

ADC²¹

The background features a colorful waveform that transitions from blue on the left to red on the right. The waveform is composed of many thin, vertical lines of varying heights, creating a dense, textured effect. The colors are vibrant and contrast sharply against the dark background.

HYBRID PROTOTYPING WITH WEB TECH AND JUICE/C++

ARTHUR CARABOTT

outputoutput



- 120 +

Done With You



Tweak



Deep Sweep

Ambience



Phase Mod

Pumping



Pitch

Parameter control strip with 14 knobs and 14 icons. The icons include: two teal waveforms, two red double-headed arrows, two yellow vertical bars, two teal waveforms, one yellow vertical bar, one red double-headed arrow, and one red double-headed arrow.

A \flat Maj











How to use web technologies for productive prototyping.

- 1. Prototyping**
2. Why Web Tech?
3. Case Study: Arpeggiator
4. C++ / WebViews



```
class FactorTouchSensor {  
public:  
    FactorTouchSensor(FactorTouchSensor() {}  
    FactorTouchSensor(int pin) {  
        _pin = pin;  
        _led = new LED(_pin);  
    }  
    FactorTouchSensor(int pin, int ledPin) {  
        _pin = pin;  
        _led = new LED(ledPin);  
    }  
private:  
    int _pin;  
    LED *_led;  
};  
  
FactorTouchSensor *factorTouchSensor = new FactorTouchSensor(10, 12);  
  
void setup() {  
    pinMode(10, OUTPUT);  
    pinMode(12, OUTPUT);  
}  
  
void loop() {  
    if (digitalRead(_pin) == HIGH) {  
        digitalWrite(_led, HIGH);  
        delay(500);  
        digitalWrite(_led, LOW);  
        delay(500);  
    }  
}
```

EVERYTHING IS
GOOD

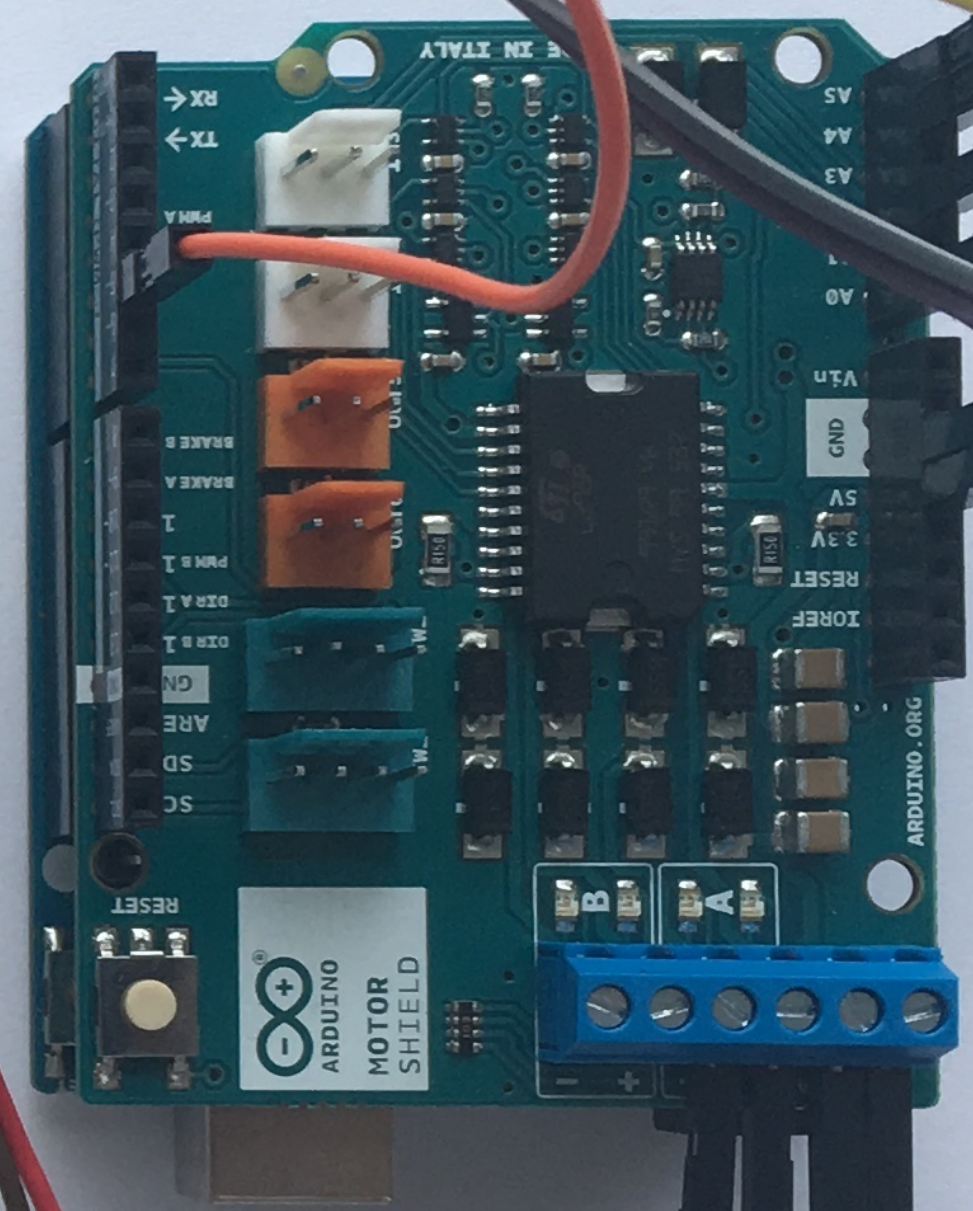
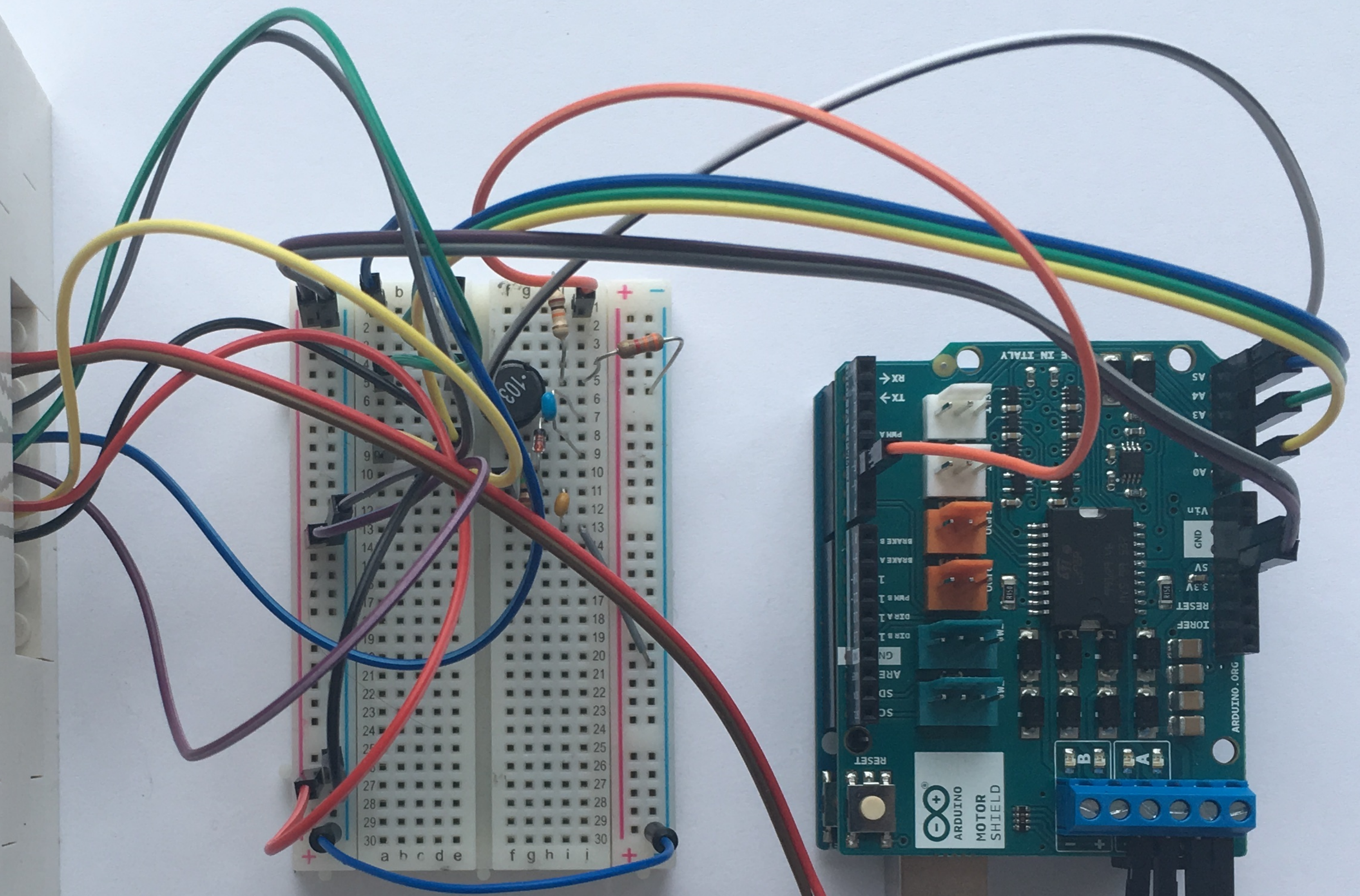
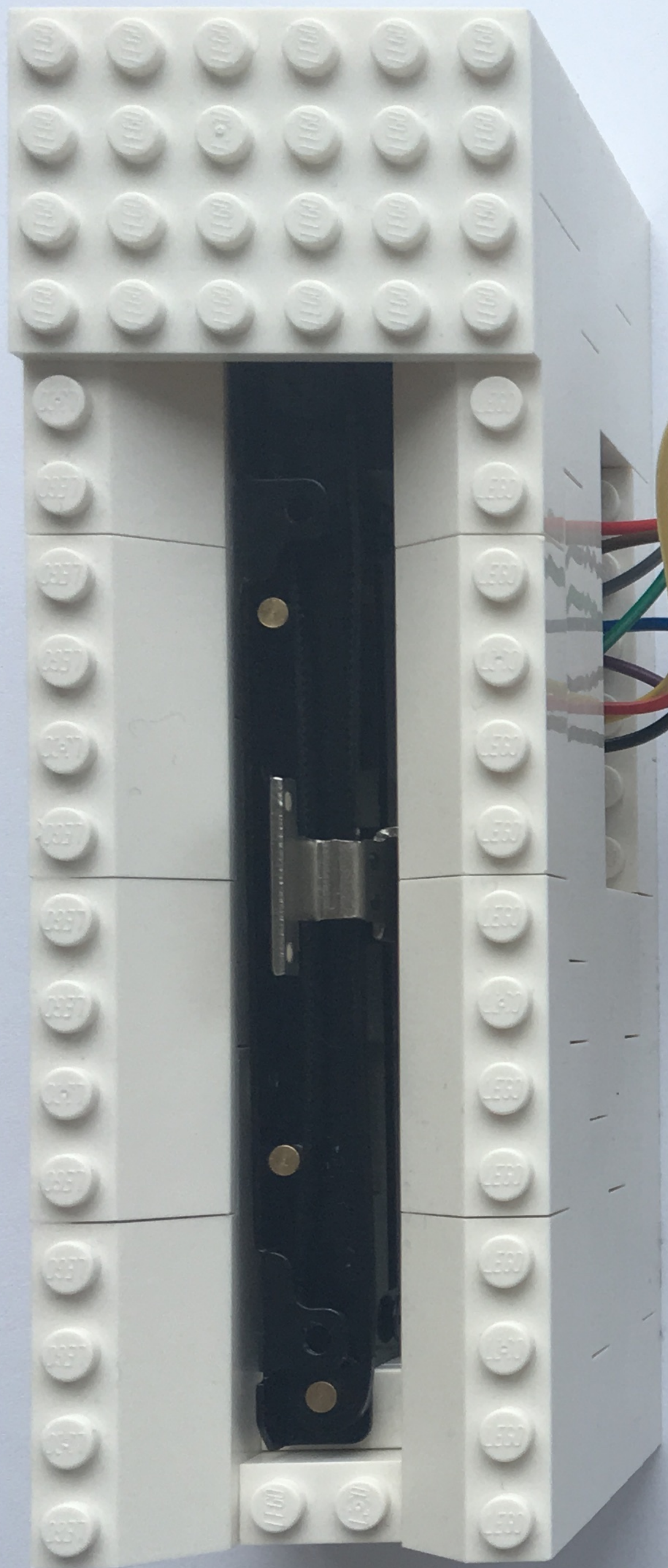
- 21.34
INFINITE PAGES
- CONNECTED APPARUS
- ONSET DETECTION
LAMPING
- HANDS
- CONNECTIONS

LEGO Architecture
Studio
16+

Oculus

duet

ONYX BLACKS



1. Communication

Feedback loop with yourself

Feedback loop with others

2. Cost

Time and money

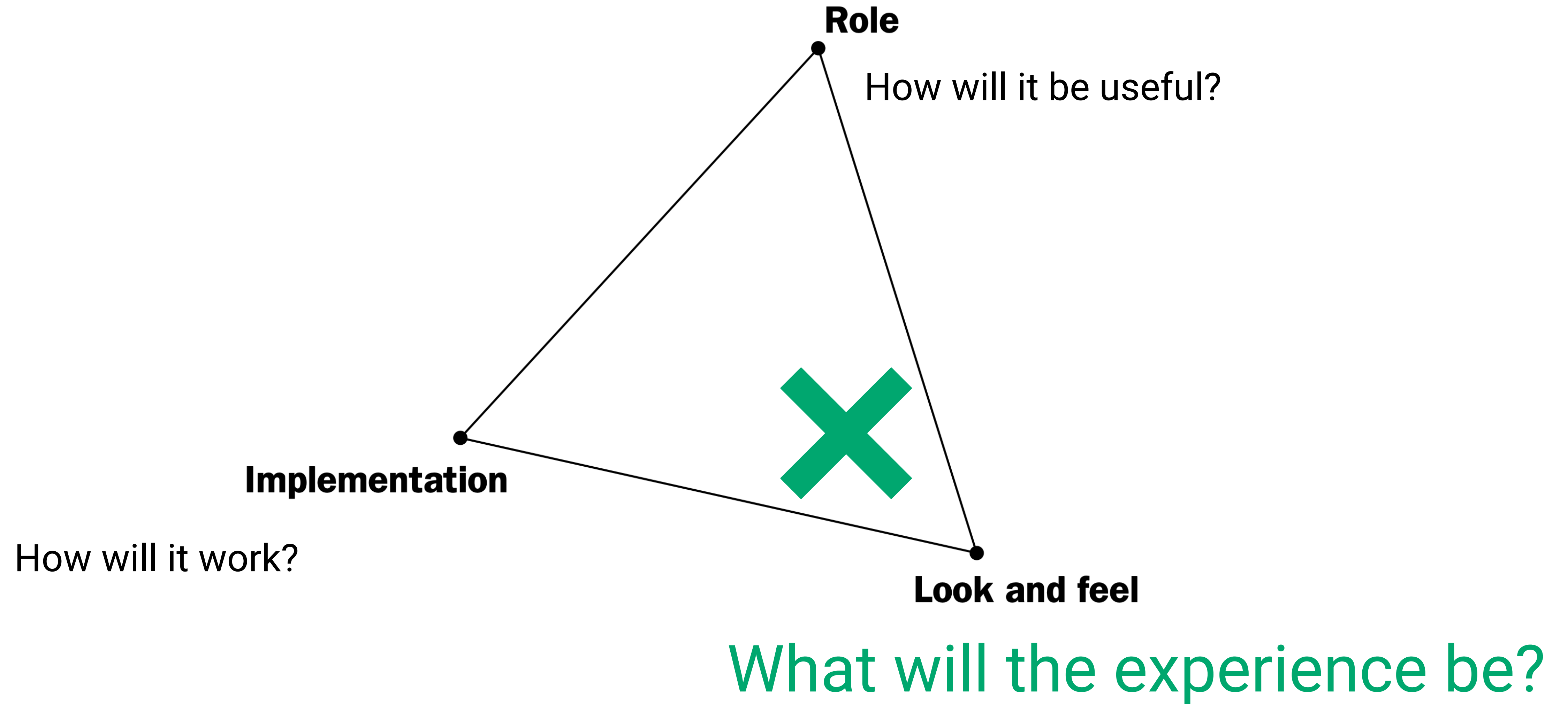
Motivation and creativity

Speed

Decrease iteration time

Increase quality

Look and feel prototypes



Prototyping



Prototyping

The screenshot displays a music production software interface with a green background. At the top, a track named "Chilea" is visible with a heart icon, a play button, and a "Tweak" button. Below this, a navigation bar includes "Layer Edit", "Mixer", "Macros", "Modulation", and "Arpeggiator". The "Arpeggiator" window is open, showing a piano roll for "Patternoids".

Arpeggiator Settings:

- Preset:** Patternoids
- Rate:** 1/32
- Steps:** 8
- Volume:** Visual bar graph
- Length:** 100%
- Chord:** Visual bar graph
- Octave:** Visual bar graph
- Panning:** Visual bar graph
- Chance:** Visual bar graph
- Offset:** Volume, Length, Chord, Octave, Panning, Chance (each with a knob)

Layer A Settings:

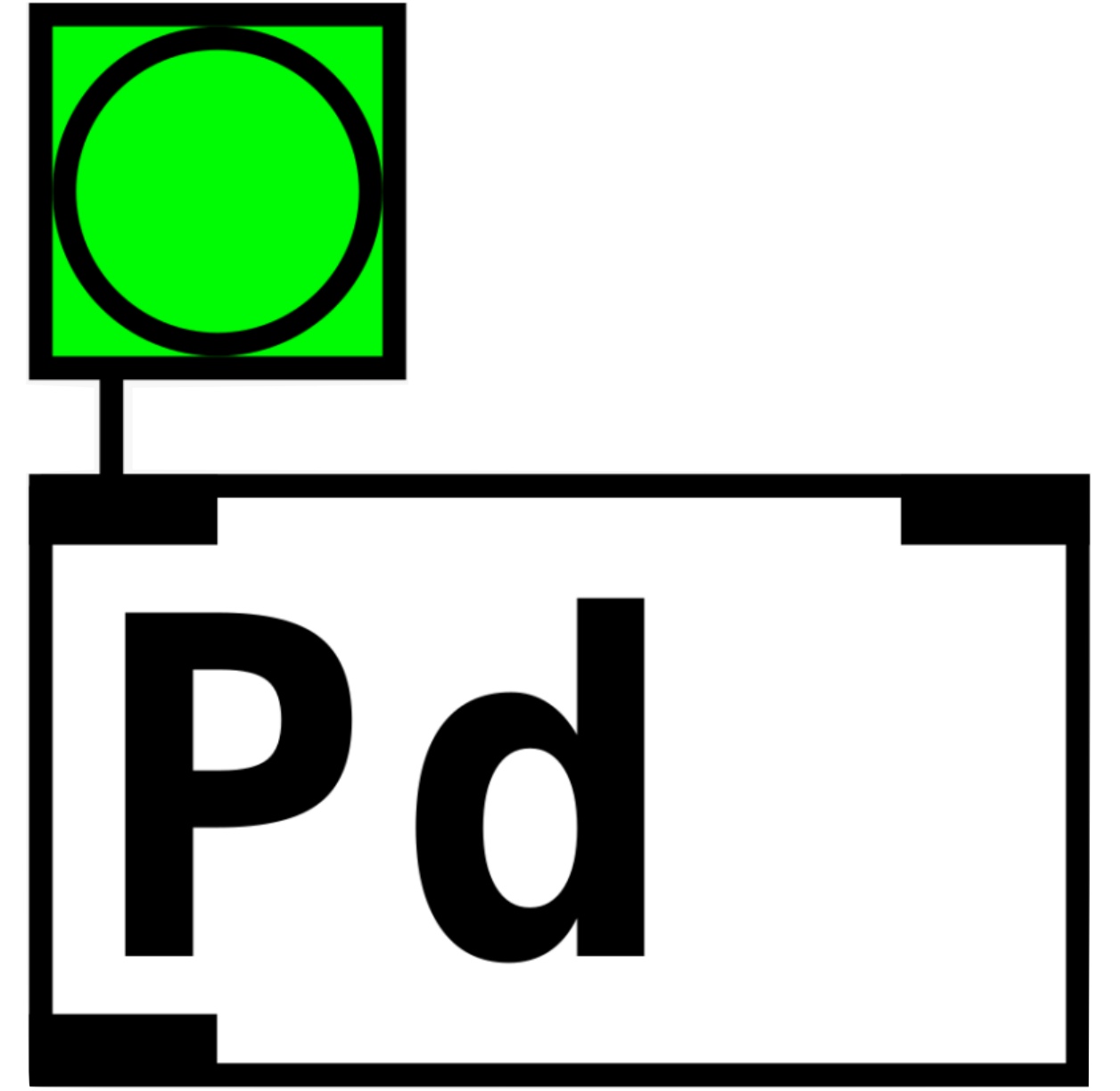
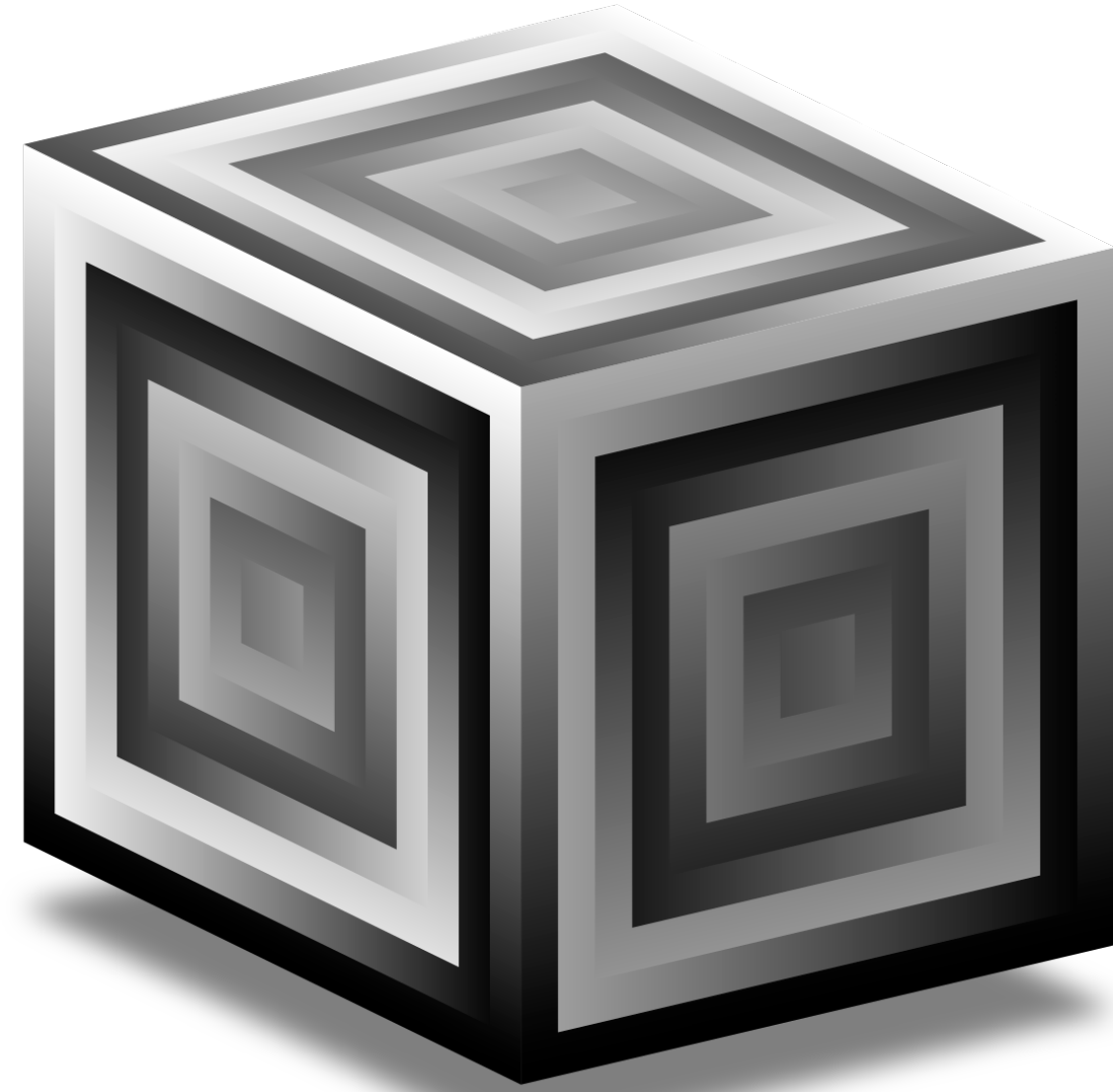
- Layer A:** Cello Vibrato Pizz Stab
- Vol:** Knob
- Speed:** Knob (x2)
- Reps:** Knob
- Swing:** Knob (3/4)
- TimeSig:** 3/4
- Type:** Up-Down
- Pedal:** LOW, OFF, HIGH

Layer B: Dusty Tape Pad

Layer C: Bright Kawala Non Vibr...

Bottom Panel: A piano roll with notes C1 through C7. C4 and C5 are highlighted in yellow.

Prototyping



Prototyping

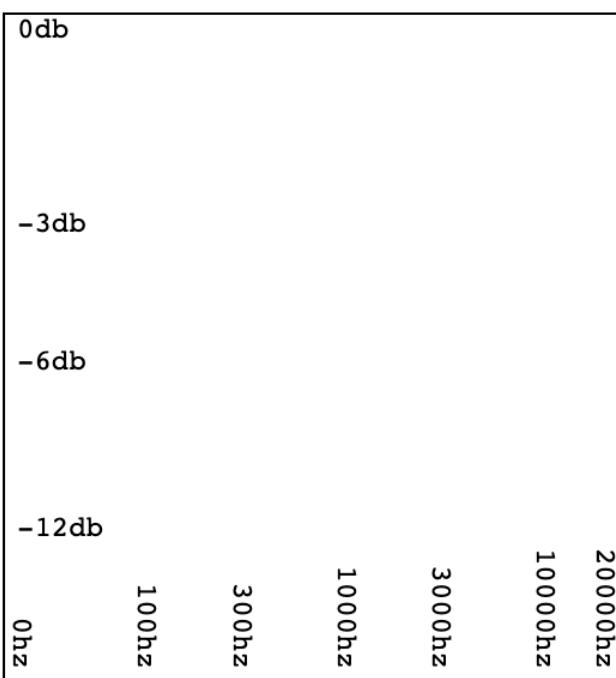
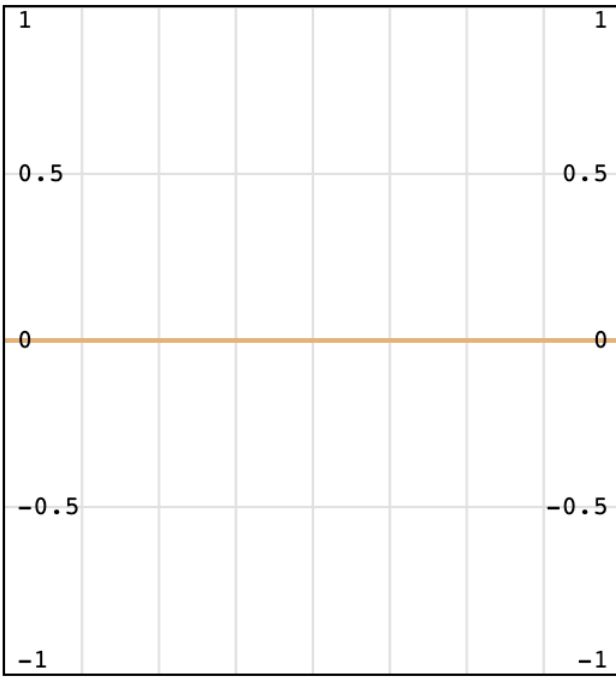
Audio DSP Playground

White Noise Sine Wave Bitcrusher Choose File No file chosen X

```
1 let time = 0;
2
3 function getSine(freq, time) {
4   return Math.sin(2 * Math.PI * freq * time);
5 }
6
7 function loop(numFrames, outL, outR, sampleRate) {
8   const freq = 666;
9   const amp = 0.1;
10
11   for (let i = 0; i < numFrames; i++) {
12     outL[i] = getSine(freq, time) * amp;
13     outR[i] = getSine(freq * 1.5, time) * amp;
14
15     time += 1 / sampleRate;
16   }
17 }
18
```

Run: cmd enter Stop: cmd .

Left Right Sum



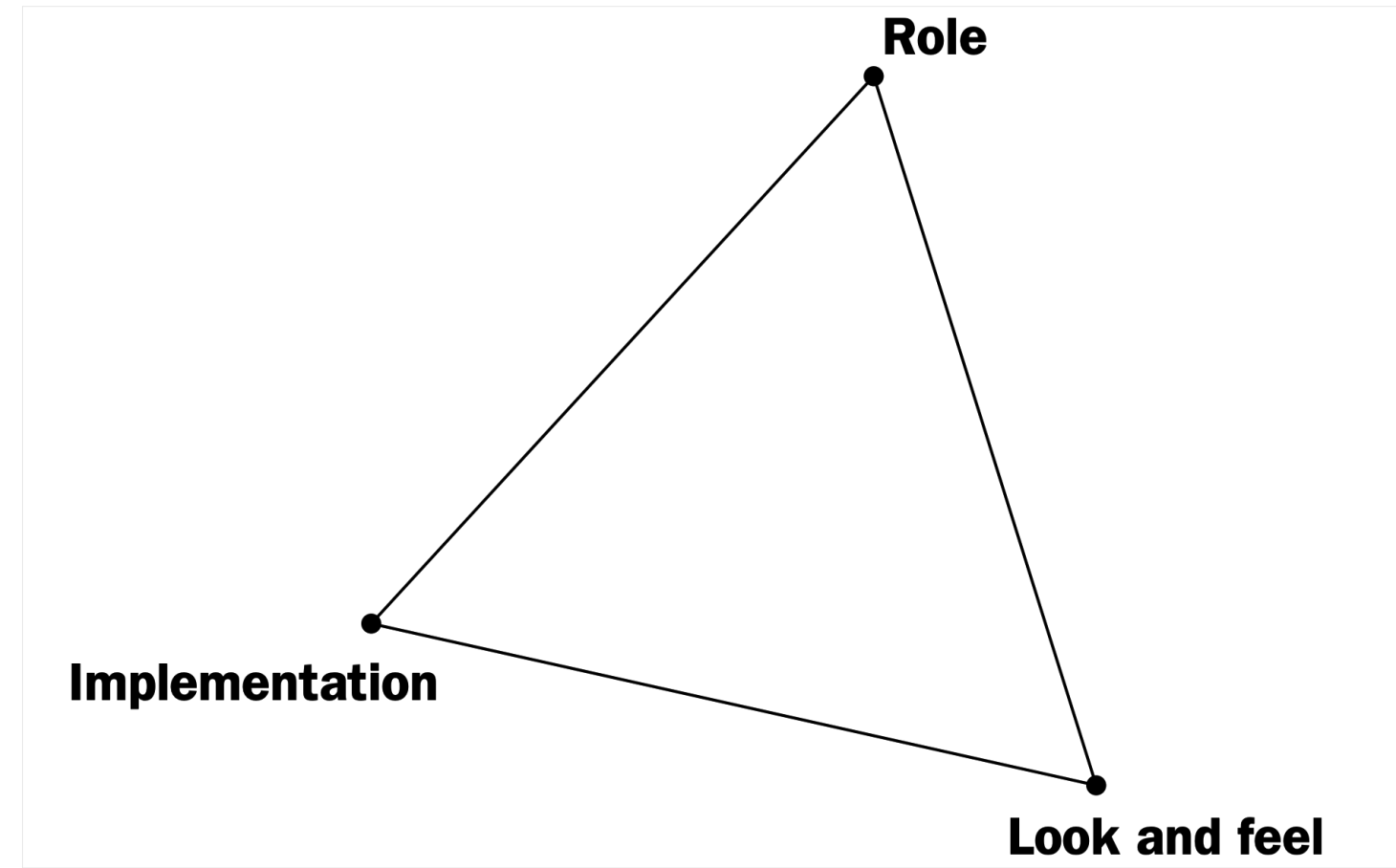
Prototyping



Marek Bereza

How to prototype Audio Software

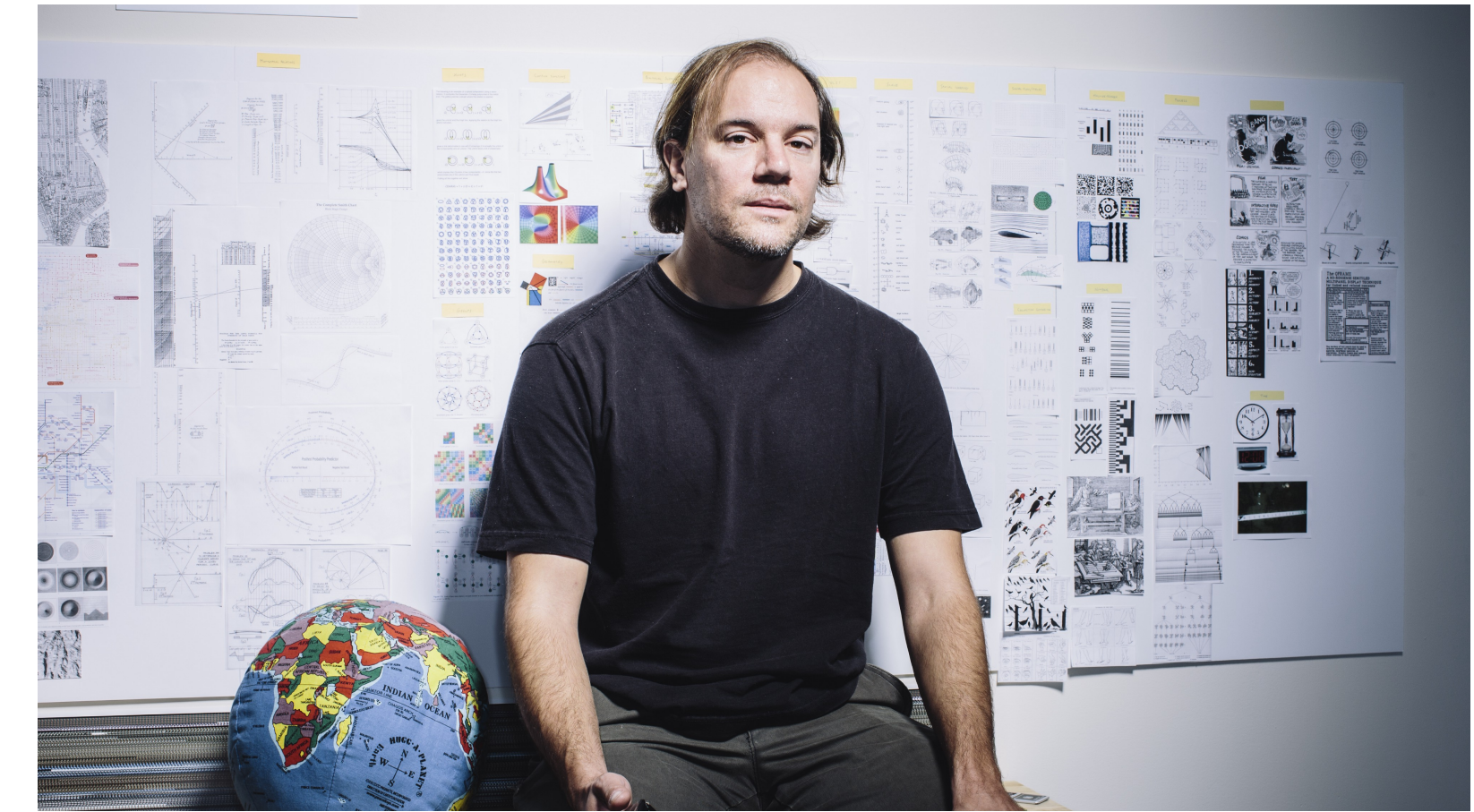
<https://youtube.com/watch?v=A01V8er6ljM>



Stephanie Houde and Charles Hill

What do Prototypes Prototype?

<https://hci.stanford.edu/courses/cs247/2012/readings/WhatDoPrototypesPrototype.pdf>



Bret Victor

Inventing on Principle

<https://youtube.com/watch?v=a-OyoVcbwWE>

1. Prototyping

2. Why Web Tech?

3. Case Study: Arpeggiator

4. C++ / WebViews

- 1. You**
2. Values of your tools
3. Things to leverage

Be good

1. You
- 2. Values of your tools**
3. Things to leverage

“The limits of my language
mean the limits of my world.”

Ludwig Wittgenstein
Tractatus Logico-Philosophicus, 1922

Why Web Tech?

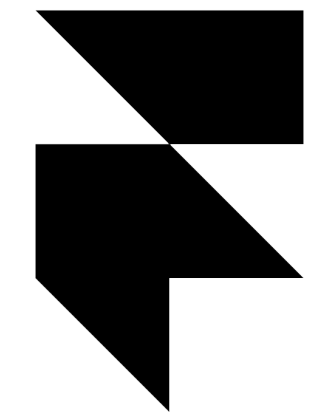


JUCE

HTML



CSS



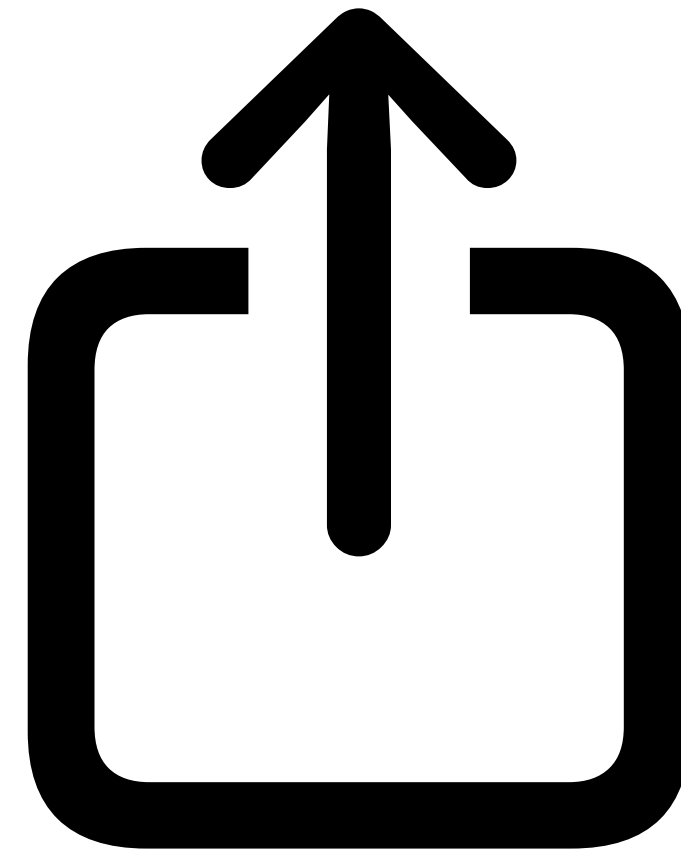
Speed


```
main.ts — speed-demo
TS main.ts 2, U x
src > TS main.ts > ...
1 import { start } from "@thi.ng/hdom";
2 import { div, h2 } from "@thi.ng/hiccup-html";
3
4 interface Param {
5   name: string;
6   value: number;
7   min: number;
8   max: number;
9   step: number;
10 }
11
12 const app = () => {
13   const params: Param[] = [
14     { name: "Gain", step: 0.01, min: 0, max: 1, value: 0.5 },
15     { name: "Freq", step: 1, min: 20, max: 20000, value: 666 },
16   ];
17
18   return () => div({},
19     h2({}, "Hello ADC"),
20   );
21 };
22
23 start(app(), { root: document.body });
24
```

Speed Demo

Hello ADC

Why Web Tech?



“If I send people an executable, they never open it. If I send them a link, they have no excuse.”

Alex Warth

1. You
2. Values of your tools
- 3. Things to leverage**

Web Audio API

Web MIDI API

Canvas / WebGL

Why Web Tech?

Playground

Learning Synths — Playground

Square Oscillator

Amplitude 100.0%

Width 0.0%

LFO Amount 0.0%

Envelope Amount 0.0%

Amplitude Envelope

Attack 30.00 ms

Decay 200.00 ms

Sustain 50.0%

Release 300.00 ms

Low-Pass Filter

Frequency 3.00 kHz

LFO Amount 0.0%

Envelope Amount 0.0%

Resonance 20.0%

LFO MODULATION

Shape

Frequency 1.00 Hz

Envelope Amount 0.0%

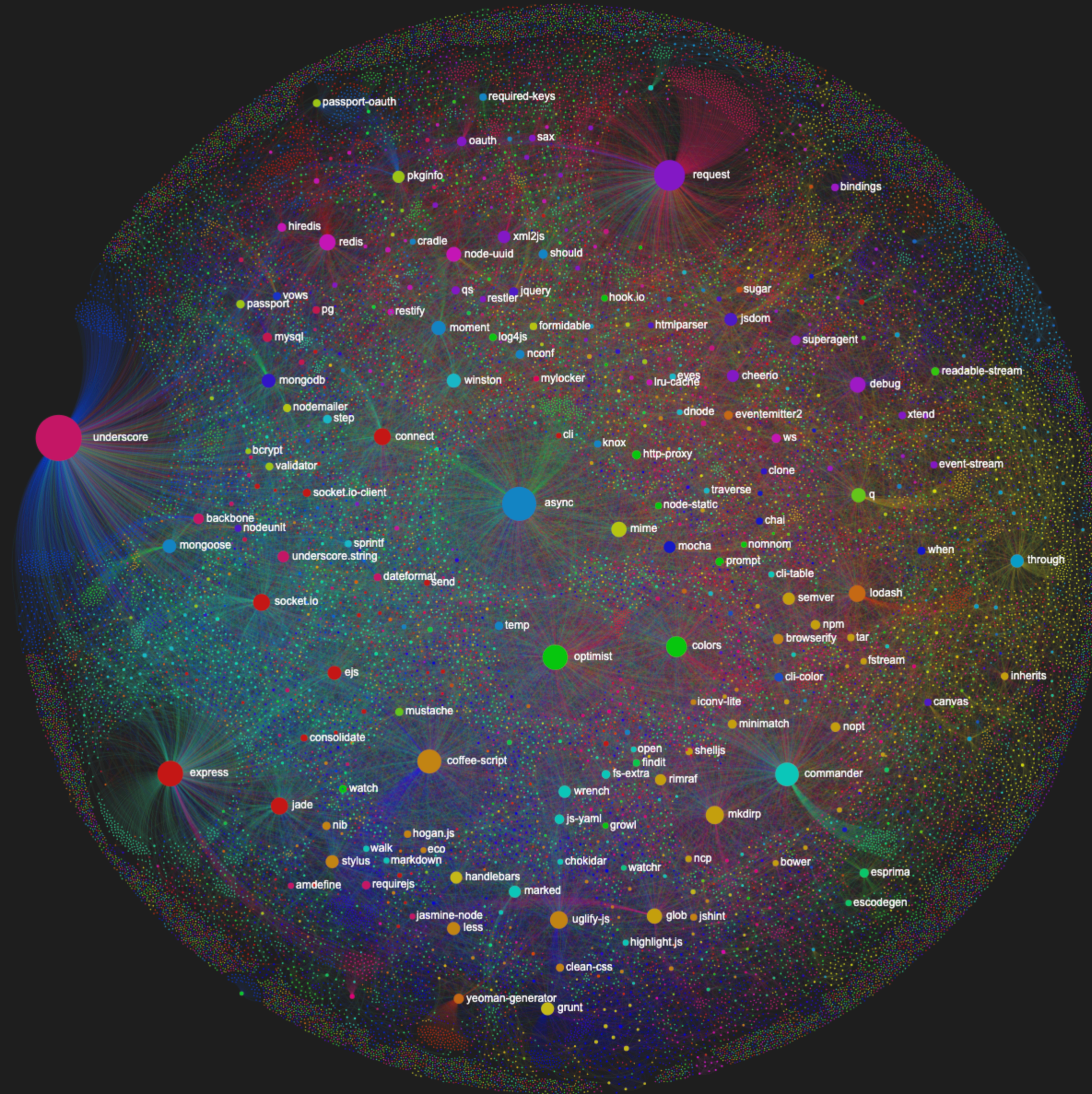
Saw Oscillator

Amplitude

Envelope MODULATION

Why Web Tech?

Your bag of tricks



Production VS Prototyping

Why Web Tech?

“Firstly, programs developed mainly to demonstrate feasibility may turn out in the long term to be **difficult to maintain and develop**; and secondly, their temporary user interfaces **may never be properly redesigned** before the final system is released.”

Stephanie Houde and Charles Hill
What do Prototypes Prototype? 1997

“There’s nothing more permanent than a temporary solution.”

Amit Shoham

1. Prototyping

2. Why Web Tech?

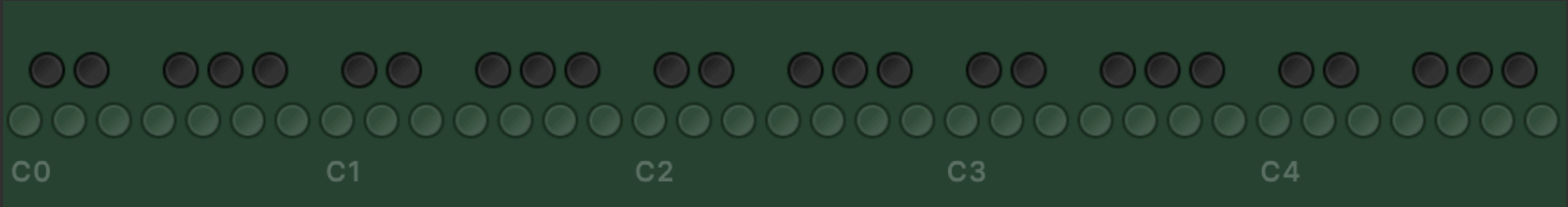
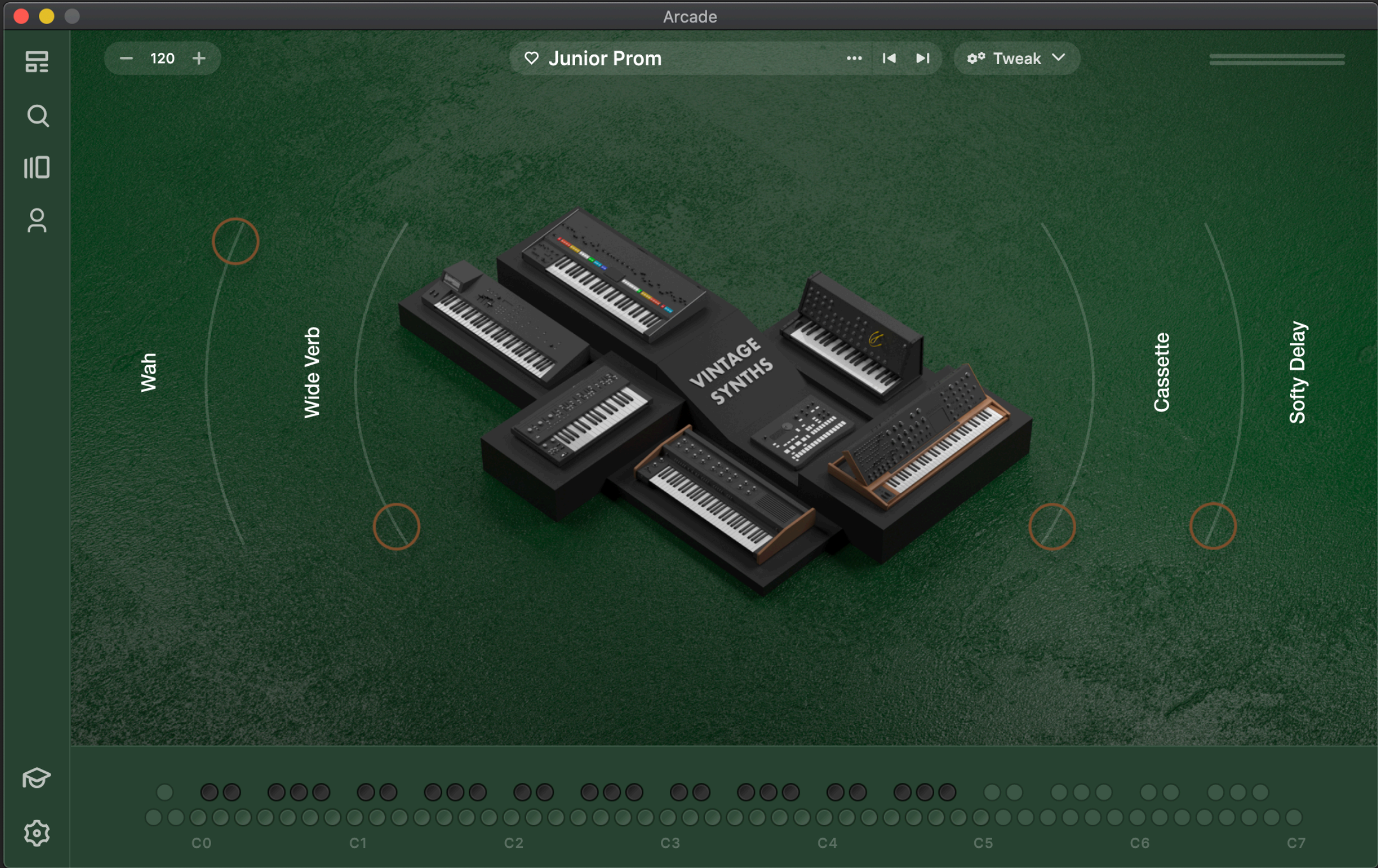
3. Case Study: Arpeggiator

4. C++ / WebViews

Case Study: Arpeggiator



Case Study: Arpeggiator



Case Study: Arpeggiator

The screenshot displays the Arcade software interface for an arpeggiator. The main window title is "Arcade" and the track name is "Junior Prom". The interface is divided into several sections:

- Top Bar:** Includes a volume knob set to 120, a track name "Junior Prom", and a "Tweak" button.
- Navigation:** A sidebar on the left contains icons for Layer Edit, Mixer, Modulation, and Macros.
- Layer Controls:** Three layers are visible, each with a waveform view and a control panel:
 - Layer A:** "Fat Filtered Bass". Controls include Semi, Freq, Cents, Reso, Follow, Drive, Amp Env, Filter Env, Pitch Env, and Amp.
 - Layer B:** "Lazerdisc Bass". Controls include Semi, Freq, Cents, Reso (set to 10.0%), Follow, Drive, Amp Env, Filter Env, Pitch Env, and Amp.
 - Layer C:** "Soft Eighties Decay". Controls include Semi, Freq, Cents, Reso, Follow, Drive, Amp Env, Filter Env, Pitch Env, and Amp.
- Bottom Section:** A piano roll view showing a sequence of notes across octaves C0 to C7.

Case Study: Arpeggiator

The screenshot displays the 'analog brass & winds' software interface. At the top, the track name is '001. Destructive Brass'. The interface is divided into several sections:

- Navigation:** 'main', 'edit', 'fx', 'rhythm', and 'arp' tabs. The 'arp' tab is currently active.
- Arpeggiator A:**
 - Pattern:** A 'down-up' pattern is selected, shown as a sequence of notes on a staff.
 - Rate:** A circular knob set to '1/16'.
 - Pedal:** A selector with 'low', 'off', and 'high' options, currently set to 'off'.
 - Pattern Editor:** A bar chart showing the sequence of notes for the selected pattern.
 - Reps:** A selector set to '1'.
 - Steps:** A selector set to '32'.
 - Buttons:** 'reverse' and 'random' buttons.
- Settings:** Three circular knobs for 'duration', 'swing', and 'octaves', and a 'fixed velocity' button.
- Bottom Bar:** Includes a power button, a volume knob, a 'Warble' effect button, and a 'tune -12.00' selector with sub-selectors for -12, -1, +1, and +12.

The 'arp type menu' is open, showing various arpeggiator patterns:

- pedal:** 'low', 'off', 'high'.
- Patterns:** 'up', 'down', 'up-down', 'down-up', 'zigzag up', 'zigzag down', 'z. z. up-down', 'z. z. down-up', 'random', 'as played', and 'chord'.
- Buttons:** 'cancel' and 'ok'.

Arp localhost:8080

MIDI Input: MIDI Output: Tempo State Input Notes

AKAI - MPKmini2 (1190016173) Apple Inc. - IAC Driver Bus 1 (-1372316423) 120 Undo Redo Reset no notes held

Layer A Current

Rate < 1/8 > ... +

Steps < 4 > Curr

Off

Volume 1 2 3 4

Length

Octave

Chord

Panning

Chance

Speed: 1

Layer B

Layer C

Overrides	Volume	Length	Octave	Chord	Panning	Chance
	0	0	0	0	0	0

Arcade

120 Space Dust Tweak

Space Sprinkles Bandpass Width

BRAIN WAVES

C0 C1 C2 C3 C4 C5 C6 C7

MIDI Studio

Default

MPKmini2 IAC Driver

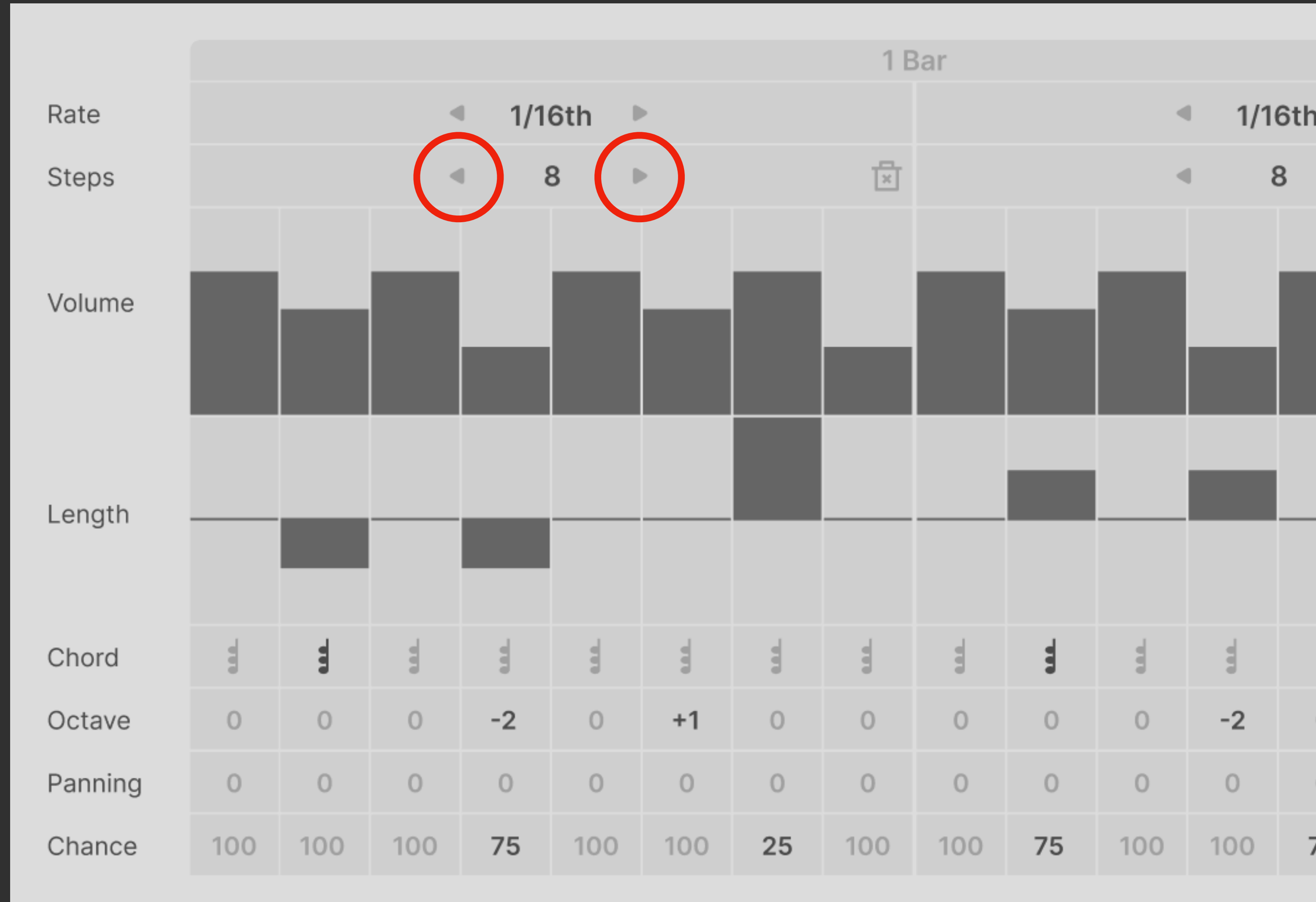
Case Study: Arpeggiator

The screenshot displays the 'Arpeggiator' interface within a DAW. The top bar shows the track name 'Chilea' and a 'Tweak' button. The main interface is divided into several sections:

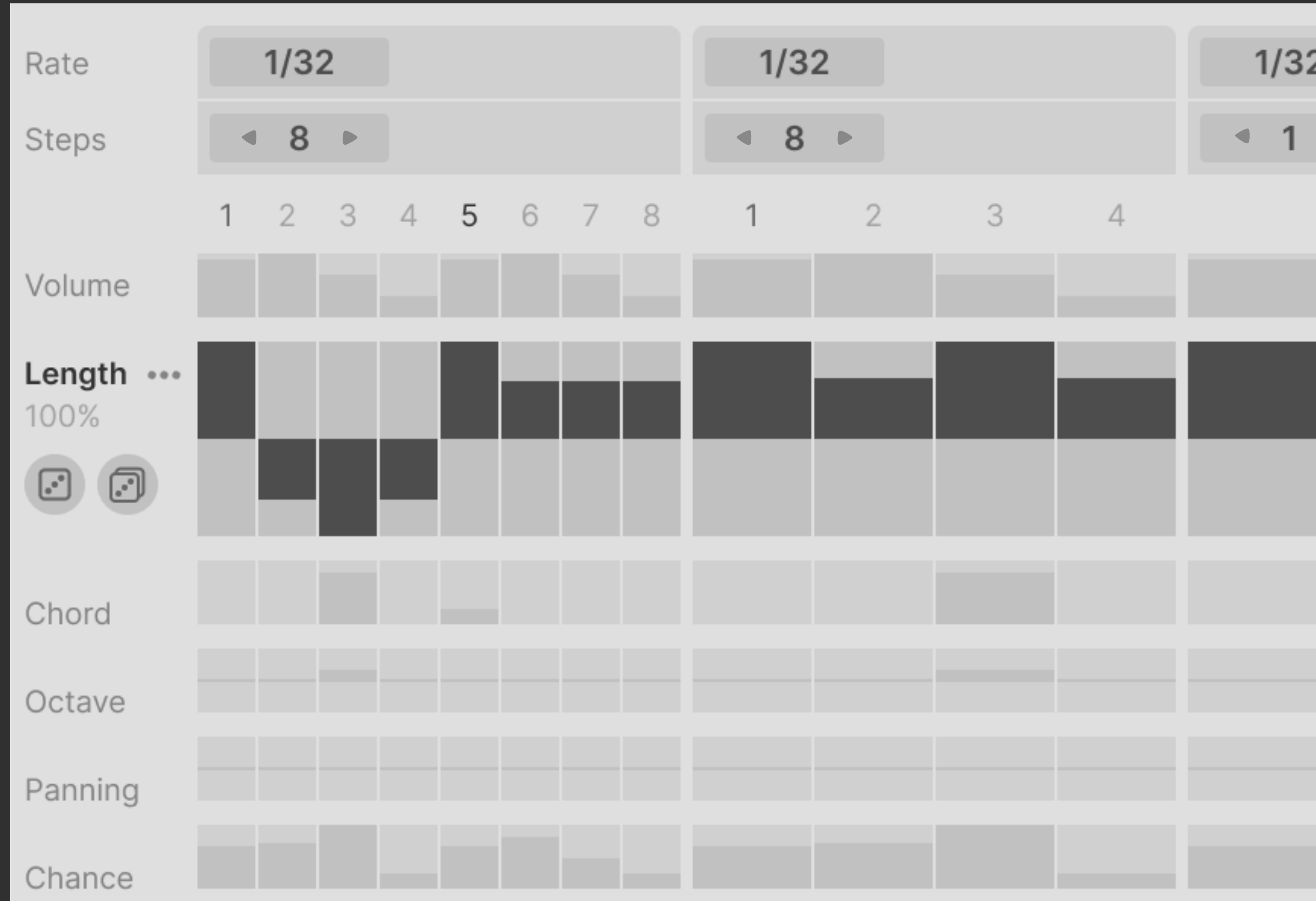
- Layer A:** Cello Vibrato Pizz Stab. Parameters include Vol, Speed (x2), Reps, Swing, and TimeSig (3/4). A 'Type' dropdown is set to 'Up-Down', and a 'Pedal' section has 'LOW', 'OFF', and 'HIGH' options.
- Arpeggiator Panel:**
 - Preset:** Patternoids
 - Rate:** 1/32
 - Steps:** 8
 - Volume:** Visual bar graph
 - Length:** 100%
 - Chord:** Visual bar graph
 - Octave:** Visual bar graph
 - Panning:** Visual bar graph
 - Chance:** Visual bar graph
 - Offset:** Volume, Length, Chord, Octave, Panning, Chance (each with a knob)

At the bottom, a piano roll shows notes for C1 through C7, with C4 and C5 highlighted in yellow.

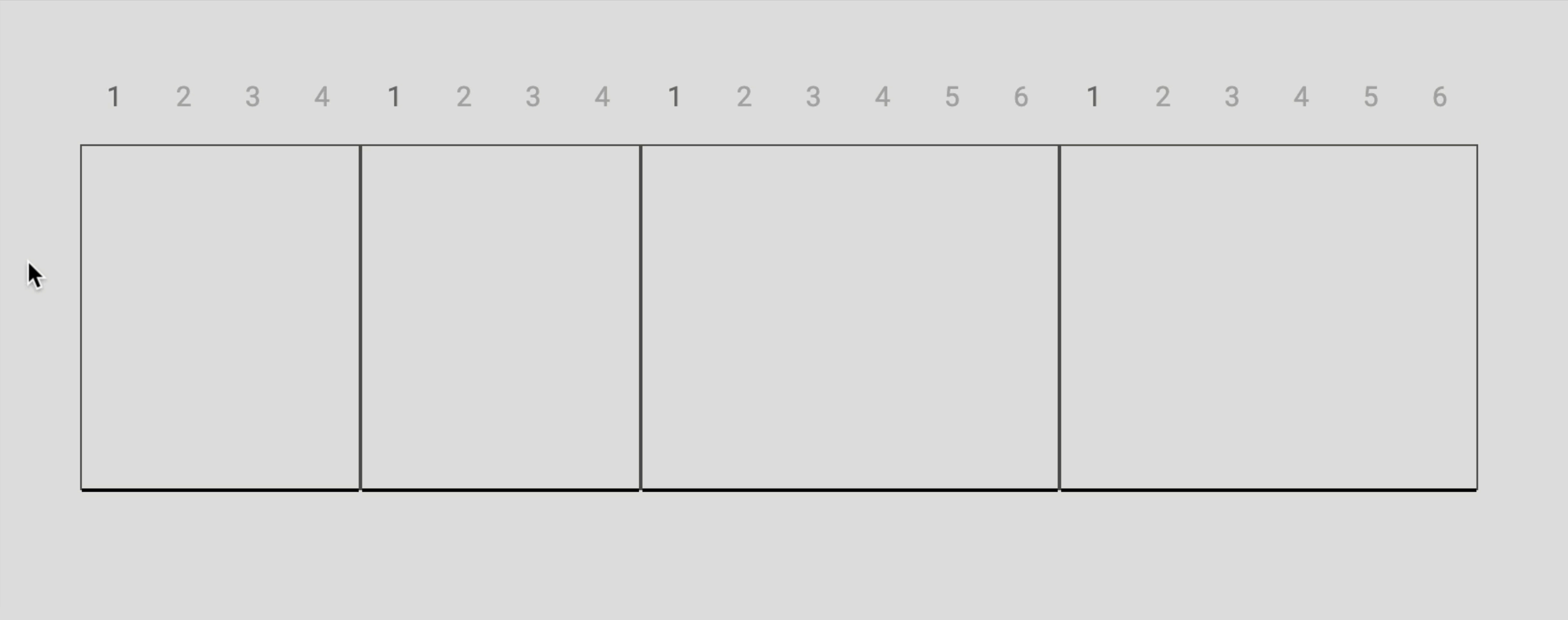
Case Study: Arpeggiator



Case Study: Arpeggiator



Case Study: Arpeggiator



Case Study: Arpeggiator


```
export interface IStep {  
  volume_01: number;  
  length_01: number;  
  octave: number;  
  chordProbability_01: number;  
  panning_11: number;  
  chance_01: number;  
}
```

```
export interface IChunk {  
  rate: number;  
  numActiveSteps: number;  
  steps: IStep[];  
}
```

```
export interface ISequence {  
  chunks: IChunk[];  
}
```

“This thing is amazing and not like anything else out there. It has everything I want in an arp mode.”

Nestor Estrada - Sound Designer, Output

 **MIDI**



1. Prototyping
2. Why Web Tech?
3. Case Study: Arpeggiator
- 4. C++ / WebViews**

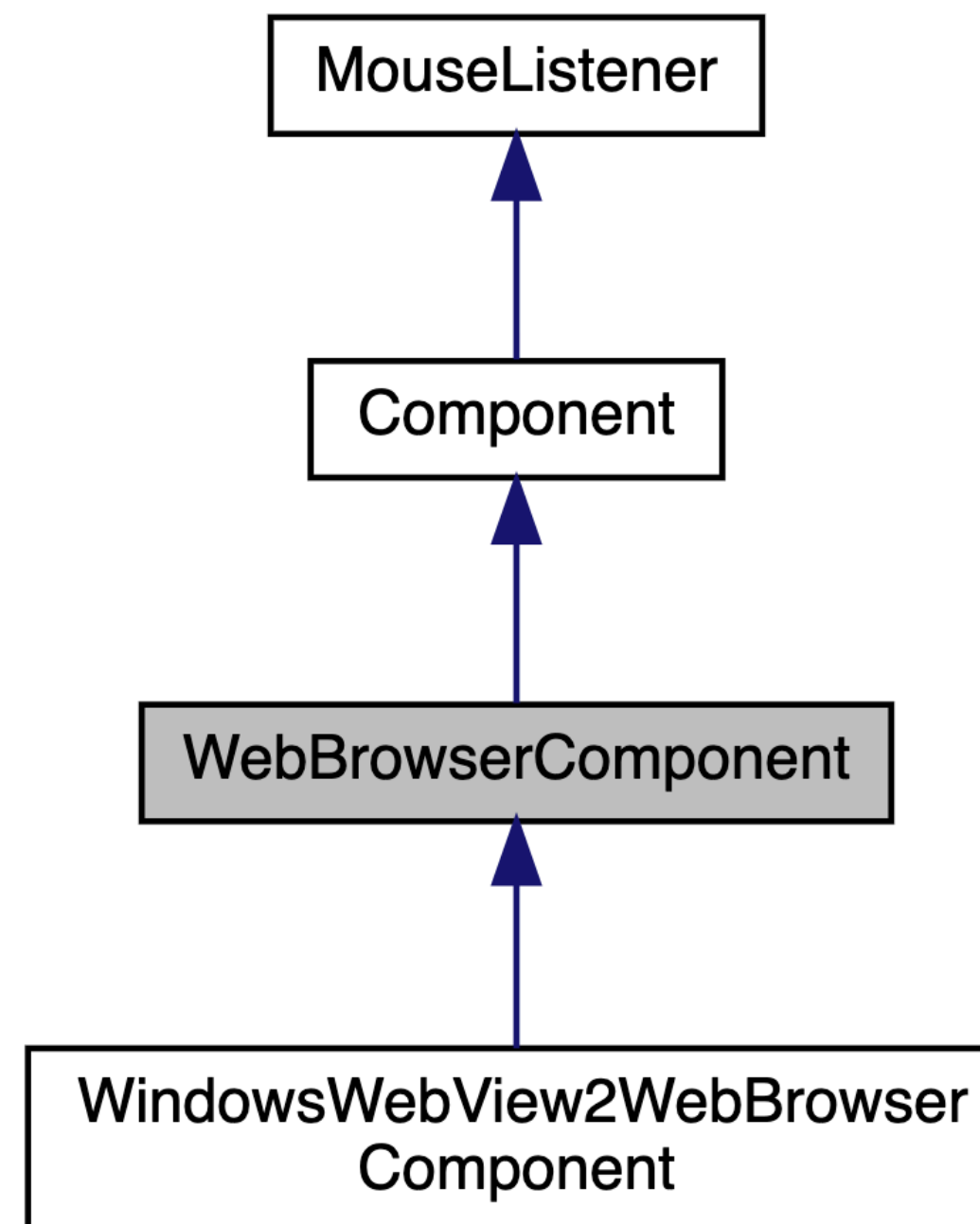




WebBrowserComponent Class Reference

[juce_gui_extra](#) » [misc](#)

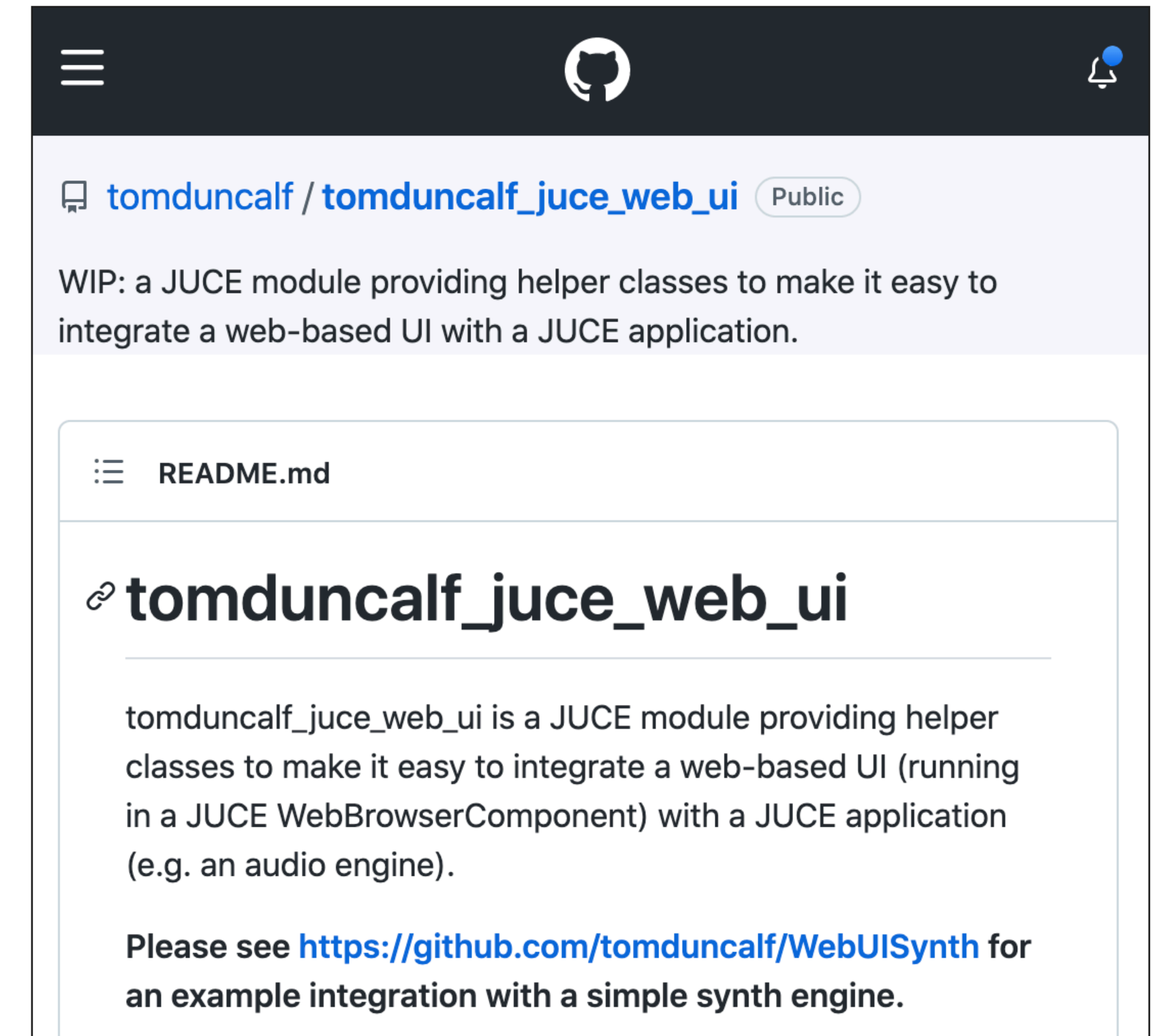
A component that displays an embedded web browser.



C++ ↔ JavaScript Communication



<https://youtube.com/watch?v=XvsCaQd2VFE>



https://github.com/tomduncalf/tomduncalf_juce_web_ui



quicktype

```
{
  "Note": {
    "type": "object",
    "properties": {
      "noteNumber": { "type": "integer" },
      "start_s": { "type": "number" },
      "duration_s": { "type": "number" },
      "velocity_01": { "type": "number" }
    },
    "required": [
      "noteNumber",
      "start_s",
      "duration_s",
      "velocity_01"
    ]
  }
}
```

JSON Schema

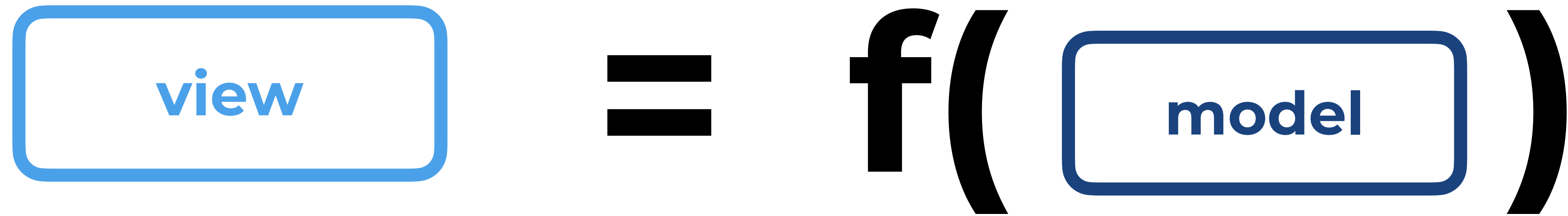
```
struct Note {
    double durationS;
    int64_t noteNumber;
    double startS;
    double velocity01;
};
```

C++

```
interface Note {
    duration_s: number;
    noteNumber: number;
    start_s: number;
    velocity_01: number;
}
```

TypeScript


```
namespace nlohmann {  
    void from_json(const json & j, MyApp::Note& x) {  
        x.durationS = j.at("duration_s").get<double>();  
        x.noteNumber = j.at("noteNumber").get<int64_t>();  
        x.startS = j.at("start_s").get<double>();  
        x.velocity01 = j.at("velocity_01").get<double>();  
    }  
  
    void to_json(json & j, const MyApp::Note & x) {  
        j = json::object();  
        j["duration_s"] = x.durationS;  
        j["noteNumber"] = x.noteNumber;  
        j["start_s"] = x.startS;  
        j["velocity_01"] = x.velocity01;  
    }  
}
```




```
export interface IState {
    tempo: number;
    activeLayer: number;
    inputNotes: { noteNumber: number; velocity_n: number };
    arpType: "Up" | "Down";
    layers: Array<{
        id: string;
        name: string;
        isEnabled: boolean;
        adders: {
            volume_n: number;
            length_n: number;
            octave: number;
            chordProbability_n: number;
            panning: number;
            chance_n: number;
        };
    };
    sequence: Array<{
        rate: number;
        numActiveSteps: number;
        steps: Array<{
            volume_n: number;
            length_n: number;
            octave: number;
            chordProbability_n: number;
            panning: number;
            chance_n: number;
        }>;
    }>;
    activeStepProperty: "volume_n" | "length_n" | "octave" | "chordProbability_n" | "panning" | "chance_n";
    speed: number;
    reps: number;
    timeSig: {
        numerator: number;
        denominator: number;
    };
    swing: number;
    randomMode: "current" | "all";
}>;
}
```

Unidirectional data flow (simplified)

model

view



`render (state) ;`



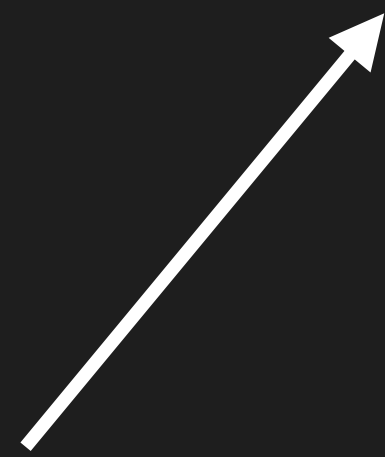
`update (newState) ;`



`sendMessage (type, data) ;`



save



load



```
localStorage.setItem("viewState",  
    JSON.stringify(viewState));
```

```
JSON.parse(localStorage.getItem("viewState"));
```

onViewStateChanged

onPageLoad

thi.ng/umbrella

typescript datastructure 2d clojure 3d workshop array binary
geometry browser animation clojurescript codegen graph c canvas functional
processing.org visualization toxiclibs ui webgl export fileformat graphics math svg conversion
dsl hiccup literate-programming tree audio interval typedarray uk acceleration async bezier color
generative hardware iterator java networked parser transducer cli clipping design image random bbox circle
fabrication interpolation mesh simulation baremetal component glsl nd particles query rstream architecture dsp
generator polygon shader stm32 stream analysis ast composition database gui physics reactive sample sort vector
agent align declarative format gpgpu html line matrix voxel wasm adjacency arc channel dataflow distance filter gradient
memory-mapped points shape subdivision typography alpha aos benchmark chart css diff event logger polyline rgb string 16bit 1d
8bit ansi area bigint blend compiler data-oriented dom encode fuzzy grid hdom immutable intersection noise opencl opengl oscillator
pipeline pixel rect render semweb serbia server text topology triples 24bit bitwise cache cartesian cellular-automata circumcenter
concatenative cubic dag fft forth germany graphviz hash language logic macro node object paper quaternion recursive scenegraph sculpture
sensor state theme union xml 32bit allocator ascii assembly bitmap blit branding convex curve datalog decode dependency drawing dual equality

In conclusion...

Do prototyping

Value iteration time

Slides and references

arthurcarabott.com/adc-2021

Contact (we are hiring)

arthur.carabott@output.com



@acarabott