela já foi pré-escrita por Lunik.

Vamos começar adicionando o código JS no arquivo **main.js**.

1. Primeiro, adicionaremos o código para acessar a webcam e definir o tamanho de sua visualização. Como no protótipo do jogo IA Mario, já programamos para acessar a webcam e definir o tamanho para ela na aula passada, copie de lá e cole na função **setup()**.

OBSERVAÇÃO: O protótipo do jogo IA Mario está presente na **Atividade do Aluno 1**, então quando o aluno estiver fazendo a atividade, peça para ele copiar o código da **Atividade do Aluno 1** e colá-lo na função `setup()`.

Copie daqui

```
function setup() {  
  createCanvas(650, 400);  
  video = createCapture(VIDEO);  
  video.size(600, 300);  
  
  poseNet = ml5.poseNet(video, modelLoaded);  
  poseNet.on('pose', gotPoses);  
}
```

Cole aqui

```
function preload() {  
  world_start = loadSound("world_start.wav");  
  setSprites();  
  MarioAnimation();  
}  
  
function setup() {  
  canvas = createCanvas(1240, 336);  
  canvas.parent('canvas');  
  
  instializeInSetup(mario);  
  
  video = createCapture(VIDEO);  
  video.size(600, 300);  
}  
  
function draw() {  
  game()  
}
```

- Atualizaremos a largura e a altura da visualização webcam de 600 e 300 para 800 e 400, respectivamente.

```
function preload() {
  world_start = loadSound("world_start.wav");
  setSprites();
  MarioAnimation();
}

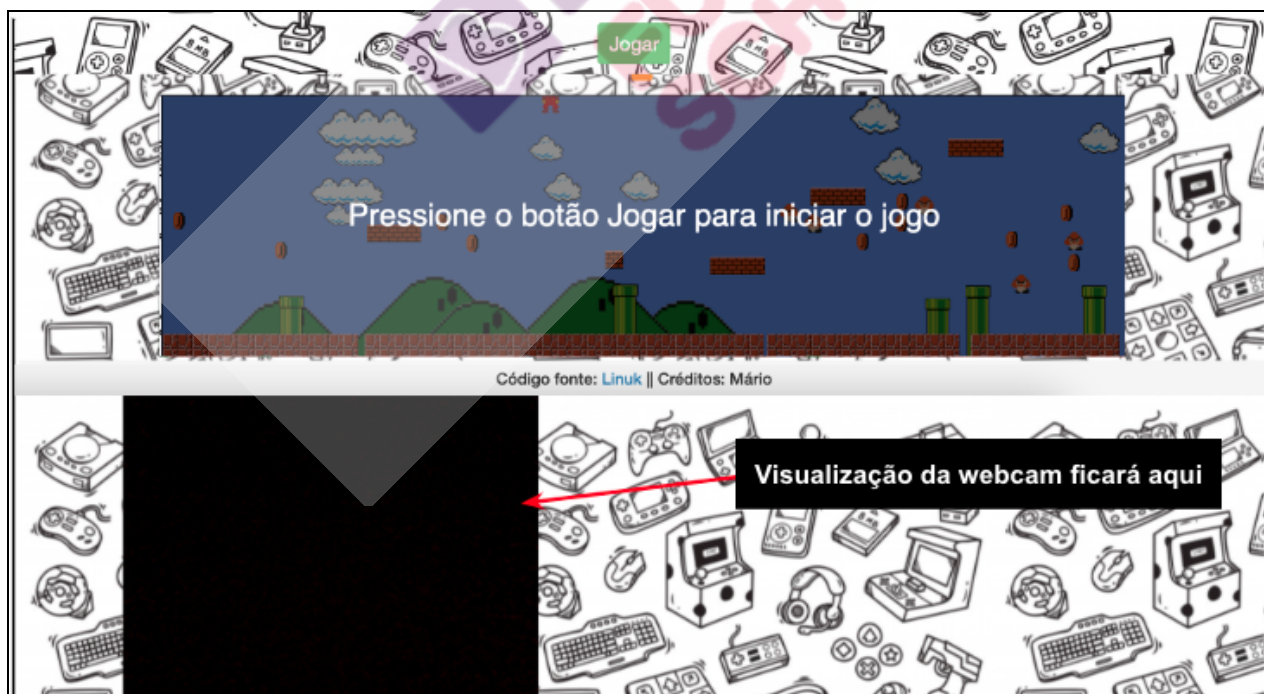
function setup() {
  canvas = createCanvas(1240,336);
  canvas.parent('canvas');

  instializeInSetup(mario);

  video = createCapture(VIDEO);
  video.size(800,400);
}

function draw() {
  game()
}
```

Resultado:



Lembre-se que na aula nº 138 criamos uma div para manter a visualização da webcam, então agora vamos escrever o código para colocar essa visualização dentro dessa div.

P - Qual função podemos usar para isso, a função **parent()** ou função **child()**?

R - **parent()** é uma função p5.js usada para colocar a visualização da webcam ou qualquer componente p5.js como tela dentro de uma div HTML.

Sintaxe: **variable_holding_component_p5_component.parent("HTML ID")**

Aqui queremos colocar a visualização da webcam dentro da div HTML, então o código será:

```
function setup() {  
  canvas = createCanvas(1240,336);  
  canvas.parent('canvas');  
  
  instializeInSetup(mario);  
  
  video = createCapture(VIDEO);  
  video.size(800,400);  
  video.parent('game_console');
```

- A variável de vídeo mantém a visualização da webcam, é por isso que **video.**
- Em seguida, a função p5.js **video.parent**
- Em seguida, o ID HTML da div que criamos para manter a visualização da webcam **video.parent('game_console');**

Como você vê, a div HTML que definimos para manter uma visualização da webcam está na tag center.

OBSERVAÇÃO: Todas as imagens abaixo estão presentes na **Atividade da Professora 2** para representação visual. Por favor, abra a **Atividade da Professora 2** durante a explicação.

```
<body background="background.jpg">
  <center>
    <div class="btn btn-primary heading">
      <h3>IA MÁRIO</h3>
      <button class="btn btn-primary" data-toggle="modal" data-target="#myModal">Instruções</button>
    </div>
    
    <br><br>
    <button class="btn btn-success" onclick="startGame()" id="start">Jogar</button>
    <br><br>
    <h3 id="status" class="btn btn-warning"></h3>
    <br><br>
    <div id="canvas"></div>
    <div id="game_console"></div>
    <h4>Código fonte: <a href="https://github.com/linuk">Linuk</a> || Créditos: Mário</h4>

    <div id="myModal" class="modal fade ">
      <div class="modal-dialog" >
        <!-- Modal content-->
        <div class="modal-content">

          <div class="modal-header">
            <button class="close" data-dismiss="modal">&times;</button>
            <h4>Instruções</h4>
          </div>

          <div class="modal-body">
            
            
            
          </div>

        </div>
      </div>
    </div>
  </center>
```

- E qualquer elemento que esteja na tag center virá no centro da página da web. Isso significa que **esta div** (definida para manter uma visualização da webcam) virá no centro da página da web.
- Armazenamos uma visualização da webcam **nesta div** (que é definida para manter a visualização da webcam). Como resultado, essa visualização da webcam também ficará no centro da página da web.

Resultado:



2. Agora vamos adicionar um pouco de CSS para esta visualização da webcam em [style.css](#). Para adicionar estilo à webcam de p5.js, podemos escrever `video {}` e depois escrever a propriedade style dentro dele.

```
video
{
  background: url('game_console.png');
  background-size: cover;
  background-position: center;
  padding: 25px;
}
```

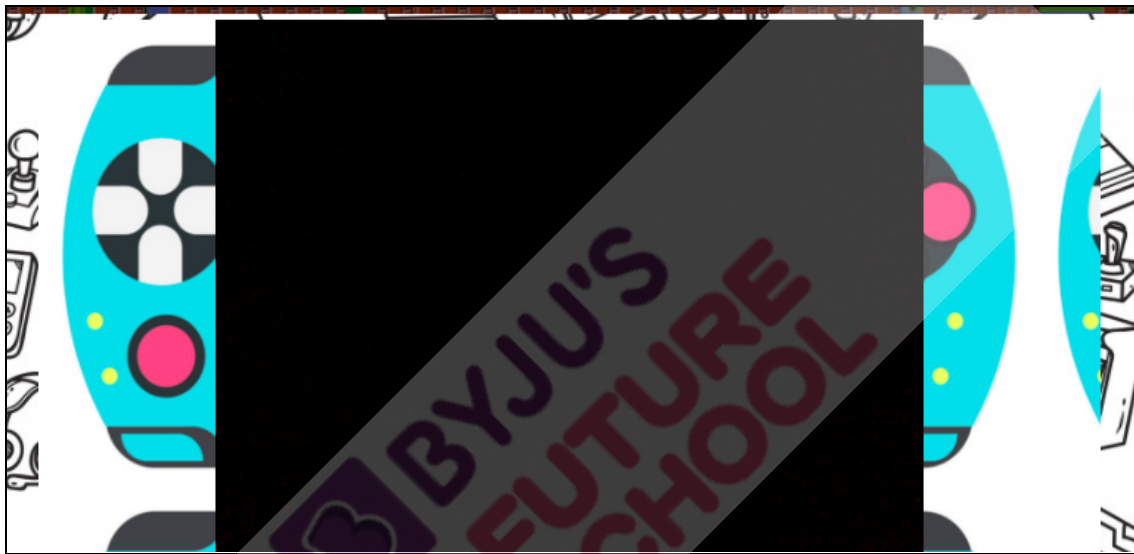
- Temos que definir a imagem game_console no fundo desta visualização da webcam. Para definir a imagem de fundo do CSS, a **propriedade background-image** é usada. A **propriedade background-image** define a imagem de fundo para um elemento. Sintaxe: **background-image: url('image_name');**

```
background: url('game_console.png');
```

- Dentro da URL mencione o nome da imagem que é **game_console.png**. Coloque o mesmo nome, pois esta imagem está presente na pasta **Jogo-Mario**.

OBSERVAÇÃO: Todas as imagens abaixo estão presentes na **Atividade da Professora 2** para representação visual. Por favor, abra a **Atividade da Professora 2** durante a explicação.

Resultado:



- Como você pode ver, a imagem de fundo é repetida, pois é muito pequena, e não cobre a tela inteira. Portanto, queremos que uma única imagem cubra completamente o plano de fundo.

P - E qual propriedade de estilo vamos usar para conseguir isso?

R - `background-size:cover;`

A propriedade `background-size` especifica o tamanho da imagem de fundo. Ela redimensiona a imagem de fundo, conforme o CSS que mencionamos.

Mencionaremos um CSS `cover` e daremos `background-size`. Então será `background-size:cover`.

cover: Redimensiona a imagem de fundo para cobrir toda a página da web, se a imagem for menor que o tamanho da tela, ela alongará a imagem, se a imagem for maior que o tamanho da tela, aparará as bordas e garantirá que a imagem se encaixe em toda a tela.

```
video
{
  background: url('game_console.png');
  background-size: cover;
}
```

Resultado:



- Como você pode ver, a imagem não está centralizada.

P - Então, para obter a imagem de fundo no centro da página, qual propriedade de estilo usaremos?

R - A propriedade de CSS, `background-position: center;`.

```
video
{
  background: url('game_console.png');
  background-size: cover;
  background-position: center;
}
```

Resultado:



- Agora vamos adicionar algum preenchimento para que a visualização da webcam se encaixe corretamente na imagem do console de jogos.

```
video
{
  background: url('game_console.png');
  background-size: cover;
  background-position: center;
  padding: 25px;
}
```

Resultado:



3. Em seguida, adicionaremos o código para:

- Inicializar o modelo poseNet;
- A função **modelLoaded()**;
- Para executar o modelo poseNet;
- Para a função **gotResult()** e buscar das coordenadas x e y.

Como no protótipo do jogo IA Mario, já programamos todo o código que acabei de mencionar. Então copie de lá e cole na função **setup()**.

OBSERVAÇÃO: O protótipo do jogo IA Mario está presente na [Atividade do Aluno 1](#), então quando o aluno estiver fazendo a atividade, peça para ele copiar o código da [Atividade do Aluno 1](#) e colá-lo no arquivo [main.js](#).

Copie daqui

```
function setup() {
  createCanvas(650, 400);
  video = createCapture(VIDEO);
  video.size(600, 300);

  poseNet = ml5.poseNet(video, modelLoaded);
  poseNet.on('pose', gotPoses);
}

function modelLoaded() {
  console.log('Model Loaded!');
}

function gotPoses(results)
{
  if(results.length > 0)
  {
    noseX = results[0].pose.nose.x;
    noseY = results[0].pose.nose.y;
    console.log("noseX = " + noseX + ", noseY = " + noseY);
  }
}
```

Cole aqui

```
function setup() {
  canvas = createCanvas(1240, 336);
  canvas.parent('canvas');

  initializeInSetup(mario);

  video = createCapture(VIDEO);
  video.size(800, 400);
  video.parent('game_console');

  poseNet = ml5.poseNet(video, modelLoaded);
  poseNet.on('pose', gotPoses);
}

function modelLoaded() {
  console.log('Model Loaded!');
}

function gotPoses(results)
{
  if(results.length > 0)
  {
    noseX = results[0].pose.nose.x;
    noseY = results[0].pose.nose.y;
    console.log("noseX = " + noseX + ", noseY = " + noseY);
  }
}
```

Vamos fazer algumas mudanças na função **gotResult()**:

- Primeiro remova a exibição das variáveis noseX e noseY da função **gotResult()**.


```
function gotPoses(results)
{
  if(results.length > 0)
  {
    noseX = results[0].pose.nose.x;
    noseY = results[0].pose.nose.y;
  }
}
```

Agora adicione a exibição da matriz results que vem do modelo poseNet.

```
function gotPoses(results)
{
  if(results.length > 0)
  {
    console.log(results);
    noseX = results[0].pose.nose.x;
    noseY = results[0].pose.nose.y;
  }
}
```

Com isso nosso código do arquivo [main.js](#) está completo.

Agora vamos começar a adicionar código no arquivo **characters_environment.js**. Este arquivo foi criado por Lunik.

4. Como você pode ver, no arquivo main.js, existe uma função **game()** dentro da função **draw()**

```
function draw() {
  game();
}
```

- E como sabemos que a função **draw()** é chamada continuamente, significa que a função **game()** também será chamada continuamente, então vamos colocar a exibição de noseX e noseY na função **game()**.
- E a função **game()** é definida no arquivo **characters_environment.js**, este é o segundo arquivo JS principal do jogo AI Mario. Existem dois propósitos para isso:

- Podemos confirmar que o código de busca das coordenadas x e y está correto.
- Se a exibição de noseX e noseY funcionar, isso nos dará uma confirmação de que essas variáveis podem ser acessadas com sucesso no arquivo **characters_environment.js**. A importância de acessar essas variáveis no arquivo **characters_environment.js** é:
 - O arquivo **characters_environment.js** contém código para controlar o Mario através das teclas do teclado, mas queremos controlar com os movimentos do nariz. Portanto, é muito importante que possamos acessar as variáveis noseX e noseY dentro do arquivo **characters_environment.js**.

Em **characters_environment.js** existe um código pré-escrito, tenha cuidado ao adicionar código neste arquivo.

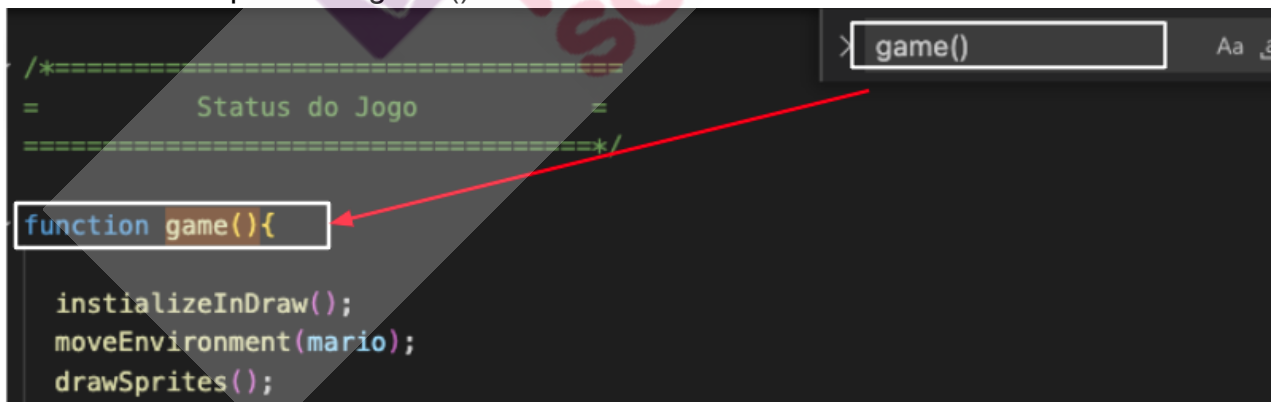
Para conseguir “exibir noseX e o noseY”, primeiro pesquisaremos “game()” no arquivo **characters_environment.js**, pois será difícil encontrar a função game().

Essa é uma dica quando lidamos com códigos grandes, estaremos procurando a coisa necessária, isso vai ajudar muito.

Para pesquisar qualquer coisa no código VS:

- Usuário Mac: command + F
- Usuário Windows: ctrl + F

5. Primeiro vamos procurar “game()”:



The screenshot shows a code editor with a search bar at the top right containing the text "game()". Below the search bar, the code editor displays the following code:

```
/*=====
=      Status do Jogo      =
=====*/

function game(){

    instializeInDraw();
    moveEnvironment(mario);
    drawSprites();
}
```

A red arrow points from the search bar to the "function game(){" line in the code.

6. Antes da função **game()**, defina 2 variáveis **noseX** e **noseY** e defina o valor como vazio.

```
/*=====
=           Status do Jogo           =
=====*/
noseX = "";
noseY = "";
GameStatus = "";

function game(){
```

7. Em seguida, na função **game()**, escreva o código para consolar o **noseX** e o **noseY**.

```
function game(){

    console.log("noseX = " + noseX + " ,noseY = " + noseY);
```

8. Defina uma variável vazia para manter o status do jogo. Mais para frente no código, usaremos essa variável para iniciar o jogo.

```
noseX = "";
noseY = "";
GameStatus = "";

function game(){

    console.log("noseX = " + noseX + " ,noseY = " + noseY);
```

9. Lembre-se que na aula C138, definimos um botão Jogar, no qual definimos uma função onclick “**startGame()**”. Então, agora precisamos definir a função “**startGame()**” no arquivo [characters_environment.js](#). Defina a função “**startGame()**” logo acima da função **game()**.

```
noseX = "";  
noseY = "";  
GameStatus = "";  
  
function startGame()  
{  
  
}  
  
function game(){  
  
    console.log("noseX = " + noseX + " ,noseY = " + noseY);
```

10. Dentro desta função, atualize a variável GameStatus com “start”.

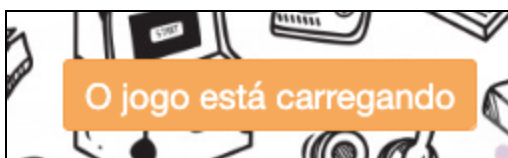
```
noseX = "";  
noseY = "";  
GameStatus = "";  
  
function startGame()  
{  
    GameStatus = "start";  
}  
  
function game(){  
  
    console.log("noseX = " + noseX + " ,noseY = " + noseY);
```

11. Em seguida, atualizaremos o elemento HTML que havíamos definido na aula nº 138 para manter o status do jogo com “O jogo está carregando”.

```
noseX = "";
noseY = "";
GameStatus = "";

function startGame()
{
  GameStatus = "start";
  document.getElementById("status").innerHTML = "O jogo está carregando";
}
```

Quando clicar no botão Jogar, o HTML ficará assim:



OBSERVAÇÃO: Enquanto o aluno cria o código, **certifique-se de que ele faça testes clicando no botão GO LIVE do VS CODE. Isso resulta na execução do arquivo no servidor ao vivo do programa.**



Como estamos usando arquivos de imagem e som, e o p5.js não nos permite executar esses tipos de arquivos a partir de um sistema local, ele precisa ser executado em um **servidor**.

A professora encerra o compartilhamento de tela

Agora é sua vez.

- Peça para o aluno pressionar a tecla ESC para retornar ao painel
- Ajude o aluno a iniciar o compartilhamento de tela
- A professora ativa o modo tela cheia

Passo 3:
Atividade Dirigida
pelo Aluno
(10 min)

1. Você deve iniciar o código JS em **main.js**.
2. Em seguida, adicione um pouco de CSS para visualização da webcam em **style.css**.
3. Depois, adicione código JS em **characters_environment.js**.

Não hospedaremos o jogo IA MARIO nesta aula. Faremos isso quando o código estiver completo. Portanto, mantenha seus arquivos em segurança e não os altere, pois qualquer modificação pode corrompê-los.

Atividade do Aluno 1 -
DIAGRAMA DO CÓDIGO

O aluno deve:

1. Começar o código JS em **main.js**.
2. Depois adicionar código CSS para a visualização da webcam, em **style.css**.
3. Por fim, adicionar código JS em **characters_environment.js**.




Após o aluno ter completado o código JS, conforme as instruções, peça que ele **execute o código, certifique-se de que ele faça o teste, clicando no botão GO LIVE do VS CODE. Isso resulta na execução do arquivo no servidor ao vivo do programa.**



Como estamos usando arquivos de imagem e som, e o p5.js não nos permite executar esses tipos de arquivos a partir de um sistema local, ele precisa ser executado em um **servidor**.

Hospedaremos o jogo IA MARIO apenas quando o código estiver completo.

A professora ajuda o aluno a encerrar o compartilhamento de tela

<p>Passo 4: Encerramento (3 min)</p>	<p>Você se saiu muito bem hoje.</p> <p>Vai receber tiradas de chapéu pelo excelente trabalho!</p> <p>P - Para definir o plano de fundo de CSS, qual propriedade de estilo é usada? E qual é a sintaxe?</p> <p>R - Usamos a propriedade de estilo background-image. Sintaxe: background-image: url('image_name');</p> <p>P - Se quisermos colocar qualquer componente p5.js, como visualização da webcam ou tela, em elementos HTML, qual função p5.js é usada? E qual é a sintaxe?</p> <p>R - Usamos a função parent(). p5.js_component.parent("HTML ID") -</p>	<p>(Dê, no mínimo, 2 tiradas de chapéu)</p> <p>Pressione o ícone de Tirada de Chapéu para Resolvendo Atividades Criativamente.</p>  <p>Pressione o ícone de Tirada de Chapéu para Ótima Pergunta.</p>  <p>Pressione o ícone de Tirada de Chapéu para "Você se Concentrou".</p>  <p>Se não houver tempo para realizar as Atividades Adicionais, peça para o aluno realizá-las após a aula. Essas atividades estão presentes apenas no painel sob Atividade do Aluno.</p> <p>Além disso, lembre o aluno de consultar as Referências para Atividades do Aluno para aumentar seu conhecimento. Isso também deve ser feito após a aula.</p>
<p>Para a solução de todas as Atividades Adicionais, abra a Atividade da Professora - 4 e navegue até o número de aula C141.</p>		

Atividade Adicional 1 -

Execute a **Atividade do Aluno-3** do **painel**

AS **TAREFAS** e **DICAS** são mencionadas no próprio website.

Atividade Adicional 2 -

Execute a **Atividade do Aluno-4** do **painel**

AS **TAREFAS** e **DICAS** são mencionadas no próprio website.

Atividade Adicional 3 -

Execute a **Atividade do Aluno-5** do **painel**

AS **TAREFAS** e **DICAS** são mencionadas no próprio website.

Atividade Adicional 4 -

Execute a **Atividade do Aluno-6** do **painel**

AS **TAREFAS** e **DICAS** são mencionadas no próprio website.

Atividade Adicional 5 -

Execute a **Atividade do Aluno-7** do **painel**

AS **TAREFAS** e **DICAS** são mencionadas no próprio website.

A professora clica em

✕ Terminar Aula

Atividade	Nome da Atividade	Links
Atividade da Professora 1	PROTÓTIPO IA MARIO	https://editor.p5js.org/lucas.diniz/sketches/ROdXRUMyB
Atividade da Professora 2	DIAGRAMA DO CÓDIGO	https://s3-whjr-curriculum-uploads.whjr.online/90db8d53-0839-4897-8336-2206ba154f9d.pdf
Atividade da Professora 3	CÓDIGO FONTE COMPLETO	https://drive.google.com/file/d/1zJaxK8JB8l_HH-FDzc1GiQxXKpCTttD1/view?usp=sharing
Atividade da Professora 4	SOLUÇÃO DAS ATIVIDADES ADICIONAIS	https://docs.google.com/spreadsheets/d/e/2PACX-1vSG0V0IQyGx7Zo7VJgaWAHYDpalldCZ-RepABc_slelct65DaM-R9YF0vCbS5JFdsF3MEFQ-WLslsRc/pubhtml
Atividade do Aluno 1	PROTÓTIPO IA MARIO	https://editor.p5js.org/lucas.diniz/sketches/BiM9McGMH
Atividade do Aluno 2	DIAGRAMA DO CÓDIGO	https://s3-whjr-curriculum-uploads.whjr.online/90db8d53-0839-4897-8336-2206ba154f9d.pdf
Referência para Atividade do Aluno 1	REFERÊNCIA DE IMAGEM DE FUNDO EM CSS	https://www.w3schools.com/cssref/pr_background-image.asp
Referência para Atividade do Aluno 2	REFERÊNCIA DA FUNÇÃO PARENT() EM p5.js	https://p5js.org/reference/#/p5.Element/parent
Solução do Projeto	IA PING PONG - PARTE 4	https://s3-whjr-curriculum-uploads.whjr.online/84b3c7be-4322-4330-9234-4d377df5a998.pdf