Tongue-computer Interface for the browser

This project is for Prof. Jacob's OOP class focusing on non-WIMP("windows, icons, menus, pointer") user interfaces, by Woodbury Shortridge (woodbury.shortridge@tufts.edu) and Jose Lemus (Jose.Lemus@tufts.edu). We worked in collaboration on both the concept and the feasibility testing.

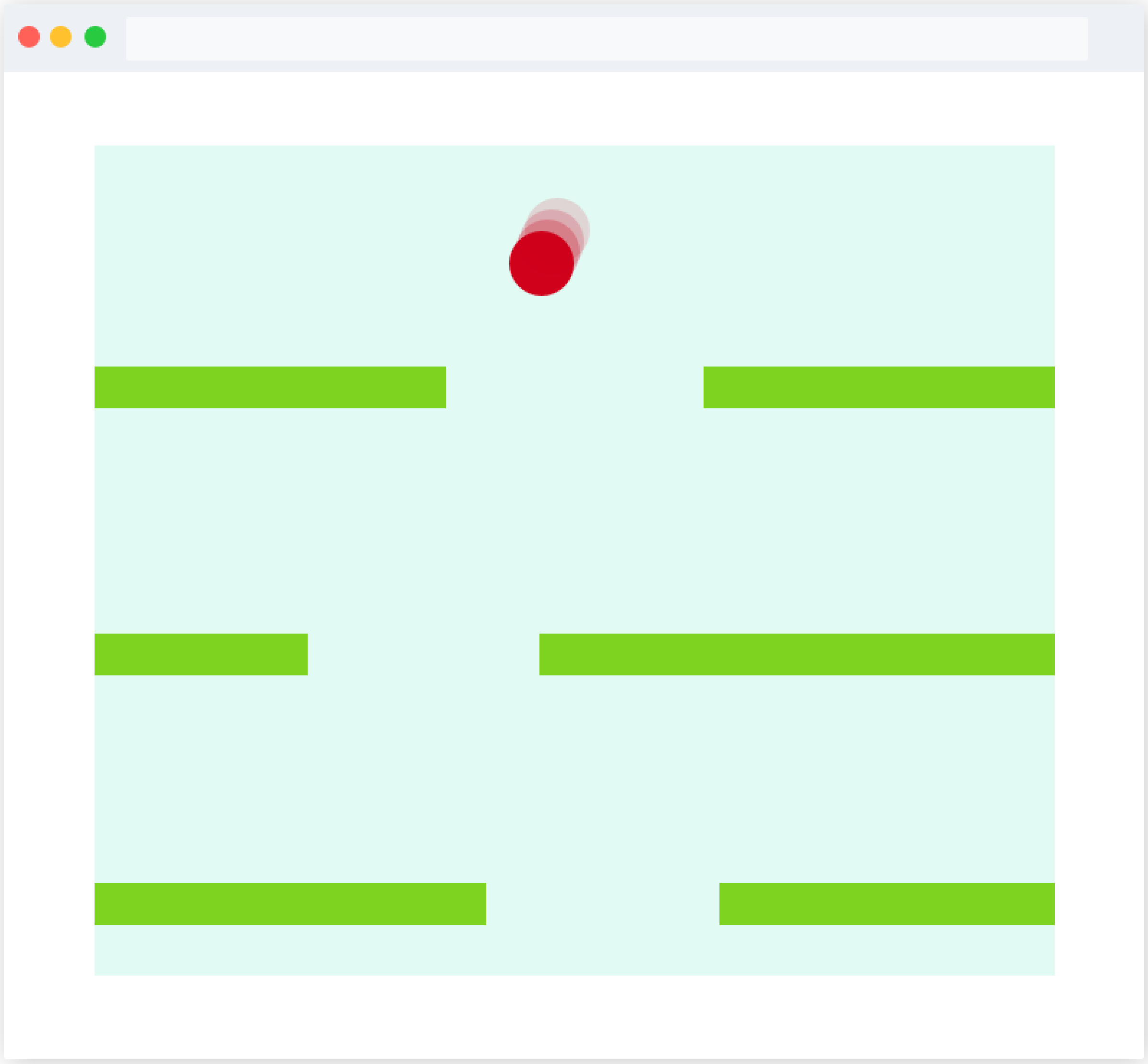
Premise

Tongue-computer interfaces are not new, there are many explorations in HCI research using various embedded sensor methods such as infrared, microphones, intraoral electrode arrays or even glossokinetic potential. And, it’s not only fun and games: tongue-computer interfaces can enable severely disabled users such as those with upper body impairments. From browsing the web to flying a drone, the tongue’s dynamic mobility offers wide potential for interaction design.

Our project

We propose a new, hardware-independent, approach to tongue-computer interaction. Using client-side JavaScript and a user’s own device webcam we aim to develop a method for input with tongue movements. Our application will be a simple JavaScript game that requires LEFT and RIGHT controls. The interface will leverage the newly released JavaScript implementation of Tensorflow to classify the player’s tongue movements without relying on sensors or even server-side computation.

Prototype:



Technologies

We plan to develop a simple JavaScript clone of Flappy Bird, a “side-scroller” where the player controls a bird, attempting to fly between columns of green pipes without hitting them. Our clone will take a vertical viewpoint, meaning the avatar will progress downwards requiring the player to move LEFT or RIGHT to avoid collisions. Our program design will exploit the features of object-oriented programming, using JavaScript classes to encapsulate graphical objects, interactive widgets and data needed for callback and drawing routines.

Our interface will use the new Tensorflow.js library and a method called “transfer learning”. Specifically, we plan to implement the k-nearest neighbors algorithm (KNN) on top of MobileNet. MobileNet is a small, low-latency, and low-power convolutional neural network trained on imagenet data. It is is designed to run efficiently in the browser. Our program will grab the second to last layer in the neural network and feed it into a KNN classifier. This will allow us to rapidly train our own tongue-based classes. The classes will be trained on images we will capture of people sticking their tongue to the left, right and mouth closed (no tongue).

Feasibility test

Our greatest technical risk is using Tensorflow.js to build a tongue classifier. Initially it was unknown if we could make reasonable inferences with a simple webcam. To test the feasibility, we created a test JavaScript app. Using Tensorflow.js, the app loads the MobileNet model and a KNN classifier (see `main.js`).

Next, we made a simple OpenCV program in python to collect training images (see `training/camera.py`). This program listens to keypresses and captures images of right, left or no tongue out. The training images are fed into app by json data and added to the classifier using KNN.

To test feasibility, the app creates a canvas to which the webcam is streamed. Then, with the new model, we classify the each canvas frame. The prediction values are printed to html using the innerText method. We found our methods begin to work reasonably well if there are at least 100 of each type of training images loaded. We plan to continue our feasibility tests with a greater quantity and more diverse images.

Demo

Serve from this root directory and visit your localhost.

I prefer to use Node.js http server, install with `npm install http-server -g` and run `http-server` at directory root.

MacOS

MacOS comes with PHP, try `PHP -S localhost:8000`

Windows

If you have Python, try `python -m SimpleHTTPServer 8000`

Develop

Install node if you don't have it: https://nodejs.org/en/download/

Install dependencies: `npm install`

Run the dev server: `npm run start`

Build production: `npm run build`

Train

Dependencies

Python 3: https://www.python.org/downloads/

OpenCV: `pip install opencv-python`

Capture photos

To take training photos, move to the training directory: `cd training`

Then, launch the Python program: `python camera.py`

With this program running, hit the 'r' key to take right tongue photos, the 'l' key to take left tongue photos, the 'n' key to take no tongue photos, and the 'q' key to quit the program.

Images will be saved to their respective folders.

To add new images to the json used to train the app, run `python data.py`