Flocking

creative audio synthesis for the web

Colin Clark

Inclusive Design Research Centre,
OCAD University

Me, quick.

FLOCKING

flockingjs.org

github.com/colinbdclark/flocking

FLOCKING

- Audio synthesis framework written entirely in JavaScript
- Dedicated specifically to supporting artists and musicians, not gaming or industry
- Inspired by SuperCollider, but increasingly different
- Very open: dual MIT/GPL license

Motivations for Flocking

- The ubiquity of the Web
- The unresolved either/or of coding environments vs. graphical tools
- “Dead end” arts programming tools for beginners
- Inadequacy of current web-based tools for high-quality, long-term music-making

The Web is Huge

- Unprecedentedly cross-platform
- Huge community of programmers
- Solid tooling
- Flexible UI presentation layer and lots of toolkits available to choose from
- Performance war

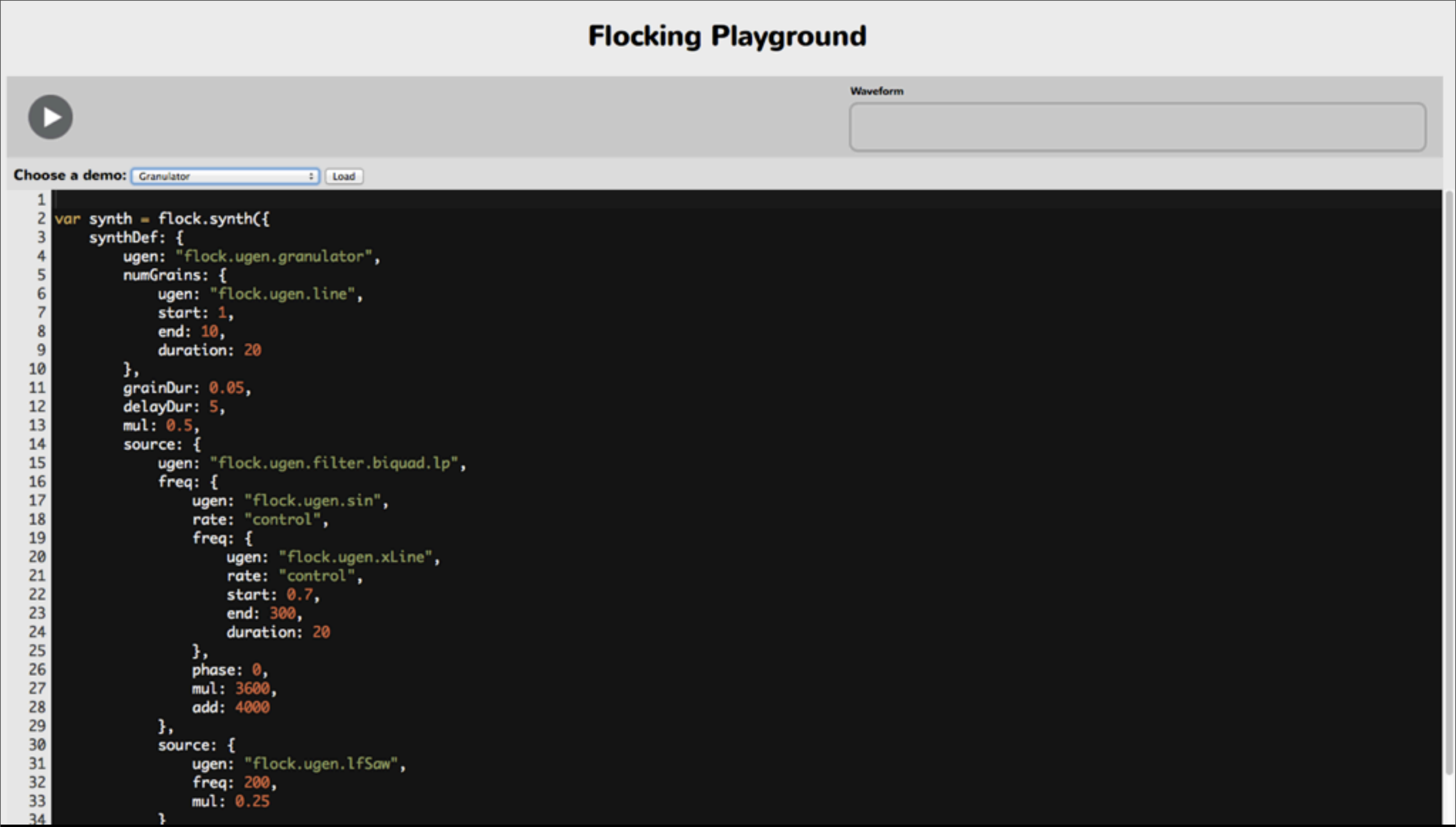
Where Does Flocking Run?

Browsers & Runtimes

- Firefox
- Chrome
- Safari
- Node.js

Operating Systems

- Mac OS X
- Windows
- Linux
- iOS
- Android *(last I checked!)*



Flocking Playground

Waveform

Choose a demo: Granulator Load

```
var synth = flock.synth({
    synthDef: {
        ugen: "flock.ugen.granulator",
        numGrains: {
            ugen: "flock.ugen.line",
            start: 1,
            end: 10,
            duration: 20
        },
        grainDur: 0.05,
        delayDur: 5,
        mul: 0.5,
        source: {
            ugen: "flock.ugen.filter.biquad.lp",
            freq: {
                ugen: "flock.ugen.sin",
                rate: "control",
                freq: {
                    ugen: "flock.ugen.xLine",
                    rate: "control",
                    start: 0.7,
                    end: 300,
                    duration: 20
                },
                phase: 0,
                mul: 3600,
                add: 4000
            },
            source: {
                ugen: "flock.ugen.lfSaw",
                freq: 200,
                mul: 0.25
            }
```

flockingjs.org/demos/interactive/html/playground.html

Flocking is Declarative

- Unit generators provide a consistent abstraction for operations on signals
- Synthesis graphs built up by declaring trees of named unit generators
- You write data, not code
- Data can be easily parsed, manipulated transformed, saved, authored, and edited by third-parties.

infusion

A Synth

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.sinOsc",
        freq: 440,
        mul: 0.25
    }
});
```

JavaScript & JSON

- JavaScript isn't a toy language any more
- Simple feature set, powerful first class functions and extremely loose typing
- JavaScript Object Notation: increasingly a standard, light format for data exchange

Object Literals

```
{
    “key”: "value"
}
```

Object Literals

```
{
    “key”: "value",
    number: 42.0,
    isLoud: true,
    method: function () { ... }
}
```

Array Literals

```
[“tenney”, “risset”, “schmickler”]
```

A JSON SynthDef

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.sinOsc",
        freq: 440,
        mul: 0.25
    }
});
```

A Unit Generator Def

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.sinOsc",
        freq: 440,
        mul: 0.25
    }
});
```

Inputs

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.sinOsc",
        freq: 440,
        mul: 0.25
    }
});
```

Rates

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.sinOsc",
        rate: “audio”,
        freq: 440,
        mul: 0.25
    }
});
```

Input Modulation

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.sinOsc",
        rate: “audio”,
        freq: 440,
        mul: {
            ugen: “flock.ugen.line”,
            rate: “control”,
            start: 0.0,
            end: 0.5,
            duration: 2.0
        }
    }
});
```

Expanded Form

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.out",
        bus: 0,
        sources: [{
            ugen: "flock.ugen.sinOsc",
            freq: 440,
            mul: 0.25
        }, {
            ugen: "flock.ugen.impulse",
            freq: 2,
            phase: 1.0
        }]
    }
});
```

Buffers

```
flock.synth({
    synthDef: {
        ugen: "flock.ugen.triggerGrains",
        buffer: {
            id: "beethoven",
            url: "../andante.aif"
        },
        trigger: {
            ugen: "flock.ugen.impulse",
            freq: 2
        },
        centerPos: 10,
        start: 0.01,
        end: 0.69,
        reset: 0.01
    }
});
```

Scheduling

- Scheduling in Flocking is currently asynchronous and "pleasantly unreliable"
- Sample accurate scheduler coming this summer
- Increasingly, the goal is use the Synth/UGen abstraction for scheduling patterns and generative algorithms
- JSON-based score format is evolving

Named Synth

```
flock.synth({
    nickName: "sin-synth",
    synthDef: {
        id: "carrier",
        ugen: "flock.ugen.sinOsc",
        freq: 440,
        mul: {
            ugen: "flock.ugen.line",
            start: 0,
            end: 0.25,
            duration: 1.0
        }
    }
});
```

Once

```
var scheduler = flock.scheduler.async();
scheduler.once(5, {
    synth: “sin-synth”,
    values: {
        “carrier.freq”: 440
    }
});
```

Once

```
var scheduler = flock.scheduler.async();
scheduler.once(5, {
    synth: “sin-synth”,
    values: {
        “carrier.freq”: 440
    }
});
```

Repeat

```
scheduler.repeat(1/16, function () {
    var freq = synth.get("carrier.freq"),
        newFreq = freq > 20000 ? 440 : freq * 7/6;

    synth.set("carrier.freq", newFreq);
});
```

Repeat

```
scheduler.repeat(1/16, function () {
    var freq = synth.get("carrier.freq"),
        newFreq = freq > 20000 ? 440 : freq * 7/6;

    synth.set("carrier.freq", newFreq);
});
```

Synth Patterns

```
var freqs = [110, 220, 330, 440, 550, 660, 880];
scheduler.schedule([
    {
        interval: "repeat",
        time: 0.25,
        change: {
            synth: "sin-synth",
            values: {
                "carrier.freq": {
                    synthDef: {
                        ugen: "flock.ugen.sequence",
                        loop: 1.0,
                        buffer: freqs
                    }
                }
            }
        }
    }
]);
```

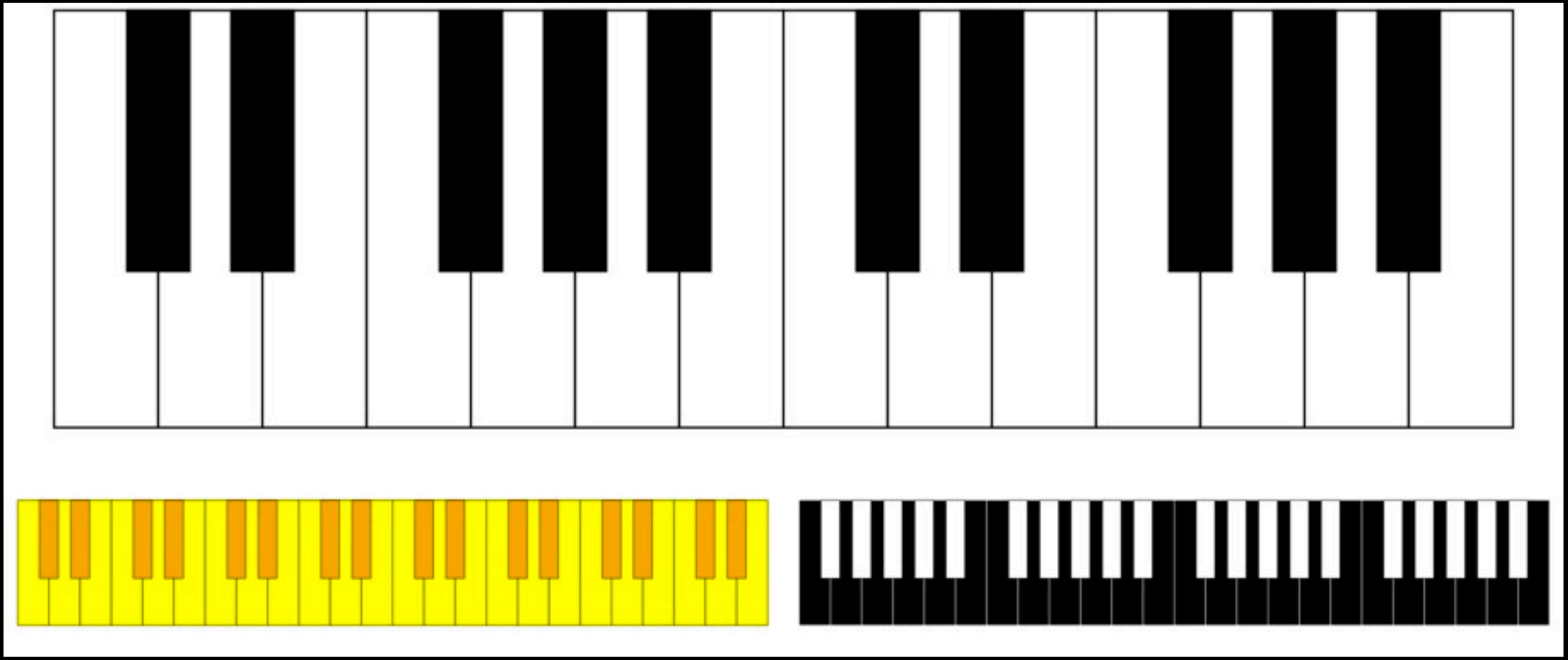
“Score”

```
scheduler.schedule([
    {
        interval: "repeat", time: 1.0,
        change: {
            synth: "sin-synth",
            values: {
                "carrier.freq": {
                    synthDef: {
                        ugen: "flock.ugen.sequence",
                        buffer: [110, 220, 330, 440, 550, 660, 880]
                    }
                }
            }
        }
    },
    {
        interval: "once", time: 8,
        change: {
            synth: "sin-synth",
            values: {
                "carrier.mul.start": 0.25,
                "carrier.mul.end": 0.0,
                "carrier.mul.duration": 1.0
            }
        }
    }
]);
```

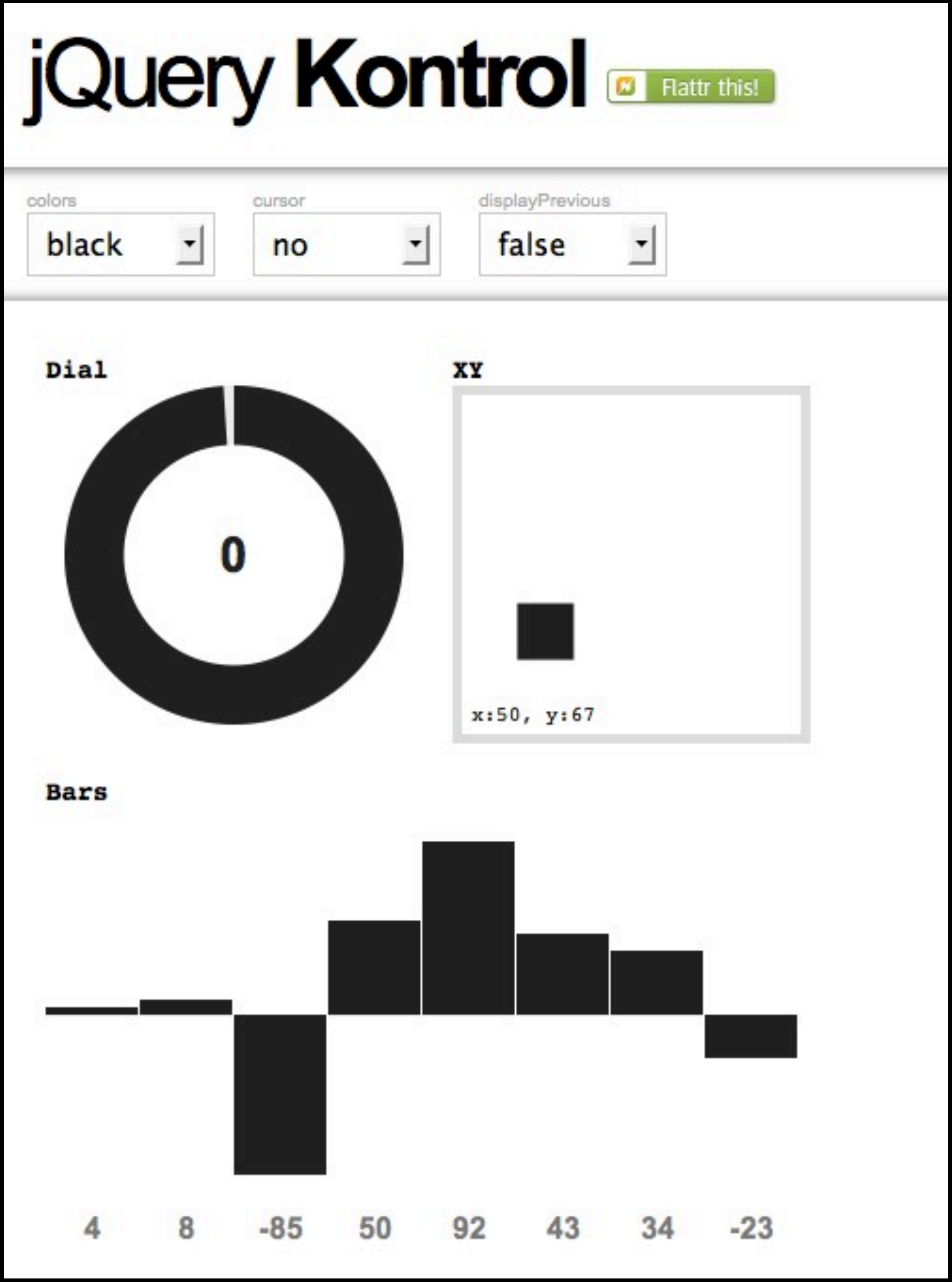
The State of Web Audio

- W3C Web Audio API and the dominance of Google
- Other libraries:
  - Timbre.js
  - Audiolib.js
  - Audiolet
- Performance directions

UI Controls



github.com/thealphanerd/Piano



colors: black
cursor: no
displayPrevious: false

x:50, y:67

github.com/aterrien/jQuery-Kontrol

Roadmap/Help!

- More unit generators!
- Finish and stabilize declarative scheduling
- Google Summer of Code: *Inclusive Music IDE*
- Full multichannel support
- MediaStream/WebRTC integration
- Node.js, OSC, WebSockets and REST
- Faust > Flocking unit generators (Myles)
- More music!

Questions?

Colin Clark

e: colin@colinclark.org

t: @colinbdclark

flockingjs.org

github.com/colinbdclark/flocking