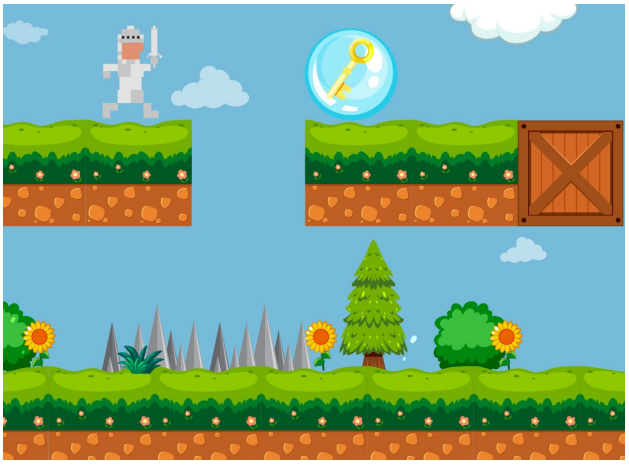
ADV-C141

WhiteHatJr
Live Online Coding for Kids

AI GAME



What is our GOAL for this MODULE?

We will convert a normal Mario Game into an AI Mario Game using a pre-trained model which is posenet.

What did we ACHIEVE in the class TODAY?

- Complete JS code for main.js file
- Start adding JS code for characters_environment.js file

Which CONCEPTS/ CODING did we cover today?

- Add code for accessing webcam, initializing and executing of posenet modal.
- Add code for accessing noseX and noseY in characters_environment.js file
- Add code for startGame() function

How did we DO the activities?

1. First we will add code for accessing the webcam and setting the size of the webcam live view.
   As in the Al Mario Game prototype we have already coded for accessing the webcam and setting the size for it in the previous class.
   So copy from there and paste it inside the **setup()** function.

Copy from here

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

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

Paste it here

```
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()
}
```

We will update the width and height of the webcam live view from 600 and 300 to 800 and 400 respectively.

```
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()
}
```

Output-

Remember in class no - 138 we had created a div to hold the webcam live view, so now we will write code to put the webcam live view inside that div.

parent() is a p5.js function used to put webcam live view or any p5.js component like canvas inside a HTML div.
Syntax - **variable_holding_component_p5_component.parent("HTML ID")** -

Here we want to put webcam live view inside HTML div then the code will be -

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

    instializeInSetup(mario);

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

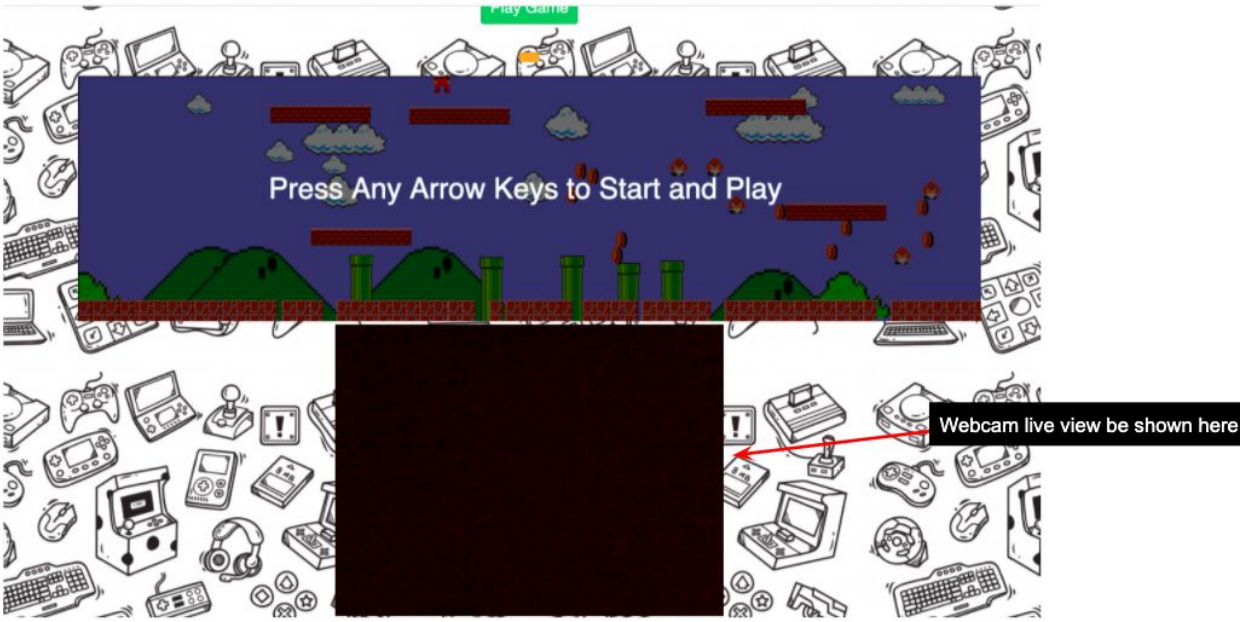
- Video variable hold the webcam live view that's why `video.`
- Then p5.js function `video.parent`
- Then the HTML ID of the div which we had created to hold the webcam live view `video.parent('game_console');`

As you see the HTML div which we defined for holding webcam live view is inside the center tag

```
<body background="background.jpg">
  <center>
    <div class="btn btn-primary heading">
      <h3>AI MARIO GAME </h3>
      <button class="btn btn-primary"  data-toggle="modal" data-target="#myModal">Instructions</button>
    </div>
    <img src="mario.jpg" class="big_image">
    <br><br>
    <button class="btn btn-success" onclick="startGame()" id="start">Play Game</button>
    <br><br>
    <h3 id="status" class="btn btn-warning"></h3>
    <br><br>
    <div id="canvas"></div>
    <div id="game_console"></div>
    <h4>Source Code : <a href="https://github.com/linuk">Linuk</a> ||  Game Credit : Mario</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>Instructions</h4>
          </div>

          <div class="modal-body">
              <img src="jump.png" class="img-responsive instruction_image">
              <img src="right.png" class="img-responsive instruction_image">
              <img src="left.png" class="img-responsive instruction_image">
            </div>

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

And any element which is inside the center tag will come at the center of the webpage - This means **this div**(which is defined for holding a webcam live view) will come at the center of the webpage.
And as we have stored a webcam live view in **this div**(which is defined for holding the webcam live view) - so as a result this webcam live view will also come at the center of the web page.

Output -



Now we will add some CSS for this webcam live view in **style.css**

For adding style to webcam live of p5.js, we can write **video { }** and then write the style property inside it

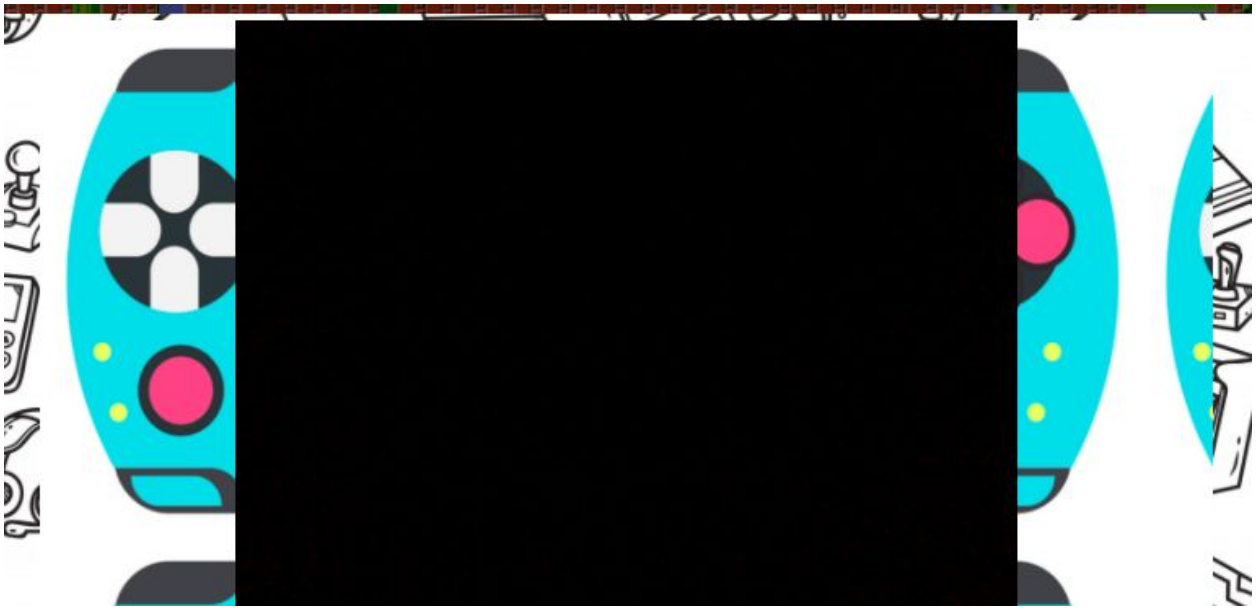
```
video
{
    background: url('game_console.png');
    background-size:cover;
    background-position: center;
    padding: 25px;
}
```

- We have to set the game_console image in the background of this webcam live view.
  To set background image from CSS **background-image property** is used. The **background-image property** sets the background image for an element.
  Syntax - **background-image: url('image_name');**

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

Inside URL mention the image name which is **game_console.png**, put the same image name because this image is present in the **Mario-Game** folder

Output -



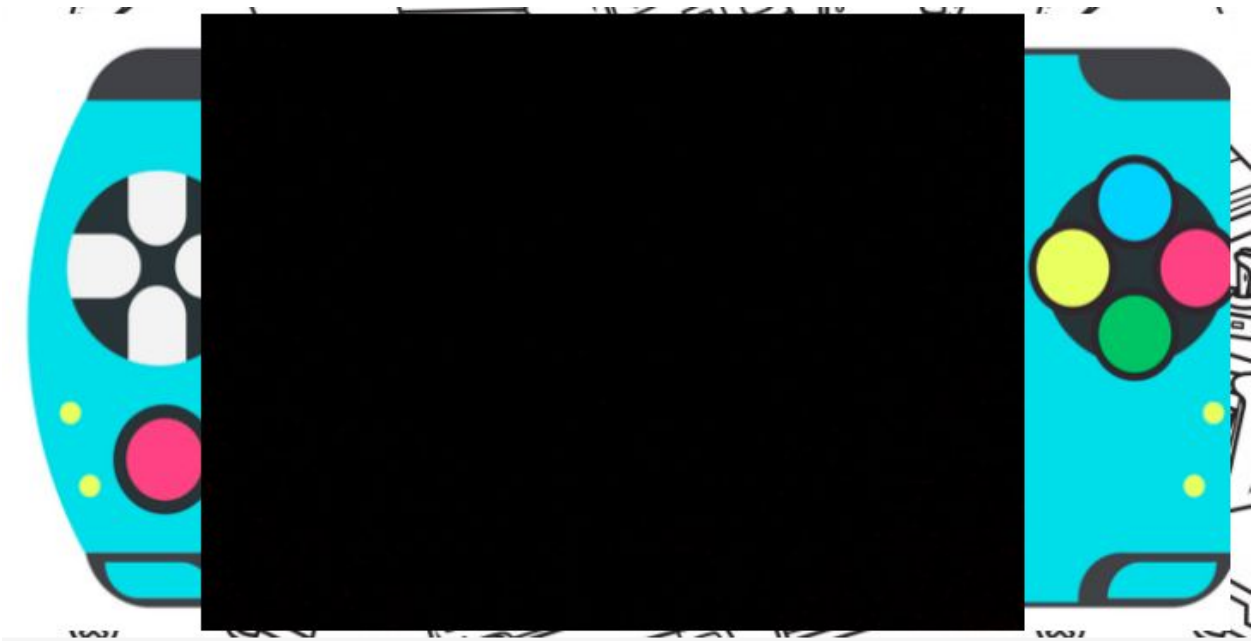
- As you see the background image is repeated since the image is too small therefore it cannot cover the complete screen. So we want a single image to completely cover the background.

  background-size:cover;.
  The background-size property specifies the size of the background image. It resizes the background image, as per the CSS we mention.
  We will mention a CSS **cover**, and give it to background-size.So it will be background-size:cover.

  cover **- Resize the background image to cover the entire webpage, if the image is smaller than the size of the screen it will stretch the image, if the image is larger than the size of the screen size it would trim the edges and make sure that the image fits on to the entire screen.**

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

Output



- As you can see the image is not centered
  To get the background image in the center of the page we will use CSS property background-position:center;.

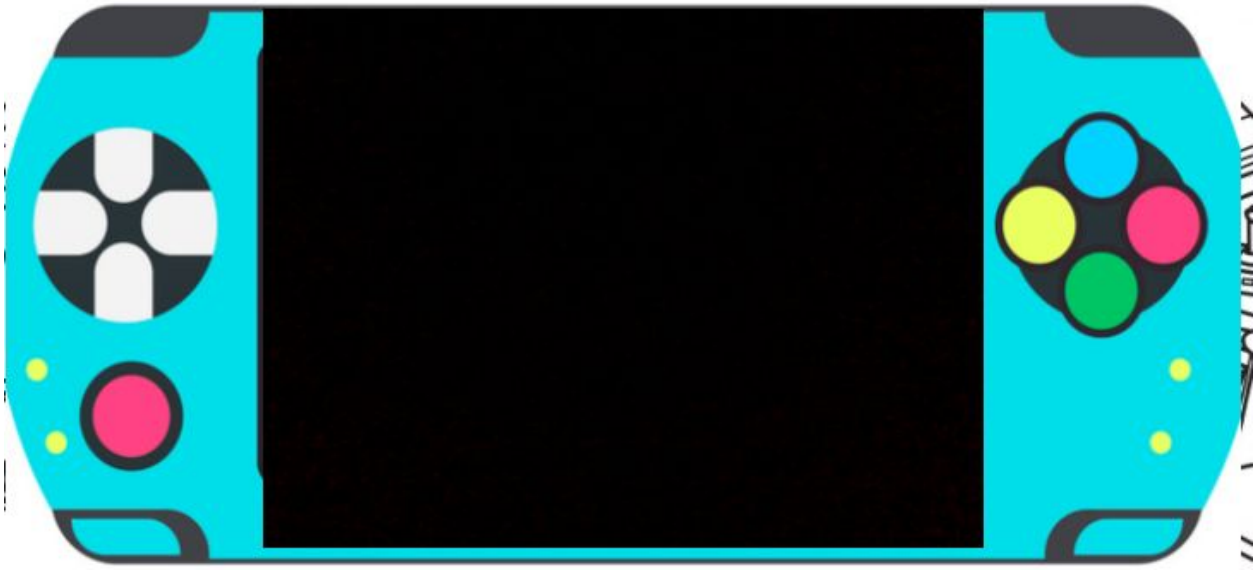
```
video
{
    background: url('game_console.png');
    background-size:cover;
    background-position: center;
}
```

Output -

- Now we will add some padding to get the webcam live view properly fit in the game console image

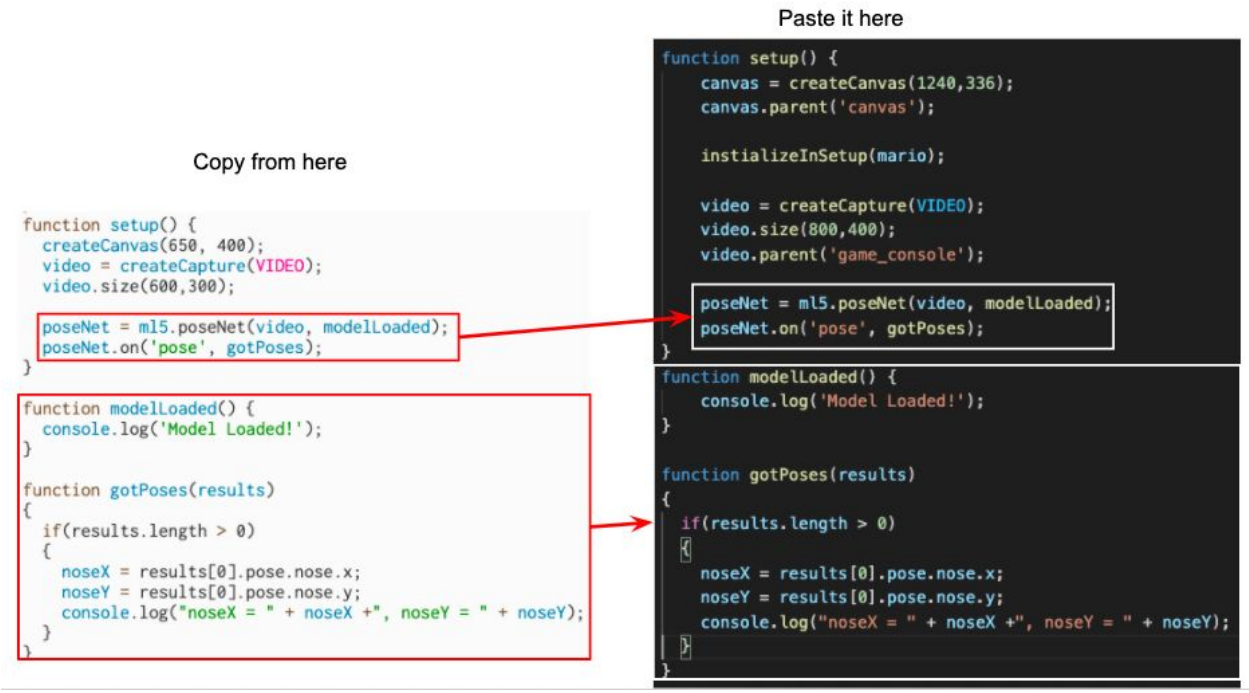
```
video
{
    background: url('game_console.png');
    background-size:cover;
    background-position: center;
    padding: 25px;
}
```

Output -



2. Then we will add code for Added code for -
    - Initializing posenet model
    - Code for **modalLoaded()** function
    - Code for executing posenet model
    - Code for **gotResult()** function, and fetching x and y coordinates.

    As in the AI Mario Game prototype we have already coded for all of the code which I just mentioned.
    So copy from there and paste it inside the **setup()** function.



Now we will do some changes in **gotResult()** function -

- First remove the consoling of noseX and noseY variable from **gotResult()** function

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

- Now add consoling of the results array which is coming from the posenet model.

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

With this our code for main.js file is completed.
Now we will start adding code in **characters_environment.js** file. This file is created by lunik.

3. As you see there is a **game()** function inside **draw()** function.

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

And as we know **draw()** function is called continuously, means **game()** function will also be called continuously, so we will put consoling of noseX and noseY inside **game()** function.
And **game()** function is defined in **characters_environment.js** file- this is the **second** main JS file of the mario game
There are two purposes of doing this -

a. We can confirm that the code of fetching x and y coordinate is correct
b. If consoling of noseX and noseY works, this will give us a confirmation that variables noseX and noseY can be successfully accessed in **characters_environment.js** file.

   Importance of accessing these variables noseX and noseY inside **characters_environment.js** file is -
   characters_environment.js file contains code for controlling mario via keyboard keys but we want to control the mario by nose movements. Therefore it is very important that we can access noseX and noseY variables inside **characters_environment.js** file

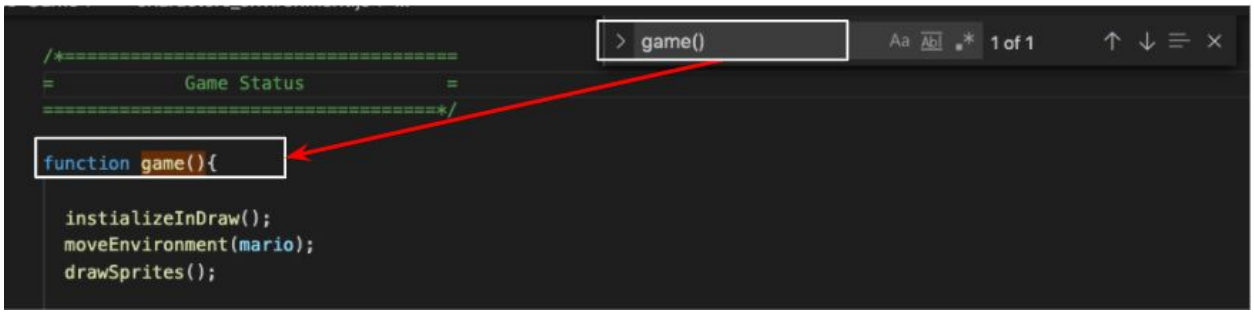
In **characters_environment.js** some prewritten code is written, be careful while adding code in this file.

To achieve “consoling of noseX and noseY ”, first we will search “game()” in

characters_environment.js file, because it will be difficult to find game() function. This is a tip when we deal with big cods, we will be searching for the required thing, this will help a lot.
To search anything on VS code -

- Mac users - command + F
- Windows user - ctrl + F

- First we will search “game()”



- Before the **game()** function, define 2 variables noseX and noseY, and set the value as empty.

- Then inside the **game()** function write code for consoling noseX and noseY.

4. Define an empty variable for holding the game status. Later in the code we will use this variable to start the game.

5. Remember in class C138, we had defined a play button, in which we had defined a onclick function "**startGame()**" function. So now we need to define the "**startGame()**" function inside **characters_environment.js** file. Define "**startGame()**" function just above the **game()** function.

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

function startGame()
{

}

function game(){

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

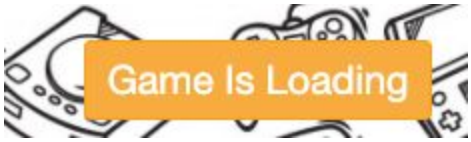
- Inside this function, update the GameStatus variable with "start"

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

function startGame()
{
  GameStatus = "start";
}

function game(){

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

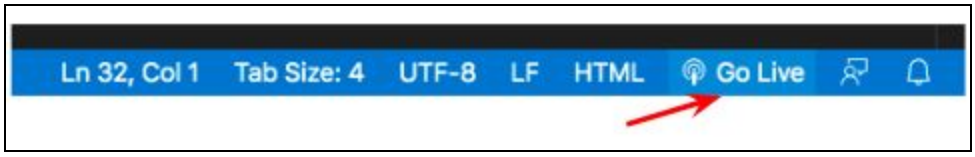
- Then we will update the HTML element which we had defined in class no.138 for holding the status of the game with "Game Is Loading"

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

function startGame()
{
  GameStatus = "start";
  document.getElementById("status").innerHTML = "Game Is Loading";
}

function game(){

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

When the button is click play button the HTML will look like this -



NOTE -
Whenever you are running the code, MAKE SURE TO TEST BY CLICKING ON GO LIVE BUTTON OF VISUAL STUDIO. THIS WILL RESULT IN RUNNING THE FILE ON THE LIVE SERVER



OF VISUAL STUDIO.

Because we are using an images and audio file, and p5.js just doesn't allow us to run any images and audio files from a local system, it needs to be run on a server.

What's NEXT?
We will continue building this AI Game.