Rapid Prototyping of New Instruments with CodeCircle

Michael Zbyszynski
EAVI Group, Department of Computing
Goldsmiths University of London
New Cross, London, SE14 6NW, UK
m.zbyszynski@gold.ac.uk

Mick Grierson
EAVI Group, Department of Computing
Goldsmiths University of London
New Cross, London, SE14 6NW, UK
m.grierson@gold.ac.uk

Matthew Yee-King
EAVI Group, Department of Computing
Goldsmiths University of London
New Cross, London, SE14 6NW, UK
m.yee-king@gold.ac.uk

ABSTRACT

Our research examines the use of CodeCircle, an online, collaborative HTML, CSS, and JavaScript editor, as a rapid prototyping environment for musically expressive instruments. In CodeCircle, we use two primary libraries: MaxiLib and RapidLib. MaxiLib is a synthesis and sample processing library which interfaces with the Web Audio API for sound generation in the browser. RapidLib is a product of the Rapid-Mix project, and allows users to implement interactive machine learning, using "programming by demonstration" to design new expressive interactions.

Author Keywords

mobile devices, machine learning, javascript, browser-based NIMEs, web audio, websockets, MIDI

ACM Classification

H.5.1 [Information Interfaces and Presentation] Multimedia Information Systems – Animations. H.5.2. [Information Interfaces]: User Interfaces – input devices and strategies; interaction styles; prototyping; user-centered design. H.5.5 [Information Interfaces and Presentation] Sound and Music Computing. D.2.2 [Software Engineering]: Design Tools and Techniques – User interfaces.

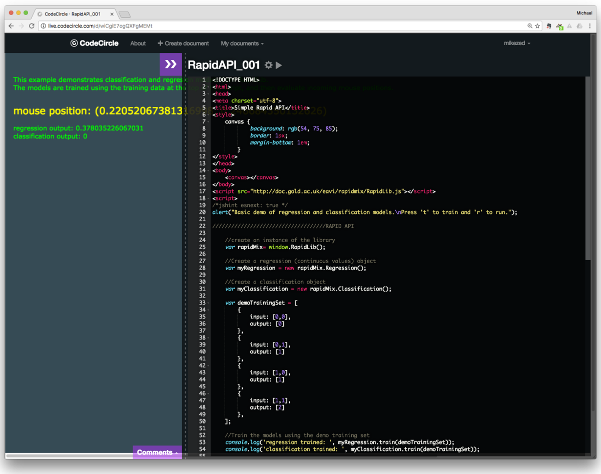
1. INTRODUCTION

When designing new instruments, it is usually desirable to refine a design by iterating over a number of prototypes. As a design iterates, it should become more aligned with the physical affordances of the performer as well as the aesthetic needs of the performance. Our research examines the use of web browsers and machine learning to streamline the prototyping process, allowing developers to program instruments and train them by example in a relatively simple environment.

Web browsers have been identified as appealing hosts for new expressive interfaces. Roberts, Wakefield and Wright[11] demonstrated the potential for browsers, and specifically the Web Audio API[1] to allow both synthesizers and interfaces to be programmed in JavaScript. Wyse and Subramanian[15] examine computer music in browsers more generally, noting the potential offered by browsers' intrinsic connection to networks, ease of access, and portability. New standards from W3C and extensions to JavaScript have made browser technology more "viable" for musical work.

However, timing can be problematic. Web Audio API scripts are run in the browser's main thread, which can be prone to latency and jitter. Jitter can be reduced by using the Web Audio Clock, rather than the JavaScript clock.[13] Large buffer sizes are required (>1024 samples), and current implementations impose a fixed latency of two buffers. As this paper will show, connectivity to sensors is also a potential "pain point."

In spite of known deficiencies when compared to platform-specific applications, we believe that web apps and web browsers can be excellent environments for collaborative prototyping. We have been exploring this space using CodeCircle and a new machine learning library, RapidLib. This paper will discuss our early prototypes with these tools. We will outline a potentially promising approach to designing NIMES and present some preliminary studies that demonstrate this promise.



This example demonstrates classification and regres
The models are trained using the training data at the

mouse position: (0.220520673813

regression output: 0.378035226067031
classification output: 0

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Simple Rapid API</title>
<style>
    canvas {
            background: rgb(54, 75, 85);
            border: 1px;
            margin-bottom: 1em;
        }
</style>
</head>
<body>
    <canvas></canvas>
</body>
<script src="http://doc.gold.ac.uk/eavi/rapidmix/RapidLib.js"></script>
<script>
/*jshint esnext: true */
alert("Basic demo of regression and classification models.\nPress 't' to train and 'r' to run.");

///////////////////////////////////RAPID API

    //create an instance of the library
    var rapidMix= window.RapidLib();

    //Create a regression (continuous values) object
    var myRegression = new rapidMix.Regression();

    //Create a classification object
    var myClassification = new rapidMix.Classification();

    var demoTrainingSet = [
        {
            input: [0,0],
            output: [0]
        },
        {
            input: [0,1],
            output: [1]
        },
        {
            input: [1,0],
            output: [1]
        },
        {
            input: [1,1],
            output: [2]
        },
    ];

    //Train the models using the demo training set
    console.log('regression trained: ', myRegression.train(demoTrainingSet));
    console.log('classification trained: ', myClassification.train(demoTrainingSet));
```

Figure 1: CodeCircle interface, with code editor pane on the right and resulting web page on the left.

2. CODECIRCLE

2.1 Description

CodeCircle was created by Fiala, Yee-King and Grierson[3] with the following objectives:

[1]`https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API`

NIME'17, May 15-19, 2017, Aalborg University Copenhagen, Denmark.

- To design and develop a web-based interface that enables real-time, collaborative, and social coding in a creative context.

- To implement an integrated code sharing and collaboration system that enables study of the process of collaborative, creative coding.

- To devise a platform that is attractive and accessible to learners and professionals alike, and can be used in a variety of contexts including computing education.

It is an online editor for web-specific code, including HTML, CSS, and JavaScript. As seen in Figure 1, it presents the user with a pane for editing, which can be hidden, and a viewer frame, where the resulting web page is running. The editor features code indentation, highlighting, and completion as well as live code validation, all of which assist the developer in quickly writing accurate, functional code. Documents can be marked private or public. Public documents are editable by multiple users at once, and updated reactively, allowing for real-time, collaborative programming.

The design of CodeCircle aligns strongly with the needs of developers of new instruments who wish to work collaboratively, quickly, in a simple but powerful environment, and with little or no installed software. It can run in any computer lab where the user has access to the internet and a modern browser, with the same interface on Linux, OSX, or Windows platforms. Furthermore, completed documents can be exported as HTML5 that can run in modern browsers without any dependency on CodeCircle. This includes mobile browsers.

2.2 Libraries

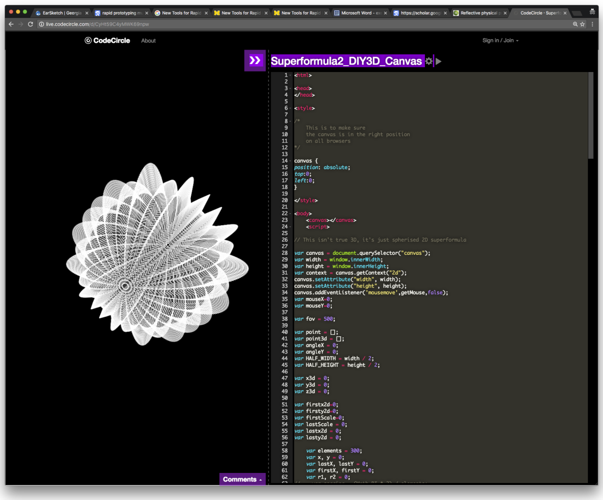
With the addition of third-party media libraries, see Table 1, CodeCircle can generate and render a large palette of text, audio, and visual media.

Table 1: Libraries available in CodeCircle

name	information
p5js	`http://p5js.org/`
MaxiLib	see below
processing.js	`http://processingjs.org/`
ThreeJS	`https://threejs.org/`
Marked	`https://github.com/chjj/marked`
FontAwesome	`http://fontawesome.io/`
jQuery	`https://jquery.com/`
SoundJS	`http://www.createjs.com/soundjs`

Grierson's Maximillian[6] is available directly in CodeCircle as a JavaScript library, MaxiLib. Maximillian and MaxiLib are open-source audio synthesis and signal processing libraries designed to facilitate artists and creatives who are learning to program or are rapidly prototyping audio applications. They include standard waveforms, envelopes, sample playback, filters with resonance, delay lines, FFTs, granular synthesis and low-level feature extraction. Although the Maximillian C++ library was designed with openFrameworks[2] in mind, the JavaScript library fits naturally into CodeCircle's pedagogical and prototyping functions.

The MaxiLib library has been ported, or "transpiled," from C++ to JavaScript using Emscripten[17]. This allows the developers to maintain one code base across multiple implementations. Also, Emscripten outputs asm.js, a strict subset of JavaScript that is optimized for ahead-of-time compilation and has been shown to run within a factor of two slowdown versus native code.[8]



```
<html>

<head>
</head>

<style>

/*
    This is to make sure
    the canvas is in the right position
    on all browsers
*/

canvas {
position: absolute;
top:0;
left:0;
}

</style>

<body>
    <canvas></canvas>
    <script>

// This isn't true 3D, it's just spherised 2D superformula

var canvas = document.querySelector("canvas");
var width = window.innerWidth;
var height = window.innerHeight;
var context = canvas.getContext("2d");
canvas.setAttribute("width", width);
canvas.setAttribute("height", height);
canvas.addEventListener('mousemove',getMouse,false);
var mouseX=0;
var mouseY=0;

var fov = 500;

var point = [];
var point3d = [];
var angleX = 0;
var angleY = 0;
var HALF_WIDTH = width / 2;
var HALF_HEIGHT = height / 2;

var x3d = 0;
var y3d = 0;
var z3d = 0;

var firstx2d=0;
var firsty2d=0;
var firstScale=0;
var lastScale = 0;
var lastx2d = 0;
var lasty2d = 0;

    var elements = 300;
    var x, y = 0;
    var lastX, lastY = 0;
    var firstX, firstY = 0;
    var r1, r2 = 0;
```

Figure 2: Although CodeCircle includes third-party drawing libraries, impressive results can also be had using HTML5's Canvas API.

3. RAPIDLIB

3.1 Description

NIME developers need to interface with external hardware. Hartmann, Abdulla, Mittal and Klemer (2007)[7] have identified that "programming by demonstration" with a layer of machine learning technology can expedite the development process and improve results. RapidLib provides a machine learning library which allows such development and fits with the rapid prototyping philosophy of CodeCircle.

RapidLib was developed as part of the *Real-time Adaptive Prototyping for Industrial Design of Multimodal Interactive and eXpressive technologies* (RAPID-MIX) project, an Innovation Action funded by the European Commission under the Horizon 2020 program. It is a lightweight set of libraries (both C++ and JavaScript) that implements Interactive Machine Learning (IML) in the style of Fiebrink's[4] Wekinator[3]. IML allows developers and end-users to quickly customize interactions by demonstration, encoding intuitive knowledge about performance gestures into trained machine learning models. IML is characterized by smaller data sets (relative to classical machine learning) and rapid iteration between training and testing. Using RapidLib in CodeCircle, performers can create a set of gestures associated with desired outcomes and immediately (a few seconds) experiment with an interactive space based on those gestures. Undesirable results can be refined or discarded as the design process continues.

Like MaxiLib, RapidLib has been transpiled into JavaScript using Emscripten and enjoys the same benefits of ease of development and high performance. Eventually, RapidLib will be incorporated into CodeCircle at the same level as the other libraries listed above. RapidLib can also be accessed outside of CodeCircle by linking to `http://doc.gold.ac.uk/eavi/rapidmix/RapidLib.js`

4. COMMUNICATING WITH SENSORS

[2]`http://openframeworks.cc/`

[3]`http://www.wekinator.org/`

This section is a brief survey of methods to bring sensor and media data into the browser environment. Characteristic of web development, there are many methods and implementations are sometimes inconsistent across browsers and platforms. The sections below describe methods we have been able to use.

4.1 Mouse & keyboard

Keyboard and mouse gestures can be used to control complex musical material. IML can allow users to quickly map two-dimensional control spaces to high-dimensional control spaces[9]; these mappings can be generated and refined without any programming by the end-user.

4.2 Gamepad API

The Web Gamepad API[4] is still in the draft stage. The current API supports buttons (which can have an analog value) and axes (normalized from -1.0 to 1.0) for up to four controllers. There is also a draft for extensions to this API that includes haptic feedback and gamepad position.[5]

The authors were able to use this API in Chrome and Firefox on OSX and Windows, using a Saitek Xbox-type controller, a GameTrak, and a Logitech Flight controller. We view this API as very promising, and are planning future prototypes to explore further.

4.3 Sensors on mobile devices

Mobile operating systems (both iOS and Android) provide specific API's for web apps directly access on=board sensors, including accelerometers, gyroscopes, magnetometers, and GPS. These data can be forwarded to a central server, as in the CoSiMa project[12], or processed directly on the device.

4.4 MIDI

The Web MIDI API[6] is also in draft form, and is currently only implemented in Chrome and Opera[7]. While this might be a serious limitation for a commercial product, it does not seem unreasonable to ask developers of experimental instruments to download a common browser. In December 2016, Chrome had more than 50% share of desktop browsing,[8] and even more users have Chrome on their computer. An example of basic MIDI interaction with CodeCircle can be found at `http://live.codecircle.com/d/H7TL6qMDf4QSxxjn5`.

4.5 WebSockets

The most secure and rich method for two-way data exchange between browser and server is currently via the WebSocket protocol. For communication between sensors and browsers running on the same computer, WebSockets require that a server be running locally. Although this is somewhat inconvenient, it is not an insurmountable obstacle.

Many manufacturers provide WebSocket software that facilitates communication with their devices. For example, Myo's daemon[9] and BITalino's python server[10]. Communication with a Myo can be seen in Figure 3.

WebSockets can also be a general transport for Open Sound Control.[14] We have developed a stand-alone NodeJS server[11] that passes OSC packets to and from the browser, and a special build of CodeCircle that understands OSC. Users of OSC-enabled software can send OSC to the server over UDP.

[4]`https://www.w3.org/TR/gamepad/`
[5]`https://w3c.github.io/gamepad/extensions.html`
[6]`https://webaudio.github.io/web-midi-api/`
[7]`http://caniuse.com/#feat=midi` – accessed 19 January 2017
[8]`http://gs.statcounter.com/#all-browser-ww-monthly-201612-201612-bar`
[9]`https://github.com/logotype/myodaemon`
[10]`https://github.com/BITalinoWorld/python-serverbit`

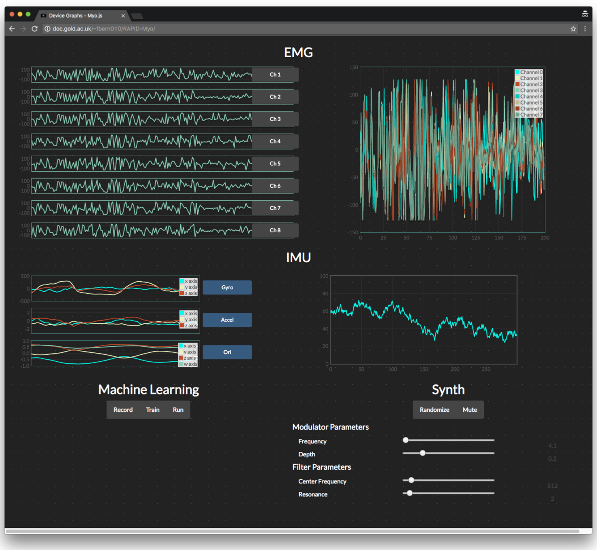


Figure 3: Myo data is captured over websockets, visualized, and used, with RapidLib to control parameters of a synthesizer implemented in MaxiLib. Page by Francisco Bernardo

4.6 WebRTC

WebRTC is a protocol for real-time communication between browsers[1], and allows for peer-to-peer exchange of audio, video, and control data. Incoming audio buffers can be passed to maxiLib as arrays of 32-bit floats for processing and/or feature extraction. An example using MFCCâĂŹs for texture/style recognition can be found at `https://live.codecircle.com/d/kpebzng3rPcCFnLnY`

5. EVALUATION

5.1 CodeCircle

CodeCircle's offers a superset of the features available in other, similar systems, including Gibber, codepen, jsfiddle and livecodelab.[3] In addition it offers rapid access to media libraries, as mentioned above. CodeCircle has been used as a research workbench for computing education pedagogy, where its detailed analytics capabilities were used to examine coding behavior in student programmers [16]. It has been used as the programming IDE for an audiovisual programming MOOC [12] wherein approximately 500 users were active on the system for a period of 6 months. It is soon to be used in larger scale research studies with thousands of users, in a new computer music MOOC and with an undergraduate cohort.

5.2 RapidLib

User-centered design is one of the core methods of RapidMix; the design of RapidLib has been repeatedly prototyped and tested with the aim of creating a machine learning API that was both simple and powerful, especially for users in musical and artistic domains. Public documentation of this

[11]`http://gitlab.doc.gold.ac.uk/rapid-mix/rapid_web_service`
[12]`https://kadenze.com`

development can be found on the project website: `http://rapidmix.goldsmithsdigital.com/downloads/`

We recently ran a hack-a-thon run with a version of RapidLib wrapped as a JUCE library. JUCE is an open-source cross-platform C++ application framework, used for the development of desktop and mobile applications.[13] Approximately twenty proficient C++ developers were asked to implement machine learning using RapidLib at a one-day event in London. In general, participants felt that the JUCE RapidLib module was a capable tool to achieve their desired hacks. After a short introduction to machine learning, participants were able to easily navigate the JUCE RapidLib module code and documentation and seemed comfortable using it.

There were, however, some conceptual misunderstandings among the participants, particularly when distinguishing between classification or regression models. There was also a desire for more insight into and control of the training process, particularly with large data sets where training seemed particularly CPU intensive. The need for a feature extraction stage between sensor or audio input and machine learning was also noted. A full write-up of this evaluation will be available at the Rapid-Mix website by the time of publication.

6. CONCLUSIONS & FUTURE WORK

Fiala, Yee-King, and Grierson[3] evaluate the design features of CodeCircle in comparison to Gibber[10], among others. Gibber is a mature environment for live audio and media coding in browsers. Whereas MaxiLib pursues efficiency through the use of asm.js and its potential for Ahead-of-Time (AOT)compilation, Gibberish generates code optimized for Javascript's Just-in-Time (JIT) compiler. More research is called for to evaluate the functional differences between these two approaches.

Rapid-Mix is an ongoing project, and there is a long road map of features to be added that would improve instrument prototyping. We are currently integrating more machine learning algorithms into the API, including algorithms that support recognition of gestures in time[5], rather than just states. We also plan to expose feature extractors from Maximillian, Pipo[14], and Essentia[2].

Although training is reasonably efficient with smaller data sets, users of larger data sets have expressed frustration. It would be possible to introduce web workers into RapidLib to expedite training with large data sets and preserve UI responsiveness during that process by moving training out of the main page thread. Unfortunately, Emscripten does not currently support transpiling of multithreaded C++ code, so threading optimizations would need to be implemented separately for each programming language.

7. ACKNOWLEDGEMENTS

The work reported here was supported in part by RAPID-MIX, an EU Horizon 2020 Innovation Action: H2020-ICT-2014-1 Project ID: 644862.

8. REFERENCES

[1] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan. Webrtc 1.0: Real-time communication between browsers. *Working draft, W3C*, 91, 2012.

[2] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. Essentia: An audio analysis library for music information retrieval. In *ISMIR*, pages 493–498. Citeseer, 2013.

[3] J. Fiala, M. Yee-King, and M. Grierson. Collaborative coding interfaces on the web. *Proceedings of the International Conference on Live Interfaces*, pages 49–57, June 2016.

[4] R. Fiebrink and P. R. Cook. The wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, 2010.

[5] J. Françoise, N. Schnell, and F. Bevilacqua. A multimodal probabilistic model for gesture–based control of sound synthesis. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 705–708. ACM, 2013.

[6] M. Grierson. Maximilian: A cross platform c++ audio synthesis library for artists learning to program. In *Proceedings of the International Computer Music Conference, New York*, 2010.

[7] B. Hartmann, L. Abdulla, M. Mittal, and S. R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–154. ACM, 2007.

[8] D. Herman, L. Wagner, and A. Zakai. asm. js: Working draft 18 august 2014. *Available online at asmjs. org/spec/latest. Accessed January 2017*, 2014.

[9] A. Momeni and D. Wessel. Characterizing and controlling musical material intuitively with geometric models. In *Proceedings of the 2003 conference on New interfaces for musical expression*, pages 54–62. National University of Singapore, 2003.

[10] C. Roberts and J. Kuchera-Morin. Gibber: Live coding audio in the browser. In *ICMC*, 2012.

[11] C. Roberts, G. Wakefield, and M. Wright. The web browser as synthesizer and interface. In *NIME*, pages 313–318, 2013.

[12] N. Schnell, S. Robaszkiewicz, F. Bevilacqua, and D. Schwarz. Collective sound checks: Exploring intertwined sonic and social affordances of mobile web applications. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 685–690. ACM, 2015.

[13] C. Wilson. A tale of two clocks - scheduling web audio with precision. *Available online at: https://www.html5rocks.com/en/tutorials/audio/scheduling/*.

[14] M. Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(03):193–200, 2005.

[15] L. Wyse and S. Subramanian. The viability of the web browser as a computer music platform. *Computer Music Journal*, 37(4):10–23, 2013.

[16] G. M. Yee-King, Matthew and M. d'Inverno. Do student programmers learn differently in stem and steam lessons. 2017, in press.

[17] A. Zakai. Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 301–312. ACM, 2011.

[13]`https://www.juce.com/`

[14]`http://ismm.ircam.fr/pipo/`