Project 3 : Mimic Me!

In this project, I learned to track faces in a video and identify facial expressions using Affectiva. As a fun visualization, I tagged each face with an appropriate emoji next to it. I then turned this into a game where the player needs to mimic a random emoji displayed by the computer!

Task 1 : Display Feature Points

I displayed the feature points on top of the webcam image that are returned along with the metrics. First, the fill style of feature points is set to blue. Then, for each feature point, I used x and y coordinate of feature point to determine the location and set radius to 2.

```
##########################################################################
##

// Draw the detected facial feature points on the image
function drawFeaturePoints(canvas, img, face) {
  // Obtain a 2D context object to draw on the canvas
  var ctx = canvas.getContext('2d');

  // TODO: Set the stroke and/or fill style you want for each feature
point marker
  // See: https://developer.mozilla.org/en-US/docs/Web/API/
CanvasRenderingContext2D#Fill_and_stroke_styles
  // <your code here>
  ctx.fillStyle = 'blue';

  // Loop over each feature point in the face
  for (var id in face.featurePoints) {
    var featurePoint = face.featurePoints[id];

    // TODO: Draw feature point, e.g. as a circle using ctx.arc()
    // See: https://developer.mozilla.org/en-US/docs/Web/API/
CanvasRenderingContext2D/arc
    // <your code here>
    ctx.beginPath();
    ctx.arc(featurePoint.x, featurePoint.y, 2, 0, 2 * Math.PI);
    ctx.fill();
  }
}

##########################################################################
##
```

Task 2 : Show Dominant Emoji

In addition to feature points and metrics that capture facial expressions and emotions, the Affectiva API also reports back what emoji best represents the current emotional state of a face. This is referred to as the dominant emoji.

In mimic.js, I implemented the drawEmoji() function to display this emoji on top of the webcam feed, tracking the user's face.

First I set the font for drawing the emoji with the function ctx.font. Then I picked a particular feature point as an anchor for emoji. The dominant emoji is determined from emojis.dominantEmoji and drawn from fillText.

```
##########################################################################
##

// Draw the dominant emoji on the image
function drawEmoji(canvas, img, face) {
  // Obtain a 2D context object to draw on the canvas
  var ctx = canvas.getContext('2d');

  // TODO: Set the font and style you want for the emoji
  // <your code here>
  ctx.font = '50px serif';
  // TODO: Draw it using ctx.strokeText() or fillText()
  // See: https://developer.mozilla.org/en-US/docs/Web/API/
CanvasRenderingContext2D/fillText
  // TIP: Pick a particular feature point as an anchor so that the
emoji sticks to your face
  // <your code here>

  aux = face.emojis.dominantEmoji;
  ctx.fillText(aux, face.featurePoints[4].x, face.featurePoints[4].y);
}

##########################################################################
##
```

Task 3 : Implement Mimic Me!

I implemented the game mechanics in the following modules.

1) mimicEmojiInitialize() to initialize the game
        1: Initialize the audio track when the player mimics the correct emoji
        2: Wait 6s to initialize
        3: Define ScoreCorrect to record the correctly recognized the emoji, set to 0
        4: Define ScoreTotal to record the total amount of emojis displayed, set to 0

```
        5: Define timeleft to record the time left to guess an emoji,
set to 10. Then start the timer to control the time left.
        6: Define the targetEmoji, set to 0
        7: Display a new random emoji

##########################################################################
##

function mimicEmojiInitialize(){

  console.log('initializing...')
  wait(2000);        // Wait 2 seconds to initialize

  init_Score();  //initialize the score

  var timeleft = 10;  // timeleft is the amount of seconds left to the
player to guess an emoji.
  timer = setInterval(timeEnd, 1000); // Set timer to control the time
left to guess an emoji.

  var TargetEmoji = 0; //initialize target emoji to 0
  displayNewEmoji();  // Display a new random emoji
}

##########################################################################
##

2) mimicEmoji() to run the game, the function keeps evaluating the
dominate emoji from the face and check if it matches the target. If
so, play the audio and add 1 to the SocreCorrect (and display the
score). Then reset the timer and display the new random emoji.

##########################################################################
##

function mimicEmoji(face) {

  if (toUnicode(face.emojis.dominantEmoji) == TargetEmoji){
    console.log('Correct face emoji');

    incrementScore();

    if(timer){
      clearInterval(timer);  // Stop the timer
    }
    timeleft = 10;  // Restart the timer
    timer = setInterval(timeEnd, 1000)
    displayNewEmoji();  // Display a new random emoji
  }
}
```

```
#########################################################################
##

3) displayEmoji(): choose a random emoji from a list of emojis, use
setTargetEmoji() to display the chosen emoji, add 1 to ScoreCorrect
using setScore

see comment for detail.

#########################################################################
##

function displayNewEmoji(){
  random = Math.floor(Math.random()*(12+1));  // Generate random emoji
  TargetEmoji = emojis[random];               // Save TargetEmoji to
compare with DominantEmoji
  setTargetEmoji(TargetEmoji);                // Display the random
emoji - target to the player
  total++;                                 // Total Score plus one
  setScore(score,total)           // display the new Total score
}

#########################################################################
##

4) Auxiliary functions:

init_Score(): initialize the score when starting, stopping or
restarting each new game;
incrementScore(): add 1 to score when correctly making facial
expressions that matches target emoji
wait(): wait the amount of time in miliseconds;
timeEnd(): monitors the time left in miliseconds, whenever it reaches
0, display the new emoji and refresh the time left;

#########################################################################
##

function init_Score() {
  score = 0;
  total = 0;
  setScore(score, total);
}

function incrementScore() {
  score++;
  setScore(score, total);
}
```

```
function wait(delay) {
  var start = new Date().getTime();
  while (new Date().getTime() < start + delay);
}

function timeEnd() {
  timeleft--;
  if(timeleft == 0){
    timeleft = 10;
    displayNewEmoji();
  }
}

##########################################################################
##

5) function gameRestart() is called when "Reset" button is pressed, it
displays the score, resets the time left and display a new emoji;
function gameStop() is called when "stop" button is pressed, it resets
the timer and display the score.

##########################################################################
##

function gameRestart(){

  init_Score(); // Display the score

  if(timer){
    clearInterval(timer);
  }
  var timeleft = 10;
  timer = setInterval(timeEnd, 1000);

  var TargetEmoji = 0;
  displayNewEmoji();  // Display a new random emoji
}

function gameStop(){
  if(timer){
    clearInterval(timer);
  }

  init_Score(); // Display the score
}

##########################################################################
##
```