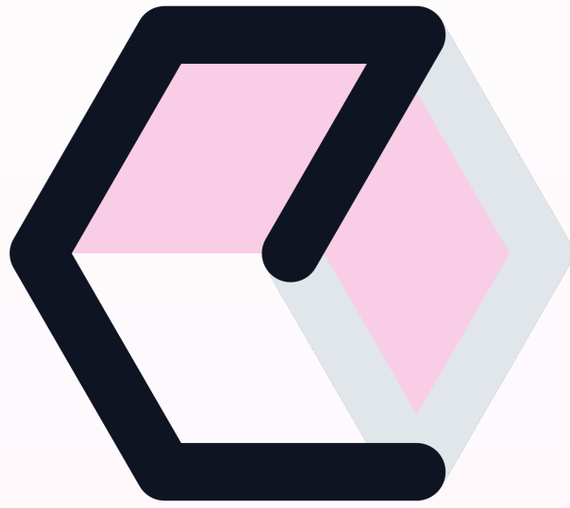
Why the Future of Audio is Functional

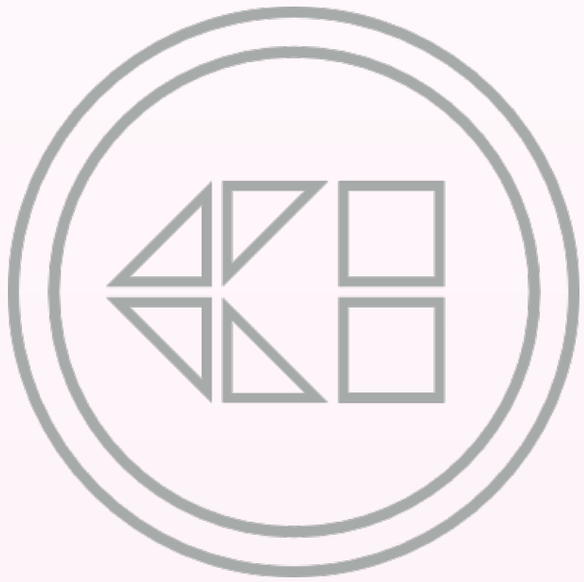
Nick Thompson

- Elementary Audio
- Creative Intent
- React-JUCE



https://www.nickwritesablog.com/

https://github.com/nick-thompson/



Setting the Scene

I believe that we can drastically simplify the process of writing audio software.

Setting the Scene

We can address huge amounts of software complexity, and drastically improve our development speed.

Setting the Scene

We can make the field of audio programming more accessible.

Agenda

How do we simplify the process of writing audio software?

- **Comparison:** declarative, mathematical expression to the conventional model for writing audio software

- **Intuition:** what problem are we solving? How do we articulate the answer?

- **Conclusion:** Elementary Audio

Comparison: Example 1

$$f(x) = lowpass(800Hz,\ 0.717,\ x)$$

Comparison: Example 1

```
class Processor {
public:
    // Called on the main thread
    void prepare (double sampleRate, int blockSize);

    // Called on the realtime thread
    void processBlock (AudioBlock& block);

private:
    BiquadFilter bq;
};
```

Comparison: Example 1

```
class Processor {
public:
    Processor()
      : bq(1, 0, 0, 0, 0) {}

    // Called on the main thread
    void prepare (double sampleRate, int blockSize);

    // Called on the realtime thread
    void processBlock (AudioBlock& block);

private:
    BiquadFilter bq;
};
```

Comparison: Example 1

```
class Processor {
public:
    Processor()
      : bq(1, 0, 0, 0, 0) {}

    // Called on the main thread
    void prepare (double sampleRate, int blockSize) {
        bq.prepare(sampleRate, blockSize);
    }

    // Called on the realtime thread
    void processBlock (AudioBlock& block);

private:
    BiquadFilter bq;
};
```

Comparison: Example 1

```
class Processor {
public:
    Processor()
      : bq(1, 0, 0, 0, 0) {}

    // Called on the main thread
    void prepare (double sampleRate, int blockSize) {
        bq.prepare(sampleRate, blockSize);
    }

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        bq.processBlock(block);
    }

private:
    BiquadFilter bq;
};
```

Comparison: Example 2

$$f(x) = lowpass(800Hz,\ 0.717,$$
$$lowpass(800Hz,\ 0.717,$$
$$highpass(200Hz,\ 0.717,\ x)))$$

Comparison: Example 2

```
class Processor {
public:
    ...

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        for (int i = 0; i < bq.size(); ++i) {
            bq[i].processBlock(block);
        }
    }

private:
    std::array<BiquadFilter, 3> bq;
};
```

Comparison: Example 3

$$f(x) = lowpass(800Hz,\ 0.717,$$
$$lowpass(800Hz,\ 0.717,$$
$$highpass(200Hz,\ 0.717,$$
$$highpass(200Hz,\ 0.717,\ x))))$$

Comparison: Example 3

```
class Processor {
public:
    ...

    // Called on the main thread
    void addFilter (FilterType t) {
        bq.push_back(BiquadFilter(t));
    }

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        for (int i = 0; i < bq.size(); ++i) {
            bq[i].processBlock(block);
        }
    }

private:
    std::vector<BiquadFilter> bq;
};
```

Comparison: Example 3

```
class Processor {
    ...

    // Called on the main thread
    void addFilter (FilterType t) {
        const ScopedLock sl (callbackLock);
        bq.push_back(BiquadFilter(t));
    }

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        const ScopedLock sl (callbackLock);
        for (int i = 0; i < bq.size(); ++i) {
            bq[i].processBlock(block);
        }
    }

    ...
};
```

Comparison: Example 4

$$f(x) = lowpass(800Hz,\ 0.717,\ g(x))$$

$$where$$

$$g(x) = add($$

$$highpass(200Hz,\ 0.717,\ x),$$

$$peak(400Hz,\ 1.414,\ 3dB,\ x))$$

Comparison: Example 4

```
class Processor {
public:
    ...

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        scratchBlock.copyFrom(block);
        bq[0].processBlock(block);
        bq[1].processBlock(scratchBlock);
        block.addFrom(scratchBlock);
        bq[2].processBlock(block);
    }

private:
    AudioBlock scratchBlock;
    std::array<BiquadFilter, 3> bq;
};
```

Comparison: Example 4

```
class Processor {
public:
    // Called on the main thread
    void prepare (double sampleRate, int blockSize) {
        graph.addNode(std::make_unique<Biquad>());
        graph.addNode(std::make_unique<Biquad>());
        ...
        graph.prepareToPlay(sampleRate, blockSize);
    }

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        graph.processBlock(block);
    }

private:
    juce::AudioProcessorGraph graph;
};
```

Comparison: Example 4

```
class Processor {
public:
    ...

    // Called on the main thread
    void addFilter (FilterType t) {
        graph.addNode(std::make_unique<Biquad>(t));
        graph.removeConnection({4, 6});
        graph.addConnection({4, 8});
        graph.addConnection({4, 6});
    }

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        graph.processBlock(block);
    }

    ...
};
```

Comparison: Example 4

```
// Called on the main thread
void prepare (double sampleRate, int blockSize) {
    auto s = graph.addNode(std::make_unique<Input>());
    auto o = graph.addNode(std::make_unique<Output>());
    auto i = graph.addNode(std::make_unique<Biquad>());
    auto j = graph.addNode(std::make_unique<Biquad>());
    auto k = graph.addNode(std::make_unique<Biquad>());
    graph.getNodeForId(i)->setType(Biquad::Lowpass);
    graph.getNodeForId(j)->setType(Biquad::Highpass);
    graph.getNodeForId(k)->setType(Biquad::Peak);
    graph.addConnection(i, s);
    graph.addConnection(j, s);
    graph.addConnection(k, i);
    graph.addConnection(k, j);
    graph.addConnection(o, k);
    graph.prepare(sampleRate, blockSize);
}
```

Comparison: Example 4

$$f(x) = lowpass(800Hz,\ 0.717,\ g(x))$$

$$where$$

$$g(x) = add($$

$$highpass(200Hz,\ 0.717,\ x),$$

$$peak(400Hz,\ 1.414,\ 3dB,\ x))$$

Comparison: Example 4

```
// Called on the main thread
void prepare (double sampleRate, int blockSize) {
    auto s = graph.addNode(std::make_unique<Input>());
    auto o = graph.addNode(std::make_unique<Output>());
    auto i = graph.addNode(std::make_unique<Biquad>());
    auto j = graph.addNode(std::make_unique<Biquad>());
    auto k = graph.addNode(std::make_unique<Biquad>());
    graph.getNodeForId(i)->setType(Biquad::Lowpass);
    graph.getNodeForId(j)->setType(Biquad::Highpass);
    graph.getNodeForId(k)->setType(Biquad::Peak);
    graph.addConnection(i, s);
    graph.addConnection(j, s);
    graph.addConnection(k, i);
    graph.addConnection(k, j);
    graph.addConnection(o, k);
    graph.prepare(sampleRate, blockSize);
}
```

Comparison: Example 4

```
void handleUserUpdate () {
    auto l = graph.addNode(std::make_unique<Biquad>());
    graph.removeConnection(i, s);
    graph.removeConnection(j, s);
    graph.removeConnection(k, i);
    graph.addConnection(j, i);
    graph.addConnection(k, j);
    graph.addConnection(o, j);
}
```

Comparison: Example 5

$$f(x) = lowpass(800Hz,\ 0.717,\ g(x))$$

where

$$g(x) = \frac{tanh(4 * x)}{tanh(4)}$$

Comparison: Example 5

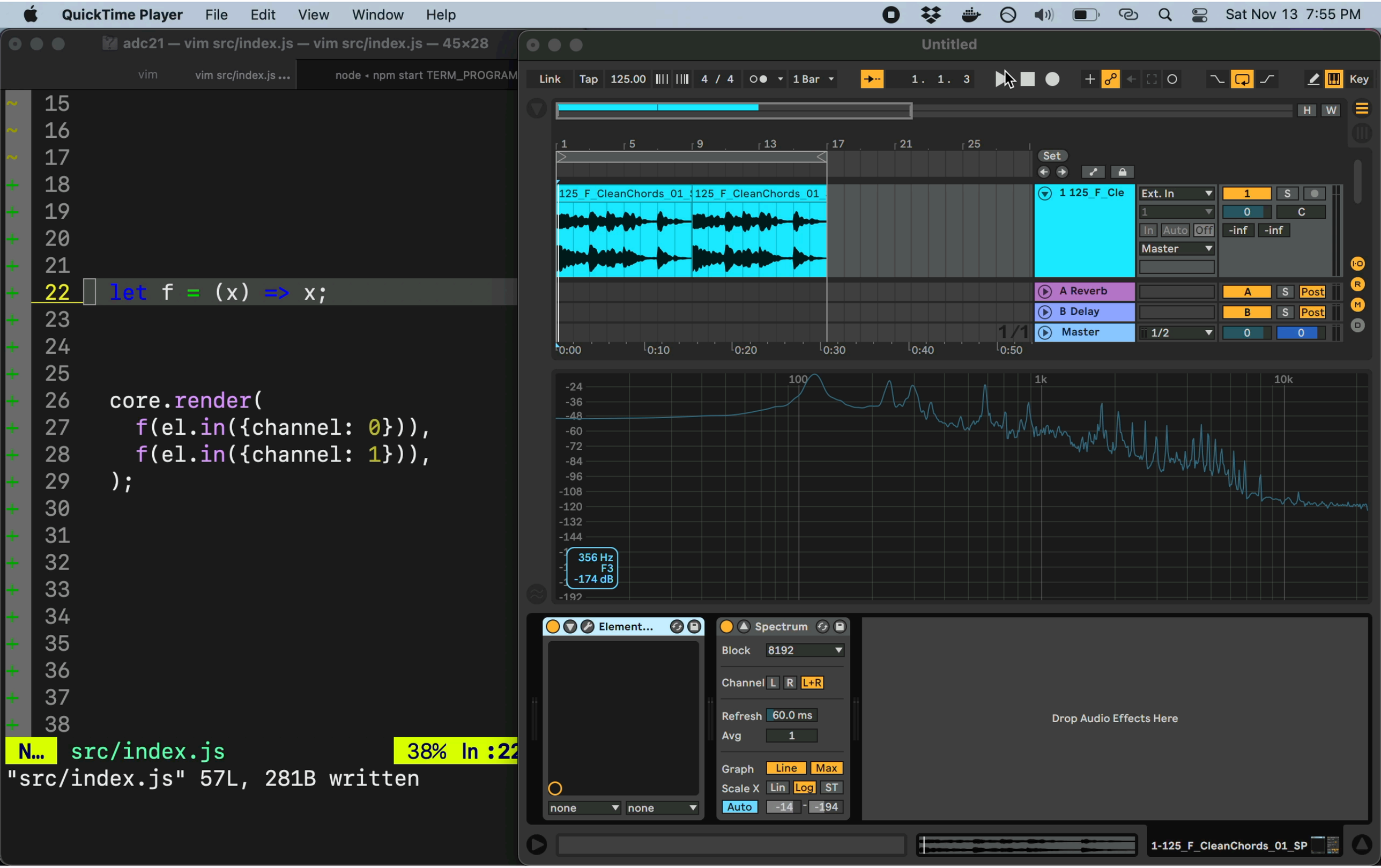
```
// Called on the main thread
void prepare (double sampleRate, int blockSize) {
    auto s = graph.addNode(std::make_unique<Input>());
    auto o = graph.addNode(std::make_unique<Output>());
    auto i = graph.addNode(std::make_unique<Biquad>());
    auto j = graph.addNode(std::make_unique<Saturator>());
    graph.getNodeForId(i)->setType(Biquad::Lowpass);
    graph.addConnection(i, s);
    graph.addConnection(j, i);
    graph.addConnection(o, j);
    graph.prepare(sampleRate, blockSize);
}
```

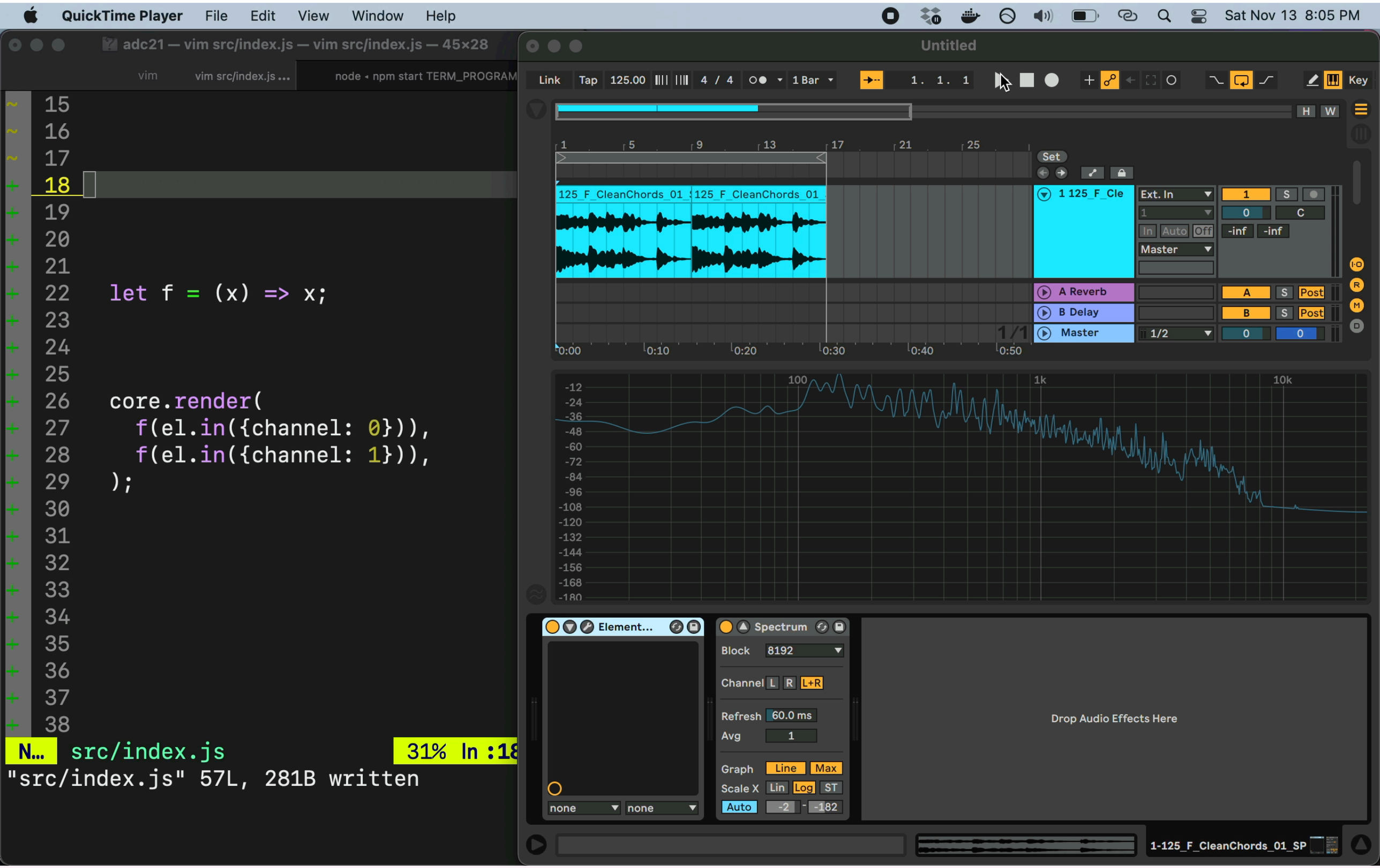
Comparison: Example 5

```
class Saturator {
public:
    ...

    // Called on the realtime thread
    void processBlock (AudioBlock& block) {
        auto* data = block.getChannelData();
        for (int i = 0; i < data.size(); ++i) {
            auto n = std::tanh(4 * data[i])
            data[i] = n / std::tanh(4);
        }
    }

};
```


QuickTime Player File Edit View Window Help
Sat Nov 13 7:55 PM
adc21 — vim src/index.js — vim src/index.js — 45×28
vim
vim src/index.js ...
node ◂ npm start TERM_PROGRAM
15
16
17
18
19
20
21
22 let f = (x) => x;
23
24
25
26 core.render(
27 f(el.in({channel: 0})),
28 f(el.in({channel: 1})),
29);
30
31
32
33
34
35
36
37
38
N... src/index.js 38% ln :22
"src/index.js" 57L, 281B written
Untitled
Link Tap 125.00 4 / 4 1 Bar 1 . 1 . 3 Key
H W
1 5 9 13 17 21 25
Set
125_F_CleanChords_01_ 125_F_CleanChords_01_
1 125_F_Cle
Ext. In
1
0
C
In Auto Off
-inf -inf
Master
A Reverb
B Delay
Master
1/2
Post
0:00 0:10 0:20 0:30 0:40 0:50
1/1
100 1k 10k
-24 -36 -48 -60 -72 -84 -96 -108 -120 -132 -144
356 Hz
F3
-174 dB
Element...
none none
Spectrum
Block 8192
Channel L R L+R
Refresh 60.0 ms
Avg 1
Graph Line Max
Scale X Lin Log ST
Auto -14 -194
Drop Audio Effects Here
1-125_F_CleanChords_01_SP


QuickTime Player File Edit View Window Help
Sat Nov 13 8:05 PM
adc21 — vim src/index.js — vim src/index.js — 45×28
vim
vim src/index.js ...
node ◂ npm start TERM_PROGRAM
15
16
17
18
19
20
21
22 let f = (x) => x;
23
24
25
26 core.render(
27 f(el.in({channel: 0})),
28 f(el.in({channel: 1})),
29);
30
31
32
33
34
35
36
37
38
N... src/index.js 31% ln :18
"src/index.js" 57L, 281B written
Untitled
Link Tap 125.00 4 / 4 1 Bar 1 . 1 . 1 Key
H W
1 5 9 13 17 21 25
Set
125_F_CleanChords_01_ 125_F_CleanChords_01_
1 125_F_Cle
Ext. In
1
0
C
In Auto Off
-inf -inf
Master
A Reverb
B Delay
Master
1/2
Post
0:00 0:10 0:20 0:30 0:40 0:50
1/1
100 1k 10k
-12 -24 -36 -48 -60 -72 -84 -96 -108 -120 -132 -144 -156 -168 -180
Element...
none none
Spectrum
Block 8192
Channel L R L+R
Refresh 60.0 ms
Avg 1
Graph Line Max
Scale X Lin Log ST
Auto -2 -182
Drop Audio Effects Here
1-125_F_CleanChords_01_SP

Live File Edit Create View Options Help

Sun Nov 14 9:03 PM

adc21 — vim src/index.js — vim src/index.js — 45×28

vim

node ◂ npm start TERM_PROGRAM

```
18
19
20
21
22  let f = (x) => s(lp(x));
23
24  let lp = (x) =>
25    el.lowpass(800, 0.717, x);
26
27  let s = (x) => el.mul(
28    0.25,
29    el.div(
30      el.tanh(el.mul(4, x)),
31      el.tanh(4),
32    )
33  );
34
35  core.render(
36    f(el.in({channel: 0})),
37    f(el.in({channel: 1})),
38  );
39
40
41
```

N… src/index.js 37% ln :25

"src/index.js" 66L, 442B written

Untitled

Link Tap 125.00 4 / 4 1 Bar 1. 2. 1 Key

H W

1 5 9 13 17 21 25

Set

125_F_CleanChords_01 125_F_CleanChords_01

1 125_F_Cle

Ext. In 1 S 0 C In Auto Off -inf -inf Master

A Reverb A S Post

B Delay B S Post

Master 1/2 0 0

1/1

0:00 0:10 0:20 0:30 0:40 0:50

-24 -36 -48 -60 -72 -84 -96 -108 -120 -132 -144 -156 -168 -180 -192

100 1k 10k

Element...

none none

Spectrum

Block 8192

Channel L R L+R

Refresh 60.0 ms

Avg 1

Graph Line Max

Scale X Lin Log ST

Auto -14 -194

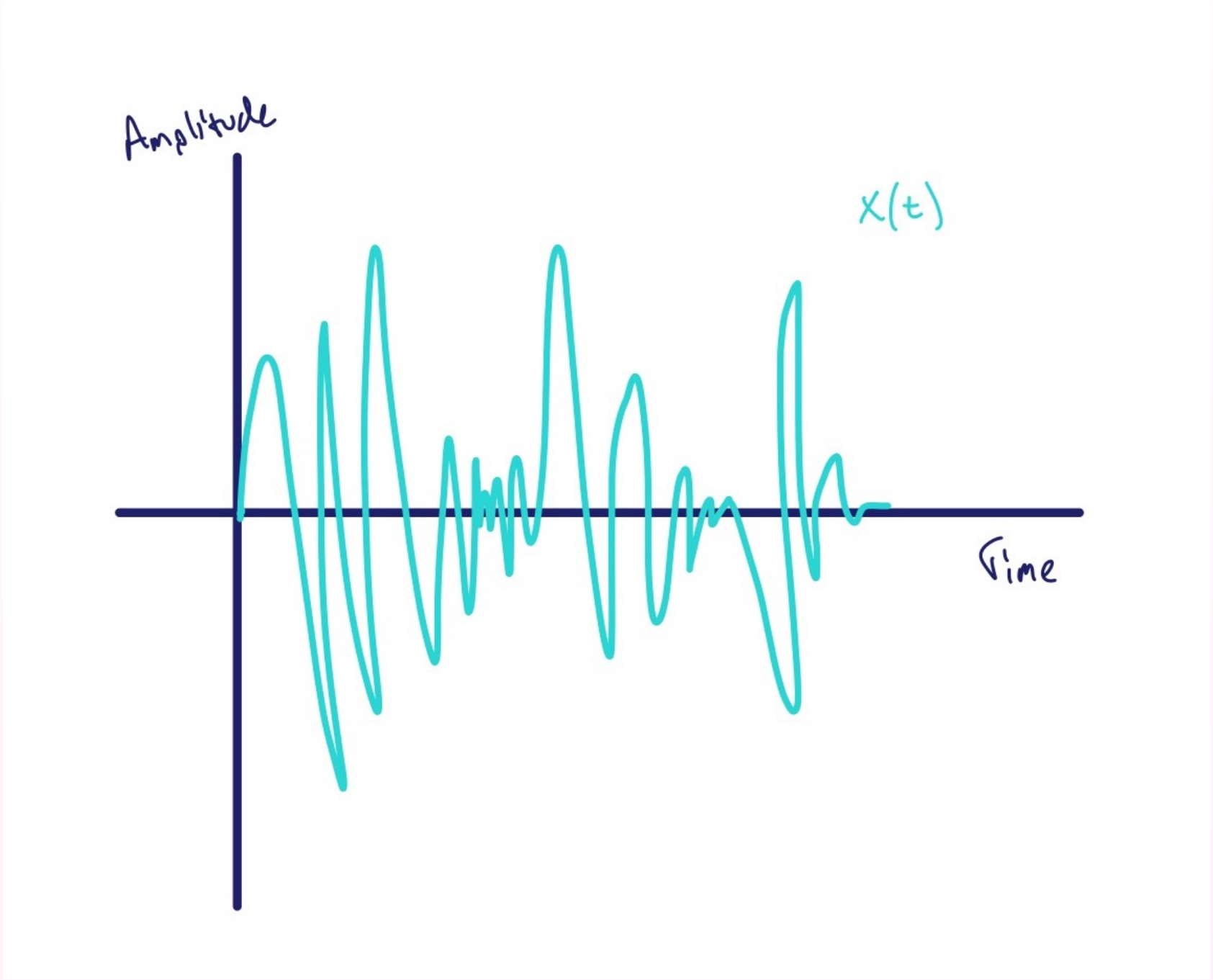
Drop Audio Effects Here

1-125_F_CleanChords_01_SP

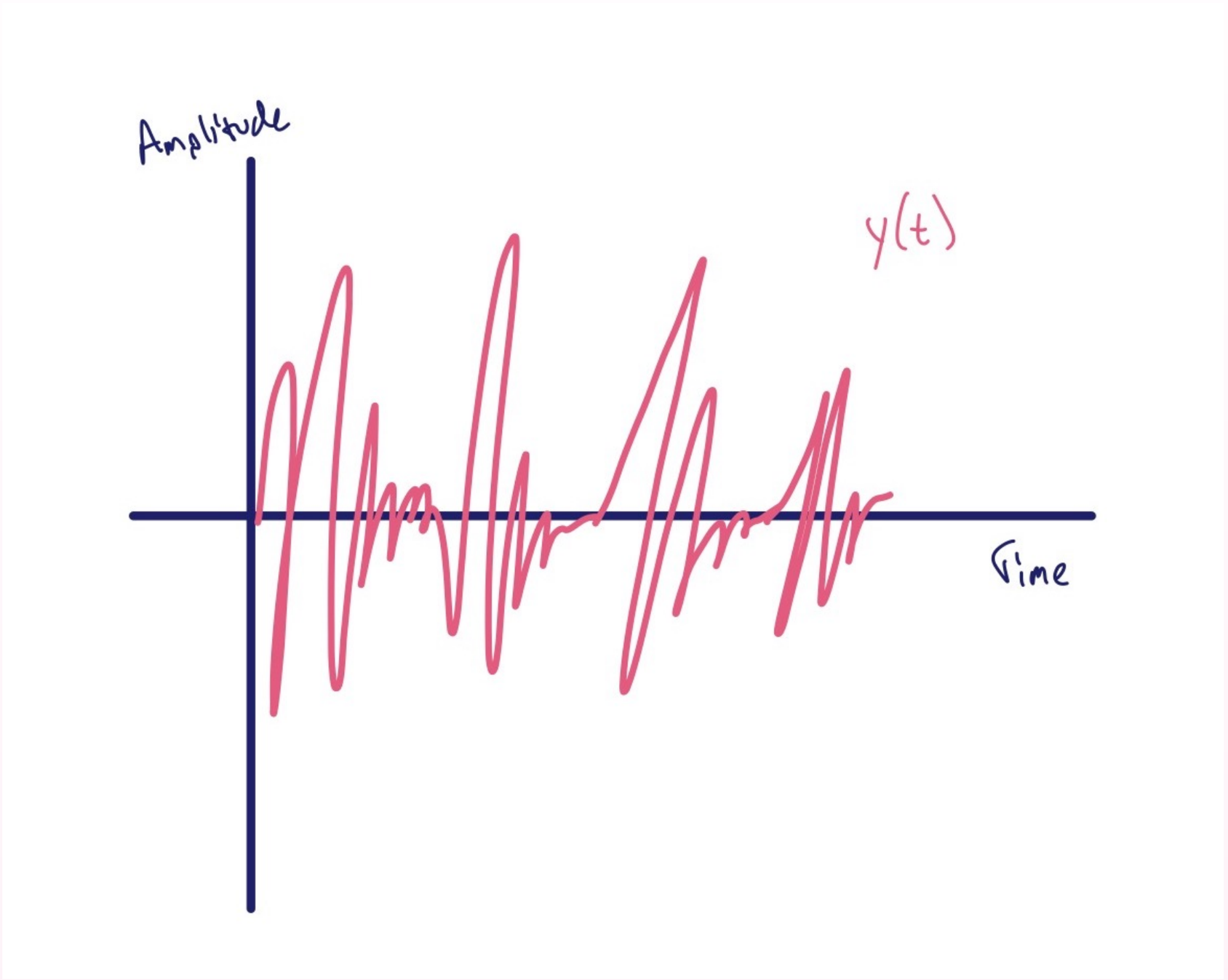
An Intuitive Approach

What problem are we solving?

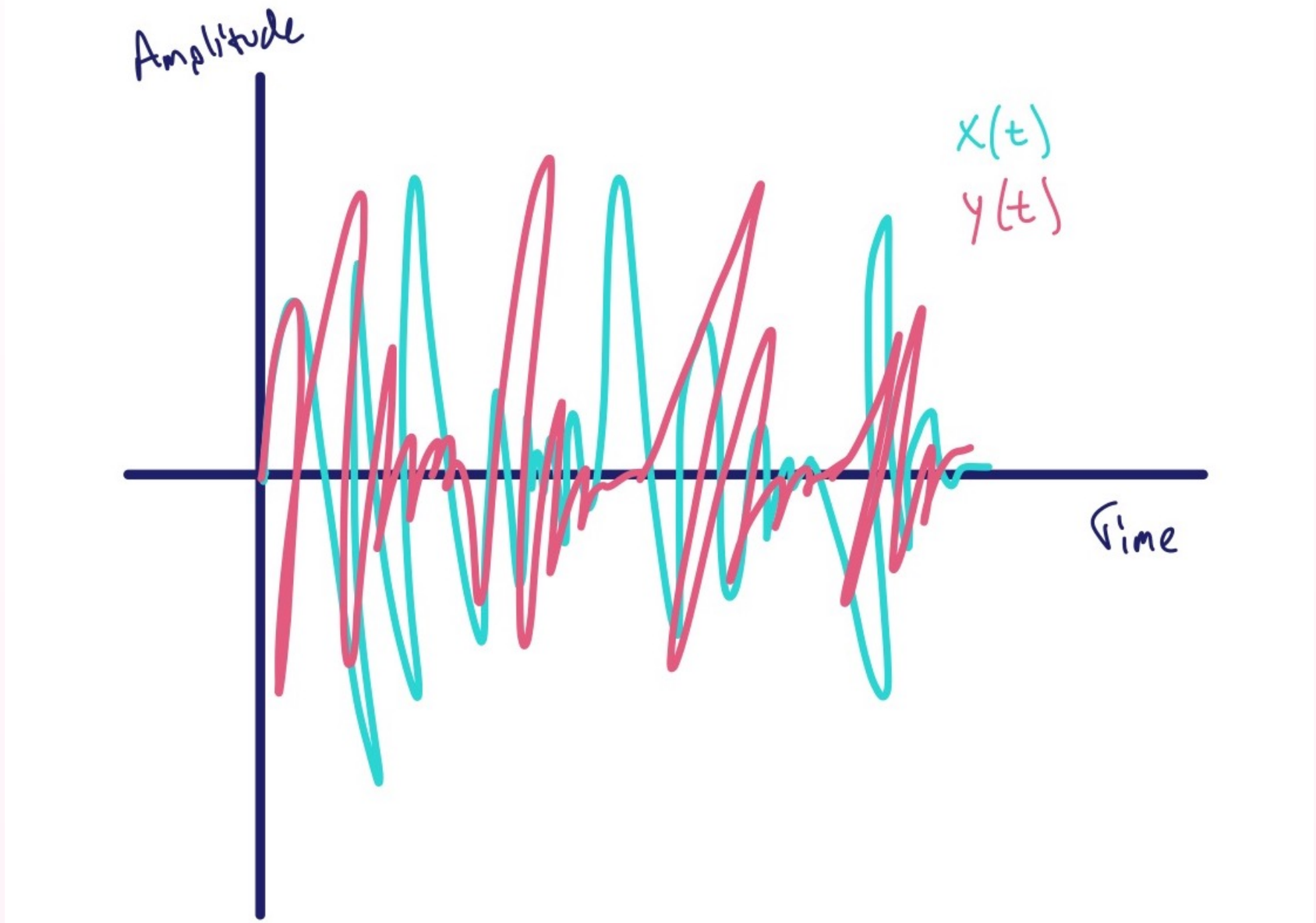
An Intuitive Approach



An Intuitive Approach



An Intuitive Approach



An Intuitive Approach

$$y(t) = f(x(t))$$

$$y(t) = f(x(t), t)$$

An Intuitive Approach

$$f(x) = lowpass(800Hz,\ 0.717,\ x)$$

Incidental Complexity

- Thread synchronization
- Memory management
- Discretization
- Deterministic & Non-Deterministic operations
- Lock-Free Programming
- ...

Incidental Complexity

- …

- Imperative programming?

An Intuitive Approach

"Imperative programming is a programming paradigm (way of designing a programming language) that describes computation in terms of the program state, and of the statements which change the program state."

https://en.wikipedia.org/wiki/State_(computer_science)

An Intuitive Approach

"In declarative programming languages, the program describes the desired results and doesn't specify changes to the state directly."

https://en.wikipedia.org/wiki/State_(computer_science)

An Intuitive Approach

"First, we want to establish the idea that a computer language is not just a way of getting a computer to perform operations but rather that it is a novel formal medium for expressing ideas about methodology."

Sussman and Abelson
Structure and Interpretation of Computer Programs
2nd Edition

An Intuitive Approach

"Thus, programs must be written for people to read, and only incidentally for machines to execute."

Sussman and Abelson
Structure and Interpretation of Computer Programs
2nd Edition

Elementary

Elementary Audio

A JavaScript framework for functional, declarative expression of realtime audio signal processes.

+

A native audio engine to deliver a high performance realization of the given audio process.

Elementary Audio

```
function saturate(p, x) {
  let a = el.tanh(el.mul(p, x));
  return el.div(a, el.tanh(p));
}

core.on('load', function(e) {
  let x = el.in({channel: 0});
  let y = el.lowpass(800, 1.414, saturate(4, x));
  core.render(y);
});
```

Elementary Audio

```
let state = {
  filters: [{type: 'lowpass', cutoff: 800, q: 1}],
  drive: 4,
};

function update(appState) {
  let x = el.in({channel: 0});
  let y = appState.filters.reduce((x, f) => (
     el[f.type](f.cutoff, f.q, x)
  ), saturate(appState.drive, x));

  core.render(y);
});

core.on('addFilter', (f) => update(addFilter(state, f));
```

Elementary Audio

- Portable: runs in your {browser, audio plugin, command line, etc...}
- Add your UI with {web frameworks, native widgets, react-juce, juce, etc...}
- Extensible
- Embeddable

Thank You

- Elementary Audio
  - https://www.elementary.audio/
  - nick@elementary.audio
- Me
  - https://www.nickwritesablog.com/
  - https://github.com/nick-thompson/