Physical Computing Workshop: Day 2

Sensors and actuators in our pockets

Anna Xambó

Department of Music, NTNU

17 October 2018

Learning Outcomes

— Identify the basic differences between sensors and actuators.

— Get a sense of how sensor data works in mobile phones.

— Be able to normalize sensor data.

— Explore mappings from sensor data to sound.

— Discern the fundamental properties of gestural interaction.

— Get familiar with web technologies (Handwaving.js) and web audio technologies (Tone.js, Flocking.js).

— Be able to adapt javascript code of a gesture-driven musical piece with custom sounds.

— Demonstrate a custom-made musical instrument in a performance setting.

— Reflect on the custom-made musical instrument and performance using a blogging style.

Preparation: Reading

— Read / skim through the following article and be ready to discuss it in class:

- Essl, G. and Rohs, M., 2009. Interactivity for mobile music-making. Organised Sound, 14(2), pp.197-207 [1]: `https://cpb-us-w2.wpmucdn.com/people.uwm.edu/dist/0/236/files/2016/09/os09-mobileinteractivity-2lt8a0i.pdf`

Preparation: What to Bring to Class?

— Your own mobile phone (it should be a smartphone).
— Your own laptop.
— Headphones / earplugs.

Preparation: What We Do Provide?

— 7 Music Angel speakers for the performance per site.
— Slides: xx.
— Code: xx.
— A handout: xx.

Pre-knowledge Activity: Mobile Music

Be ready to discuss topics related to mobile music from the suggested reading.

Outline

— Block I: Getting familiar with sensor data from your mobile phone
— Block II: Basic interactive behavior activities: mappings from sensor data to sound using gestural interaction
— Block III: Rehearsal and performance

Exercise 1: Sensor test

Most smartphones have built-in sensors that measure motion, orientation, and environmental conditions.

— Install two mobile apps that measure sensor data of your mobile phone and compare them: what are the similarities and differences?
— Compare the sensors between the mobile phones of your group members: what are the similarities and differences?
— Observe the units of measure for each sensor: any discoveries or surprises?
— How would you use these sensors for a sensor-based musical app? (e.g. microphone to sing, speakers to play a sound...)
— What sensors (and how) would you use for a gesture-based musical app?

Further reading:
— Sensors overview (Android):
  https://developer.android.com/guide/topics/sensors/sensors_overview
— Sensors overview (iPhone):
  https://developer.apple.com/documentation/coremotion
— Accelerometer vs. Gyroscope: What's the Difference?:
  https://www.livescience.com/40103-accelerometer-vs-gyroscope.html

Exercise 2: Accelerometer test in JavaScript

**Accelerometer
Javascript Test**

- acceleration x: g
- acceleration y: g
- acceleration z: g

Modern browsers now support direct access to the accelerometer. The *devicemotion* event is fired at a regular interval and indicates the amount of physical force of acceleration the device is receiving at that time.

— Get familiar with the code from the file "JS_01_accelerometer_API/index.html" and compare it with the specification of the *devicemotion* event: `https://developer.mozilla.org/en-US/docs/Web/Events/devicemotion` – To inspect the code you will need a code editor (e.g. Atom, Sublime), a local web server (e.g. *Python SimpleHTTPServer*, *Node http-server*), a browser in your laptop and a browser in your smartphone.

— What part of the code calculates the event and what part of the code prints the results?

Exercise 3: Normalization of accelerometer data in JavaScript

Accelerometer Javascript Test

- acceleration x: g
- acceleration y: g
- acceleration z: g

Normalization by scaling between 0 and 1 (feature scaling) is a common calculation to facilitate relations with other domains e.g. mappings to sound. The formula for normalization is:

$x_{new} = \dfrac{x - x_{min}}{x_{max} - x_{min}}$ where x_{new} is the normalized value, x_{min} and x_{max} are the minimum and maximum values within a range.
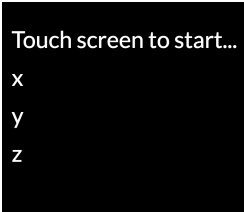
— Normalize the accelerometer data from file "JS_02_norm_accelerometer_API/index.html". Create a function so that when you call the function you get as a result the normalized value:
x_{new} = *normalize(x, x_max, x_min)*

Exercise 4: Handwaving

Handwaving is a a system for participatory mobile music based on accelerometer gesture recognition [2]:
https://github.com/g-roma/handwaving

Touch screen to start...
x
y
z

— Get familiar with the code of Handwaving (*HW_01_basic*, especially the files "index.html" and "example.js").
— Incorporate the code of Exercise 3 (normalization of accelerometer data) to the file (*HW_02_normalization/example.js*).

Exercise 5: Handwaving + Flocking.js

Flocking is a JavaScript audio synthesis framework:
`http://flockingjs.org`.

— Get familiar with the code of Handwaving + Flocking
(*HW_03_basic_sound_flocking*, especially *function setSound()*
(lines 11-21) and lines 196–238 from "example.js").

— Get familiar with the code of Handwaving + Flocking
(*HW_04_basic_sound_samples_flocking*, especially *function setSound()* (lines 14–195) and lines 381–432 from "iberlin.js").

Exercise 6: Handwaving + Tone.js

Tone.js is a framework for creating interactive music in the browser:
https://tonejs.github.io

— Explore the Tone.js library and implement it to Handwaving as
the sound engine.

References

[1] Georg Essl and Michael Rohs. "Interactivity for Mobile Music-Making". In: *Organised Sound* 14.2 (2009), pp. 197–207.

[2] Gerard Roma, Anna Xambó, and Jason Freeman. "Handwaving: Gesture Recognition for Participatory Mobile Music". In: *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*. ACM. 2017, p. 26.