Mimic Me! report

Artem Larionov

7/1/2017

Mimic Me! application

Introduction

`Mimic Me!` sets a goal for a player to mimic an emotion and uses an Affectiva's API[1] to recognize a dominant emotion of the player to check the result.

The game has 20 rounds after which it will be stopped. Every round is a 20 second time frame during which the player should successfully mimic the target emotion. If the player hasn't managed to mimic the target emotion during the round, next round is started and new emotion is set.

Changes in the base code

Base repository

The base repository[2] consists of a server and a skeleton of web client for the game.

Drawing feature points

This function draws feature points recieved from Affectiva's API on a canvas with a video stream of the player.

```
function drawFeaturePoints(canvas, img, face) {
  var ctx = canvas.getContext('2d');

  ctx.strokeStyle = "#000000";

  for (var id in face.featurePoints) {
    var featurePoint = face.featurePoints[id];
    ctx.beginPath();
    ctx.arc(featurePoint.x, featurePoint.y, 2, 0, 2 * Math.PI);
    ctx.stroke();
  }
}
```

Drawing a dominant emotion

This function draws an emoji based on a dominant emotion on a canvas with the video stream of the player. The dominant emotion is recieved from Affectiva's API. The emoji is positioned relatively to one of feature point to make it move with the face of the player.

[1] Affectiva's Emotion-as-a-Service API
[2] Base repository

```
function drawEmoji(canvas, img, face) {
  var ctx = canvas.getContext('2d');

  ctx.strokeStyle = "#ffff00";
  ctx.font = '50px serif';

  ctx.fillText(face.emojis.dominantEmoji, face.featurePoints[5].x - 100, face.featurePoints[5].y);
}
```

Setting a target emoji

These functions provide a random emoji from a set of emojis Affectiva's API detects and set it as a target emoji for the game.

```
function randomEmojiCode() {
  return emojis[Math.floor(Math.random() * emojis.length)];
}

function nextTargetEmoji() {
  emojis_total += 1;
  target_emoji_code = randomEmojiCode();
  setTargetEmoji(target_emoji_code);
}
```

Changes to make the game work

During the initialization these variables are introduce to keep the state of the game:

```
let target_emoji_code;  // contains a code for a target emoji
let emojis_total = 0;   // the total number of emojis/rounds have passed
let emojis_correct = 0; // the number of emojis, the player mimicked
let start_ts = 0;       // the timestamp when the latest round has started

const max_emojis = 20;  // number of rounds for the game
const try_time = 25;    // time frame, during which the player has to mimic the emotion
```

Also some changes were introduced into the Event Listener:

```
detector.addEventListener("onImageResultsSuccess", function(faces, image, timestamp) {
  if (emojis_total < max_emojis) {
    if (timestamp - start_ts > try_time) {
      start_ts = timestamp;
      return nextTargetEmoji();
    }

    var canvas = $('#face_video_canvas')[0];

    if (!canvas) return;

    $('#results').html("");
    log('#results', "Timestamp: " + timestamp.toFixed(2));
    log('#results', "Number of faces found: " + faces.length);
    if (faces.length > 0) {
      log('#results', "Appearance: " + JSON.stringify(faces[0].appearance));
      log('#results', "Emotions: " + JSON.stringify(faces[0].emotions, function(key, val) {
```

```
        return val.toFixed ? Number(val.toFixed(0)) : val;
      }));
      log('#results', "Expressions: " + JSON.stringify(faces[0].expressions, function(key, val) {
        return val.toFixed ? Number(val.toFixed(0)) : val;
      }));
      log('#results', "Emoji: " + faces[0].emojis.dominantEmoji);

      drawFeaturePoints(canvas, image, faces[0]);
      drawEmoji(canvas, image, faces[0]);

      const current_code = toUnicode(faces[0].emojis.dominantEmoji);

      if (current_code == target_emoji_code) {
        emojis_correct += 1;
        start_ts = timestamp;
        return nextTargetEmoji();
      }

      setScore(emojis_correct, emojis_total);
    }
  }
  else {
    setScore(emojis_correct, emojis_total);
    onStop();
  }
});
```