Why you Shouldn’t Write a DAW

Dave Rowland

CTO Audio Squadron

PRISM SOUND

tracktion

POWERED BY tracktion ENGINE

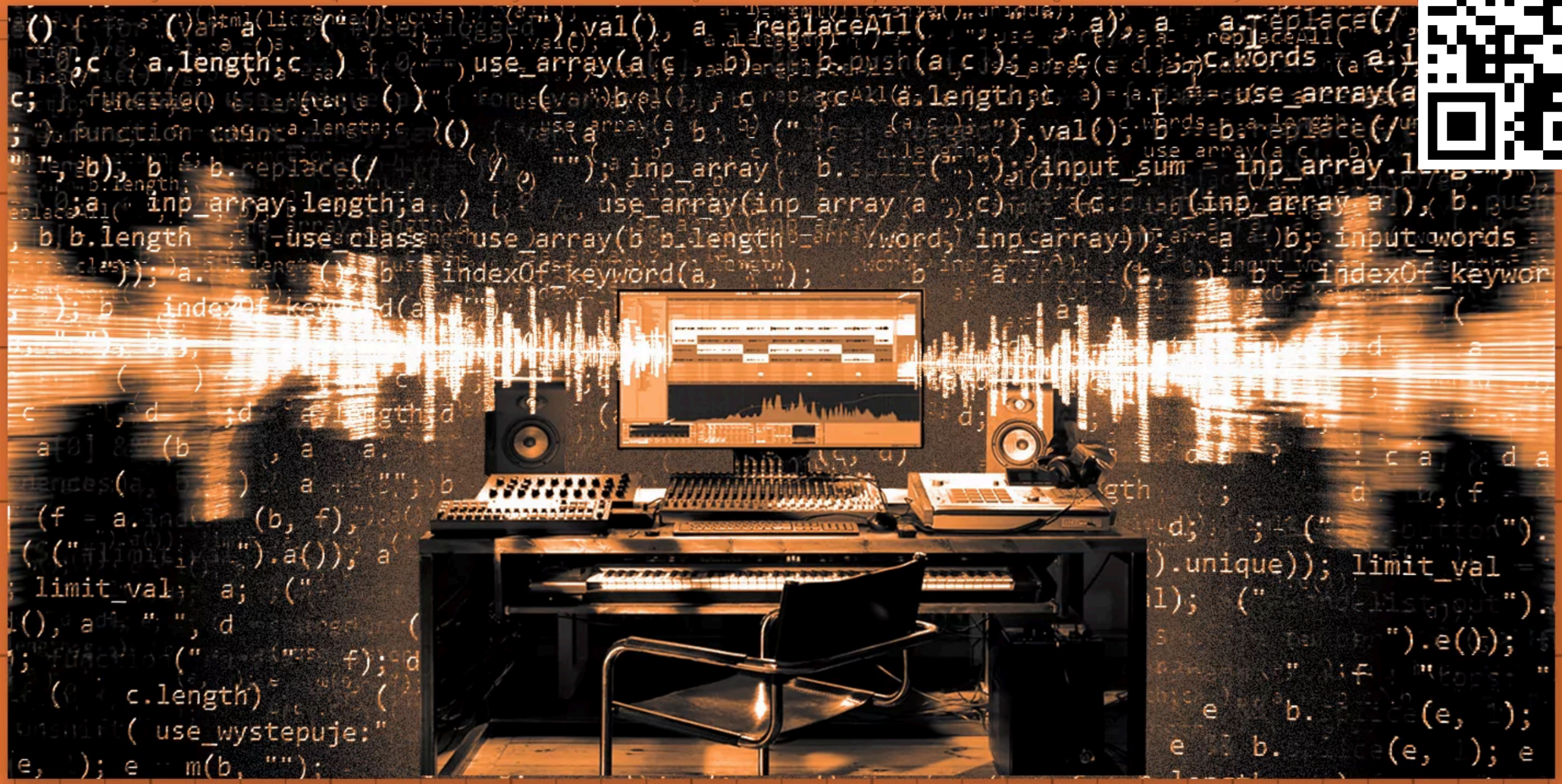
When is a DAW not a DAW?

When it’s a jar

“I can see why no one in their right mind would undertake [building a DAW]”

Lex Dromgoole, CEO of Bronze

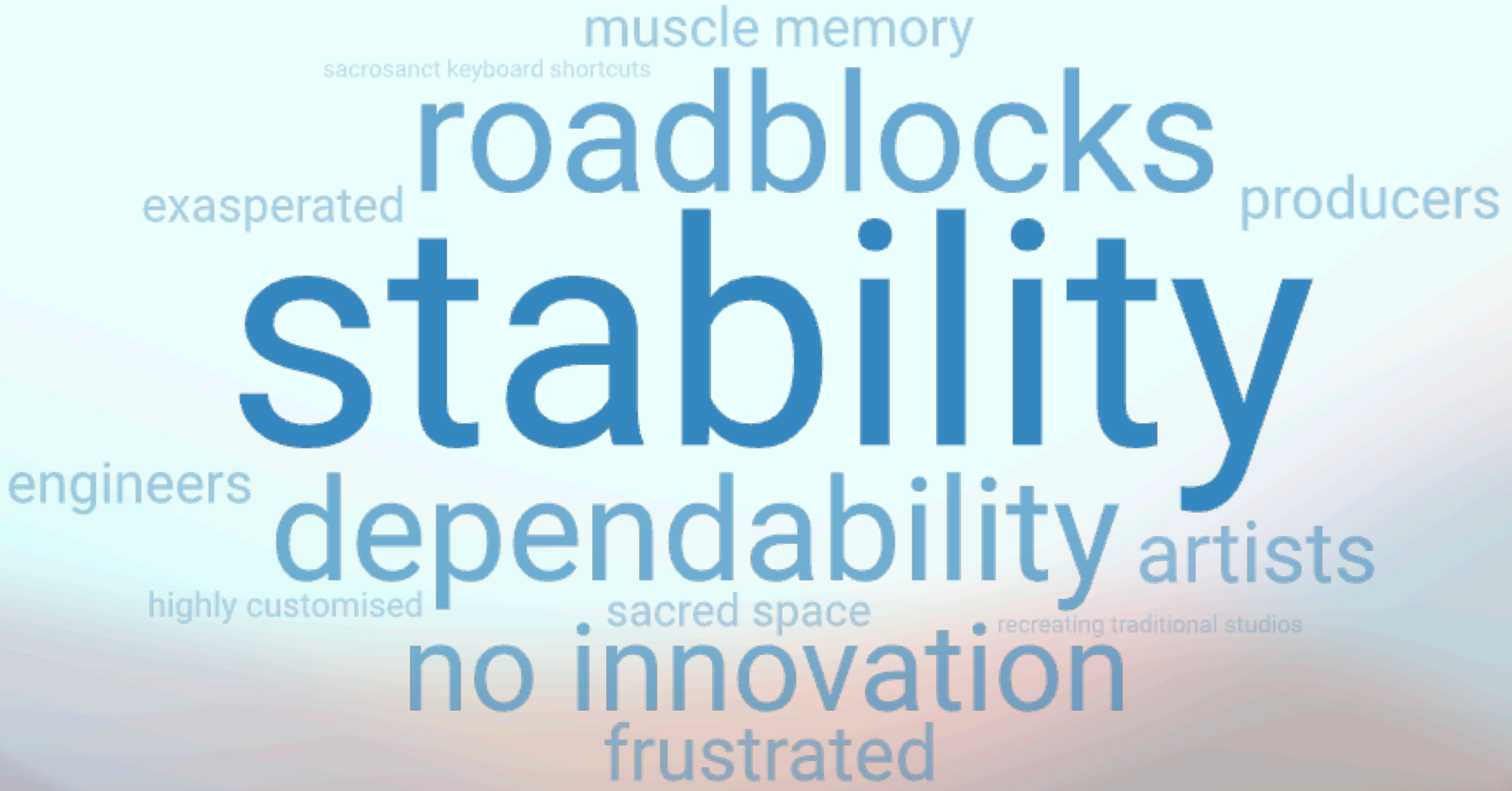
What is the future of the DAW?



djmag.com/features/what-future-daw

“If you add the fact that people can’t really make any money out of it because it takes so long to develop, you can see why there’s no innovation. I think that’s actually a big part of it — the economic incentives aren’t there.”

Lex Dromgoole, CEO of Bronze


muscle memory
sacrosanct keyboard shortcuts
roadblocks
exasperated
producers
stability
engineers
dependability
artists
highly customised
sacred space
recreating traditional studios
no innovation
frustrated

“I feel like it’s hit a plateau in terms of creativity and where a lot of people are just creating the same thing. A lot of people are wondering: what is the next thing, what is it going to look like, how can [we] expand on the current capability of what we’re doing with a DAW? But I don’t think anyone has quite nailed what it’ll look like.””

Joshua Hodge, The Audio Programmer

Audio Software Products

DAW

- Recording
- Editing
- MIDI
- Sequencing
- Arrangement
- Playback
- Mixing
- Exporting
- File management

- *Synths*
- *Effects*

- *Pro Tools AudioSuite*
- *Melodyne ARA*
- *Reaper/Presonus extensions*

PLUGINS

- Synths
- Effects
- MIDI generators

Audio Software Products

APPS

- Recording
- Editing
- MIDI
- Sequencing
- Arrangement
- Playback
- Mixing
- Exporting
- File management
- Synths
- Effects
- MIDI generators

“If you add the fact that people can’t really make any money out of it because it takes so long to develop, you can see why there’s no innovation. I think that’s actually a big part of it — **the economic incentives aren’t there**.”

Lex Dromgoole, CEO of Bronze

Why aren’t the economic incentives there?

- Recording
- Editing
- MIDI
- Sequencing
- Arrangement
- Playback
- Mixing
- Exporting
- File management

Costs

D: 0 £0

- Glassdoor: Senior Software Engineer (Audio)
  - *https://www.glassdoor.co.uk/Salaries/senior-audio-software-engineer-salary-SRCH_KO0,30.htm*
- **£72,741 py**
- **~£300 / $375 pd**
- Including tests and validation

Time

Time

D: 2 £600

- Samples
  - *Sample rate*
- Time (s)
  - *Tempo*
- Beats
  - *Time signature*
- Bars + beats

$$Sr = S/T$$

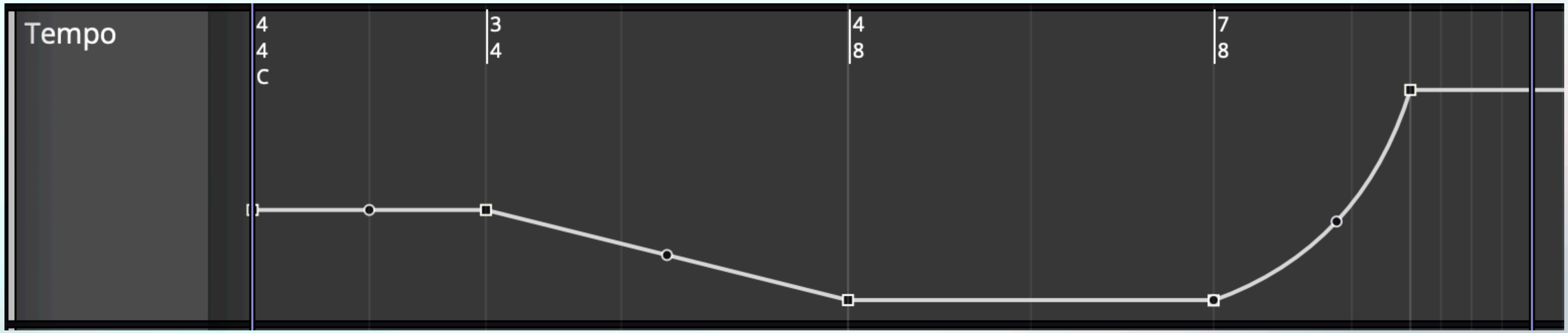
$$bpm = B/(T.60)$$

$$B = bpm/(T.60)$$

$$bars = B/numerator$$

Dynamic Tempo

D: 2 £600



Dynamic Tempo

D: 16 £5k

Sequence Creation

```
inline Sequence::Sequence (std::vector<TempoChange> tempos, std::vector<TimeSigChange> timeSigs, std::vector<KeyChange> keys,
                           LengthOfOneBeat lengthOfOneBeat)
{
    if (keys.empty())
        keys.push_back ({});

    assert (tempos.size() > 0 && timeSigs.size() > 0);
    assert (tempos[0].startBeat == BeatPosition());
    assert (timeSigs[0].startBeat == BeatPosition());
    assert (keys[0].startBeat == BeatPosition());

    // Find the beats with changes
    std::vector<BeatPosition> beatsWithChanges;
    beatsWithChanges.reserve (tempos.size() + timeSigs.size() + keys.size());

    for (const auto& tempo : tempos)
    {
        beatsWithChanges.push_back (tempo.startBeat);

        hash_combine (hashCode, tempo.startBeat.inBeats());
        hash_combine (hashCode, tempo.bpm);
        hash_combine (hashCode, tempo.curve);
    }

    for (const auto& timeSig : timeSigs)
    {
        beatsWithChanges.push_back (timeSig.startBeat);

        hash_combine (hashCode, timeSig.startBeat.inBeats());
        hash_combine (hashCode, timeSig.numerator);
        hash_combine (hashCode, timeSig.denominator);
        hash_combine (hashCode, timeSig.triplets);
    }

    for (const auto& keyChange : keys)
    {
        beatsWithChanges.push_back (keyChange.startBeat);

        hash_combine (hashCode, keyChange.startBeat.inBeats());
        hash_combine (hashCode, keyChange.key.pitch);
        hash_combine (hashCode, keyChange.key.scale);
    }

    std::sort (beatsWithChanges.begin(), beatsWithChanges.end());
    beatsWithChanges.erase (std::unique (beatsWithChanges.begin(), beatsWithChanges.end()),
                            beatsWithChanges.end());

    // Build the sections
    TimePosition time;
    BeatPosition beatNum;
    double ppq              = 0.0;
    size_t timeSigIdx   = 0;
    size_t tempoIdx     = 0;
    size_t keyIdx       = 0;

    auto currTempo    = tempos[tempoIdx++];
    auto currTimeSig  = timeSigs[timeSigIdx++];
    auto currKey      = keys[keyIdx++];

    const bool useDenominator = lengthOfOneBeat == LengthOfOneBeat::dependsOnTimeSignature;

    for (size_t i = 0; i < beatsWithChanges.size(); ++i)
    {
        const auto currentBeat = beatsWithChanges[i];
        assert (std::abs ((currentBeat - beatNum).inBeats()) < 0.001);

        while (tempoIdx < tempos.size() && tempos[tempoIdx].startBeat == currentBeat)
            currTempo = tempos[tempoIdx++];

        if (timeSigIdx < timeSigs.size() && timeSigs[timeSigIdx].startBeat == currentBeat)
            currTimeSig = timeSigs[timeSigIdx++];

        if (keyIdx < keys.size() && keys[keyIdx].startBeat == currentBeat)
            currKey = keys[keyIdx++];

        const bool nextTempoValid = tempoIdx < tempos.size();
        double bpm = nextTempoValid ? calcCurveBpm (currTempo.startBeat.inBeats(), currTempo, tempos[tempoIdx])
                                    : currTempo.bpm;

        int numSubdivisions = 1;

        if (nextTempoValid && (currTempo.curve != -1.0f && currTempo.curve != 1.0f))
            numSubdivisions = static_cast<int> (std::clamp (4.0 * (tempos[tempoIdx].startBeat - currentBeat).inBeats(), 1.0, 100.0));

        const auto numBeats = BeatDuration::fromBeats ((i < beatsWithChanges.size() - 1)
                                                         ? ((beatsWithChanges[i + 1] - currentBeat).inBeats() / (double) numSubdivisions)
                                                         : 1.0e6);

        for (int k = 0; k < numSubdivisions; ++k)
        {
            Section it;

            it.bpm              = bpm;
            it.numerator        = currTimeSig.numerator;
            it.prevNumerator    = it.numerator;
            it.denominator      = currTimeSig.denominator;
            it.triplets         = currTimeSig.triplets;
            it.startTime        = time;
            it.startBeat        = beatNum;

            it.secondsPerBeat   = SecondsPerBeat { useDenominator ? (240.0 / (bpm * it.denominator))
                                                                  : (60.0 / bpm) };

            it.beatsPerSecond   = 1.0 / it.secondsPerBeat;

            it.ppqAtStart = ppq;
            ppq += 4 * numBeats.inBeats() / it.denominator;

            it.key              = currKey.key;

            if (sections.empty())
            {
                it.barNumberOfFirstBar = 0;
                it.beatsUntilFirstBar = {};
                it.timeOfFirstBar = {};
            }
            else
            {
                const auto& prevSection = sections[sections.size() - 1];

                const auto beatsSincePreviousBarUntilStart = (time - prevSection.timeOfFirstBar) * prevSection.beatsPerSecond;
                const auto barsSincePrevBar = (int) std::ceil (beatsSincePreviousBarUntilStart.inBeats() / prevSection.numerator - 1.0e-5);

                it.barNumberOfFirstBar = prevSection.barNumberOfFirstBar + barsSincePrevBar;

                const auto beatNumInEditOfNextBar = BeatPosition::fromBeats ((int) std::lround ((prevSection.startBeat + prevSection.beatsUntilFirstBar).inBeats())
                                                                             + (barsSincePrevBar * prevSection.numerator));

                it.beatsUntilFirstBar = beatNumInEditOfNextBar - it.startBeat;
                it.timeOfFirstBar = time + it.beatsUntilFirstBar * it.secondsPerBeat;

                for (int j = (int) sections.size(); --j >= 0;)
                {
                    auto& tempo = sections[(size_t) j];

                    if (tempo.barNumberOfFirstBar < it.barNumberOfFirstBar)
                    {
                        it.prevNumerator = tempo.numerator;
                        break;
                    }
                }
            }

            sections.push_back (it);

            time = time + numBeats * it.secondsPerBeat;
            beatNum = beatNum + numBeats;

            bpm = nextTempoValid ? calcCurveBpm (beatNum.inBeats(), currTempo, tempos[tempoIdx])
                                 : currTempo.bpm;
        }
    }
}
```

Curve Calculations

```
inline double Sequence::calcCurveBpm (double beat, const TempoChange t1, const TempoChange t2)
{
    const auto b1 = t1.startBeat.inBeats();
    const auto b2 = t2.startBeat.inBeats();
    const auto bpm1 = t1.bpm;
    const auto bpm2 = t2.bpm;
    const auto c = t1.curve;

    const auto [x, y] = getBezierPoint (b1, bpm1, b2, bpm2, c);

    if (c >= -0.5 && c <= 0.5)
        return getBezierYFromX (beat,
                                b1, bpm1, x, y, b2, bpm2);

    double x1end = 0;
    double x2end = 0;
    double y1end = 0;
    double y2end = 0;

    getBezierEnds (b1, bpm1, b2, bpm2, c,
                   x1end, y1end, x2end, y2end);

    if (beat >= b1 && beat <= x1end)
        return y1end;

    if (beat >= x2end && beat <= b2)
        return y2end;

    return getBezierYFromX (beat, x1end, y1end, x, y, x2end, y2end);
}

inline std::pair<double /*x*/, double /*y*/> getBezierPoint (double x1, double y1, double x2, double y2,
                                                             double c) noexcept
{
    if (y2 > y1)
    {
        auto run  = x2 - x1;
        auto rise = y2 - y1;

        auto xc = x1 + run / 2;
        auto yc = y1 + rise / 2;

        auto x = xc - run  / 2 * -c;
        auto y = yc + rise / 2 * -c;

        return { x, y };
    }

    auto run  = x2 - x1;
    auto rise = y1 - y2;

    auto xc = x1 + run / 2;
    auto yc = y2 + rise / 2;

    auto x = xc - run  / 2 * -c;
    auto y = yc - rise / 2 * -c;

    return { x, y };
}

inline void getBezierEnds (const double x1, const double y1, const double x2, const double y2, const double c,
                           double& x1out, double& y1out, double& x2out, double& y2out) noexcept
{
    auto minic = (std::abs (c) - 0.5f) * 2.0f;
    auto run   = minic * (x2 - x1);
    auto rise  = minic * ((y2 > y1) ? (y2 - y1) : (y1 - y2));

    if (c > 0)
    {
        x1out = x1 + run;
        y1out = (float) y1;

        x2out = x2;
        y2out = (float) (y1 < y2 ? (y2 - rise) : (y2 + rise));
    }
    else
    {
        x1out = x1;
        y1out = (float) (y1 < y2 ? (y1 + rise) : (y1 - rise));

        x2out = x2 - run;
        y2out = (float) y2;
    }
}

inline double getBezierYFromX (double x, double x1, double y1, double xb, double yb, double x2, double y2) noexcept
{
    // test for straight lines and bail out
    if (x1 == x2 || y1 == y2)
        return y1;

    // test for endpoints
    if (x <= x1)  return y1;
    if (x >= x2)  return y2;

    // ok, we have a bezier curve with one control point,
    // we know x, we need to find y

    // flip the bezier equation around so its an quadratic equation
    auto a = x1 - 2 * xb + x2;
    auto b = -2 * x1 + 2 * xb;
    auto c = x1 - x;

    // solve for t, [0..1]
    double t;

    if (a == 0)
    {
        t = -c / b;
    }
    else
    {
        t = (-b + std::sqrt (b * b - 4 * a * c)) / (2 * a);

        if (t < 0.0f || t > 1.0f)
            t = (-b - std::sqrt (b * b - 4 * a * c)) / (2 * a);
    }

    jassert (t >= 0.0f && t <= 1.0f);

    // find y using the t we just found
    auto y = (std::pow (1 - t, 2) * y1) + 2 * t * (1 - t) * yb + std::pow (t, 2) * y2;
    return y;
}
```

Beats <-> Time <-> Bars

```
inline BeatPosition toBeats (const std::vector<Sequence::Section>& sections, TimePosition time)
{
    for (int i = (int) sections.size(); --i > 0;)
    {
        auto& it = sections[(size_t) i];

        if (it.startTime <= time)
            return it.startBeat + (time - it.startTime) * it.beatsPerSecond;
    }

    auto& it = sections[0];
    return it.startBeat + ((time - it.startTime) * it.beatsPerSecond);
}

inline TimePosition toTime (const std::vector<Sequence::Section>& sections, BeatPosition beats)
{
    for (int i = (int) sections.size(); --i >= 0;)
    {
        auto& it = sections[(size_t) i];

        if (toPosition (beats - it.startBeat) >= BeatPosition())
            return it.startTime + it.secondsPerBeat * (beats - it.startBeat);
    }

    auto& it = sections[0];
    return it.startTime + it.secondsPerBeat * (beats - it.startBeat);
}

inline TimePosition toTime (const std::vector<Sequence::Section>& sections, BarsAndBeats barsBeats)
{
    for (int i = (int) sections.size(); --i >= 0;)
    {
        const auto& it = sections[(size_t) i];

        if (it.barNumberOfFirstBar == barsBeats.bars + 1
             && barsBeats.beats.inBeats() >= it.prevNumerator - it.beatsUntilFirstBar.inBeats())
            return it.timeOfFirstBar - it.secondsPerBeat * (BeatDuration::fromBeats (it.prevNumerator) - barsBeats.beats);

        if (it.barNumberOfFirstBar <= barsBeats.bars || i == 0)
            return it.timeOfFirstBar + it.secondsPerBeat * (BeatDuration::fromBeats (((barsBeats.bars - it.barNumberOfFirstBar) * it.numerator)) + barsBeats.beats);
    }

    return {};
}

inline BarsAndBeats toBarsAndBeats (const std::vector<Sequence::Section>& sections, TimePosition time)
{
    for (int i = (int) sections.size(); --i >= 0;)
    {
        auto& it = sections[(size_t) i];

        if (it.startTime <= time || i == 0)
        {
            const auto beatsSinceFirstBar = ((time - it.timeOfFirstBar) * it.beatsPerSecond).inBeats();

            if (beatsSinceFirstBar < 0)
                return { it.barNumberOfFirstBar + (int) std::floor (beatsSinceFirstBar / it.numerator),
                         BeatDuration::fromBeats (std::fmod (std::fmod (beatsSinceFirstBar, it.numerator) + it.numerator, it.numerator)),
                         it.numerator };

            return { it.barNumberOfFirstBar + (int) std::floor (beatsSinceFirstBar / it.numerator),
                     BeatDuration::fromBeats (std::fmod (beatsSinceFirstBar, it.numerator)),
                     it.numerator };
        }
    }

    return { 0, {} };
}
```

Audio File Reading

Audio File Reading

Problems:

- Many files
- Long files
- Transport position jumping (click avoidance)
- Clip/region loop points
- Real-time (wait-free)
  - File reads (system calls)
  - Threading

Real-time (wait-free)

- File reads (system calls)
  - Background thread loads files in to memory
  - Audio thread reads from memory
  - Can’t keep all files in memory
    - Large files will be constantly paged to disk
  - Keep track of all the upcoming read positions and load sections of the files in to memory
  - Memory mapping can help this process (not for compressed files)

Real-time (wait-free)

- Audio thread
  - Audio thread has to read this memory
  - Wait-free synchronisation around the memory
    - Fabian Renn-Giles & Dave Rowland - Real-time 101: youtu.be/Q0vrQFyAdWI

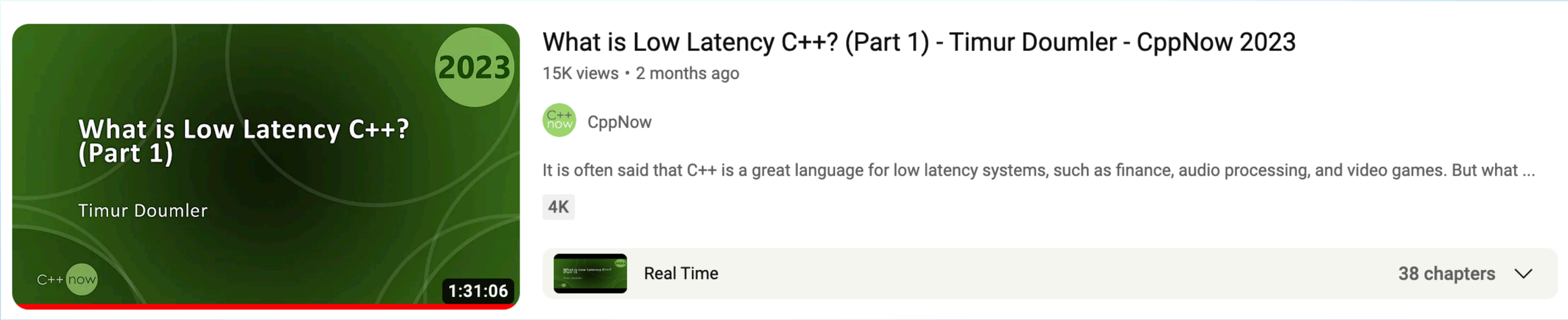
Fabian Renn-Giles & Dave Rowland - Real-time 101 - part I: Investigating the real-time problem space

ADC - Audio Developer Conference • 6.7K views • 3 years ago

Thank you to our VIP patrons: Ahmet Levent Tasel Art and Logic Auxy Elk Audio Felipe Tonello Glenn Kasten Inphonik Jerry Chan Larry Mickie Matt Gilg Overloud Simon Holt Sound Radix Steve Flower...

Real-time (wait-free)

- What is Low Latency C++? Timur Doumler - CppNow 2023
- 180 mins
- youtu.be/EzmNeAhWqVs



Audio File Reading

D: 144 £43k

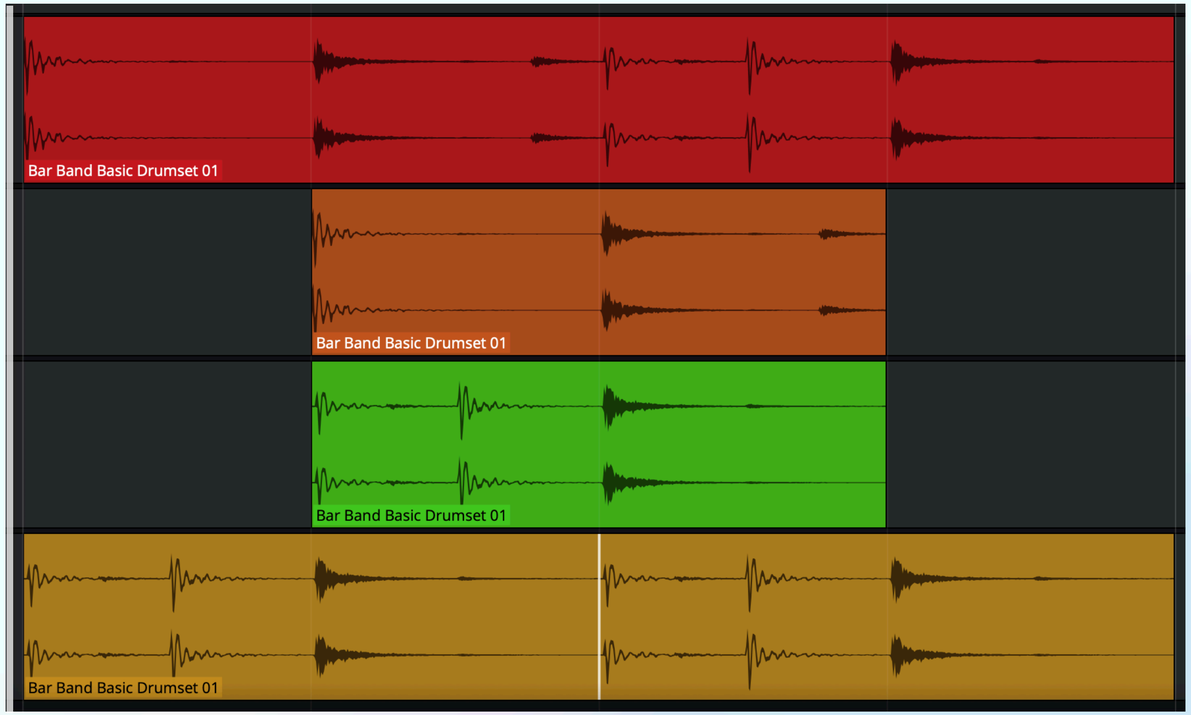
Problems:

- Many files
- Long files
- Transport position jumping (click avoidance)
- Clip/region loop points
- Real-time (wait-free)
  - File reads (system calls)
  - Threading

D: 154 £46k

Clip/region loop points

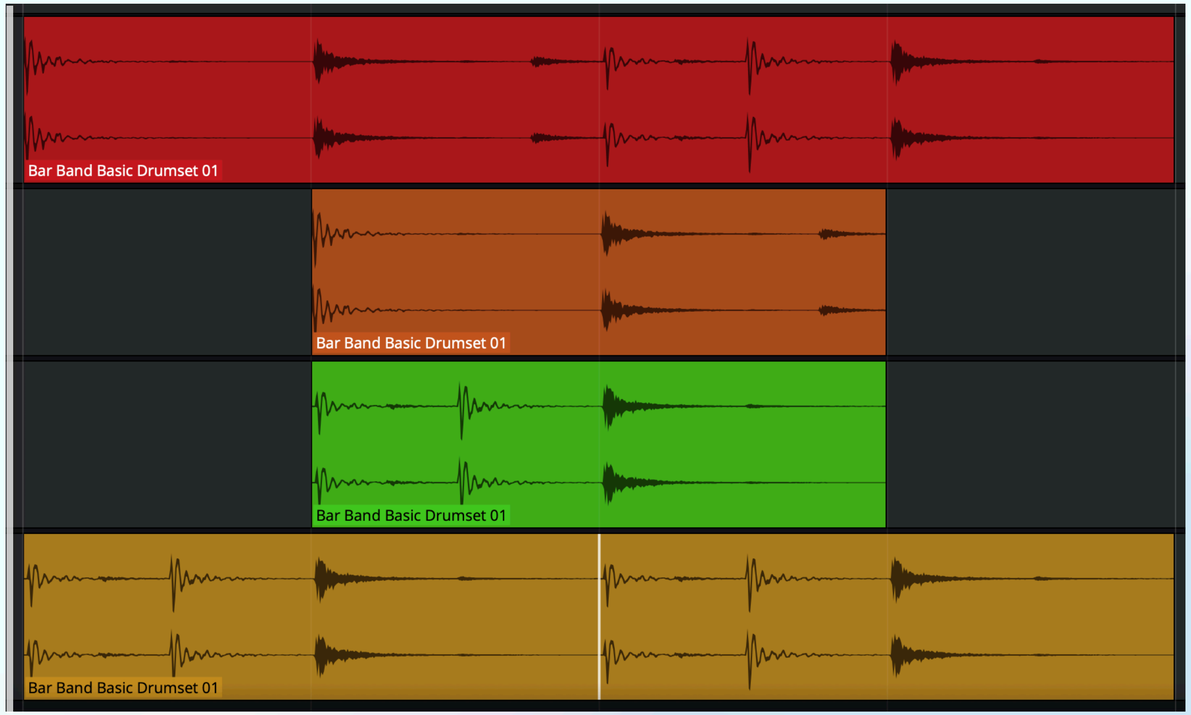
- Clips/regions are not files from start to finish
  - Start/end
  - Offset
  - Loop start/length
- Different read positions
- Multiple read buffers
- Multiple memory touches (memory mapping)



D: 174 £52k

Transport position jumping

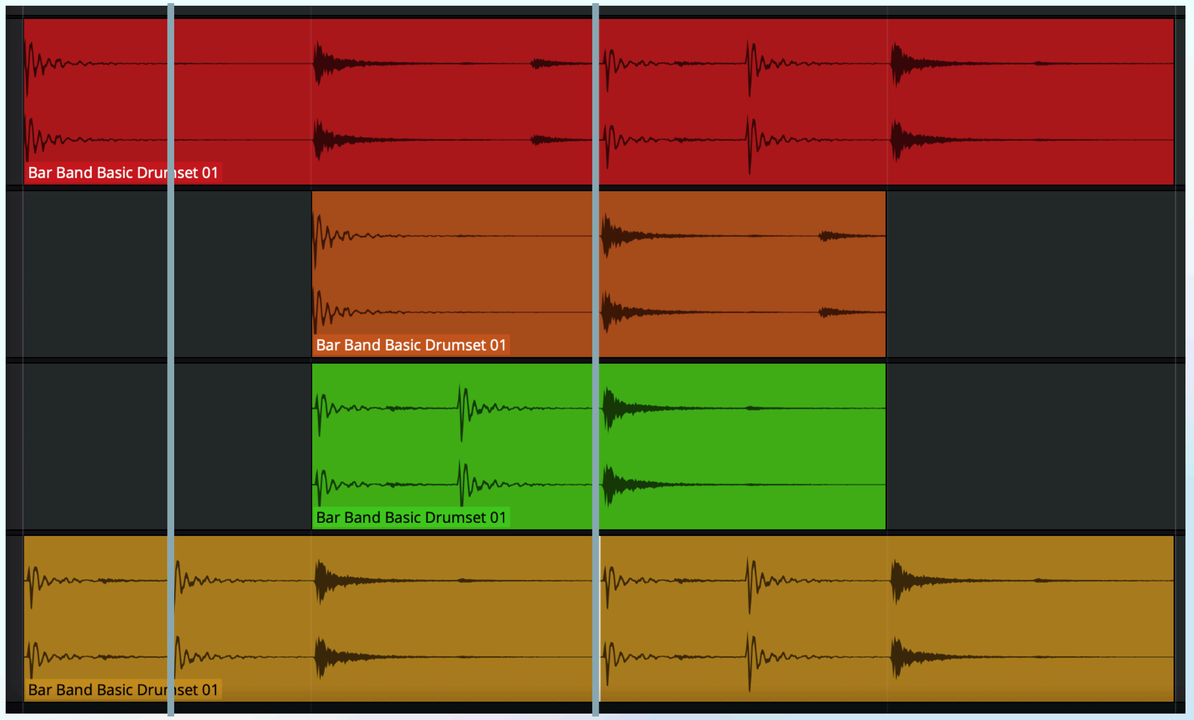
- Playhead jumps (random)
- Read position jumps
- File caching needs to be quick to avoid missing the first samples (transients)
  - Decompress compressed formats
- Fade last samples in to new samples to avoid clicks
  - More difficult than it sounds if the clip/region isn't present before/after the jump



D: 194 £58k

Transport position looping

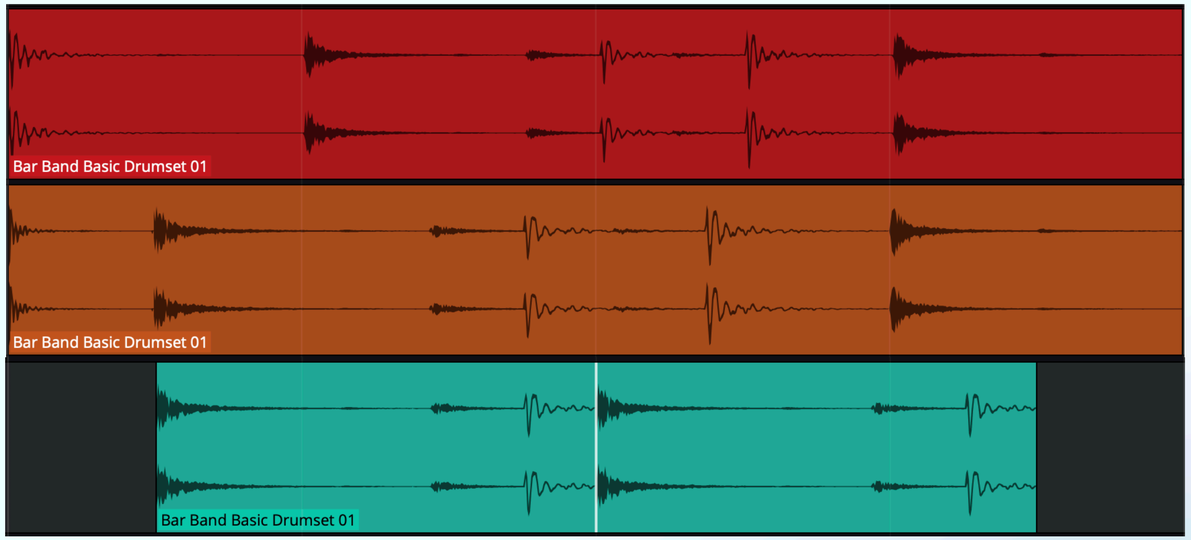
- Transport looping is similar
- You know the jump-to position
- You know when it will happen
- This may be in the middle of a block
- Need to chunk appropriately
- With any jump, tempo/time-sig may have changed



Audio File Reading (Time-stretching)

D: 322 £97k

- Audio file may not be at the same tempo as the transport position
- Clip/region might have a dynamic warp applied to it
- AND be looped
- So time-stretching needs to be applied quickly and without latency to keep everything in sync



Review

D: 322 £97k

- Time/beats (£5k)
- Read audio files (£92k)

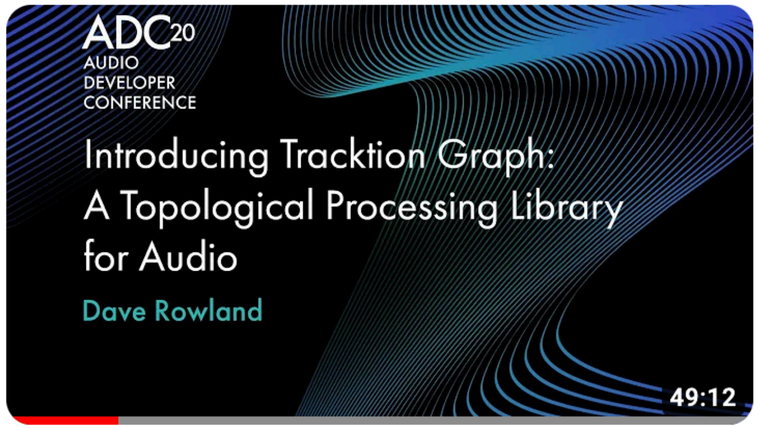
- Playback
- Arrangement
- Recording
- Editing
- MIDI
- Sequencing
- Mixing
- Exporting
- File management

Playback

Playback

D: 687 £206k

- What do we mean?
  - Discussed taking samples and converting them to time/beats
  - Some kind of graph we can build that the audio device can process
- youtu.be/Mkz908eP_4g



Introducing Tracktion Graph: A Topological Processing Library for Audio - Dave Rowland - ADC20

2.3K views • 2 years ago

ADC - Audio Developer Conference

This talk aims to introduce the concept of an audio graph, what the inherent difficulties are and how the **Tracktion Graph** library ...

Introduction | Outline | Topological Graphs | Directed Graphs | Graph Traversal | Graph Cycles |... 17 chapters

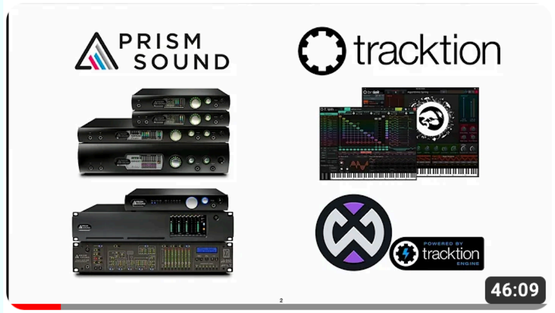
Playback

Problems:

- Flexible
- Scale
- Persist state
- Real-time
- Multiple threads

D: 687 £206k

youtu.be/FpymA7NLNDs



Optimising a Real-Time Audio Processing Library - Dave Rowland - ADC22

917 views • 6 months ago

ADC - Audio Developer Conference

Optimising a Real-time Audio Processing Library - Dave **Rowland** - ADC22 This talk will take you through optimising a codebase ...

Intro | PRISM SOUND | Chapters | What are Benchmarks? | Micro vs Macro Benchmarks? |... 17 chapters

youtu.be/TEeBcjKZhfg

An Engineering Approach to Optimising C++ - Dave Rowland - C++ on Sea 2023

2.8K views • 2 weeks ago

cpponsea

An Engineering Approach to Optimizing C++ - Dave **Rowland** - C++ on Sea 2023 This talk will take you through optimising the ...

Review 2

D: 687 £206k

- Time/beats (£5k)
- Read audio files (£92k)
- Playback graph (£109k)

- Playback
- Arrangement
- Recording
- Editing
- MIDI
- Sequencing
- Mixing
- Exporting
- File management

Arrangement

Arrangement

- Data model
- Specify properties of objects in this model
  - Clip with start/length/source file etc.
- Higher level concept than a playback graph
  - Usually maps fairly well to UI
- Audio graph needs to be built from this
- UI will be built on top of this

Arrangement

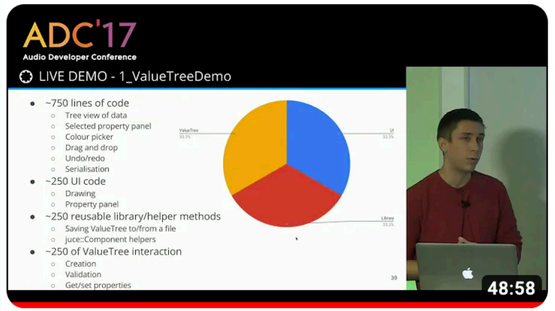
D: 815 £245k

Problems

- Serialisable (file format)
- Scale
- Expandable
- Undo/redo
- Copy/paste
- Sharable (file references)

Using JUCE Value Trees and Modern C++
to Build Large Scale Applications

youtu.be/3IaMjH5lBEY



David Rowland - Using JUCE value trees and modern C++ to build large scale applications (ADC'17)

8.8K views • 5 years ago

ADC - Audio Developer Conference

Using JUCE **value trees** and modern C++ to build large scale applications David **Rowland**, Lead Software Developer, Tracktion ...

Value Tree are Like XML | Serialise Non-primitive Data to Strings | Reduce Boilerplate with... 7 moments

Arrangement

Data model to playback graph

D: 835 £251k

- ~2000 lines of code in Tracktion Engine
- Not always straightforward
  - Aux send/returns are just bus numbers and positions in the model
    - Graph needs to creates nodes with shared buffers, gains, summing nodes which all handle ordering and latency correctly
  - Track mute/solo
    - Two properties control the whether a track should be muted or not
    - Including upstream/downstream tracks from folders/groups
- Lots of places the graph needs to be optimised
  - Clips on a track don't all need to be processed all the time, only those near the playhead
  - Muted tracks/bypassed plugins *may* need to be processed if they introduce latency in order to un-mute/un-bypass correctly

Audio Recording

Audio Recording

- Fundamental problem is writing real-time audio buffers to disk without dropping any samples
  - Real-time thread ending in file writes (system calls)
- Missed blocks in recorded audio are impossible to recover
- Need to deal with audio device latency to ensure created regions/clips align correctly with the timeline
- Looped/take recording
- Punch in/out
- Count-ins
- Recording thumbnails

Audio Recording

D: 963 £289k

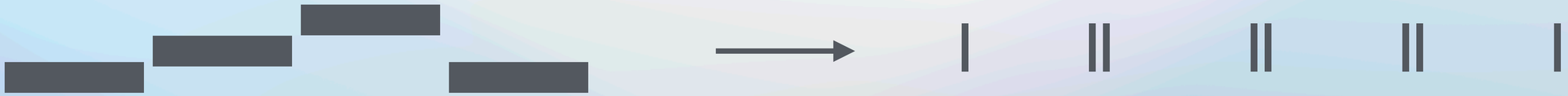
- Need a number of buffers that can be used to transfer audio from the audio thread to the file writing thread
- Stored in lock-free queue
- Multiple inputs reserve a buffer for the current block, write their samples to it and pass it to the thread to write to disk
- Problems:
  - Running out of pre-allocated blocks e.g. disk writes can’t keep up with audio input rate
  - Changing buffer sizes
  - Changing channel numbers
  - Failed file writes
  - Running out of disk space?!

MIDI

MIDI

Playback

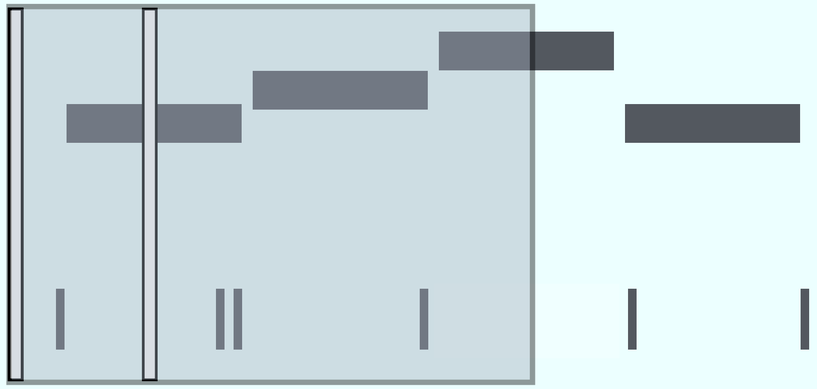
- MIDI is event based (note on/off, CC etc.)
  - Can’t reuse anything from audio 😢
- Less data 😁
- Represented in data model as ***notes***, converted to an event list
  - Notes have start/length, pitch, velocity, channel etc.



MIDI

Problems

D: 1,091 £327k



- Missing notes
  - Starting playback from the middle of a note can be problematic
  - Can happen in lots of situations such as undo/redo, copy/paste/duplicate, transport looping, pitch/groove/quantise/swing changes etc.
- Stuck notes
  - A note is playing, then the playhead jumps ***or*** the note is deleted*
  - How do you know to send a note-off for that playing note
  - Complicated by different MIDI sources e.g. live inputs, multiple clips/tracks, sends, MIDI generators etc.

MIDI

Recording

D: 1,155 £347k

- Simple recording straightforward
  - MIDI thread separate to the audio thread
  - Copy to a pre-allocated buffer
  - Less time-sensitive
- Problems:
  - Quantising incoming notes
  - Playing back the sequence you've recorded (e.g. looping)
  - Converting a linear stream of events in seconds to beats

MIDI

Sequencing

D: 1,283 £385k

- Similar to clip/region arrangement
- Modifying a sequence that is playing back (stuck missing/notes)
- Quantisation
- Grooves/swing
- MPE

Review 3

D: 1,283 £385k

- Time/beats (£5k)
- Read audio/write files (£130k)
- Data model (£45k)
  - State save/load, undo/redo, UI hooks
- Playback graph (£109k)
- Record, sequence and playback MIDI (£96k)

- Playback
- Arrangement
- Recording
- Editing
- MIDI
- Sequencing
- Mixing
- Exporting
- File management

Plugin Hosting

Plugin Hosting

D: 1,347 £404k

Problems

- Initialisation/de-initialisation at the correct time
- Deletion
- Channel busses
- State saving/loading
  - Detecting when they need their states saving
- Automation

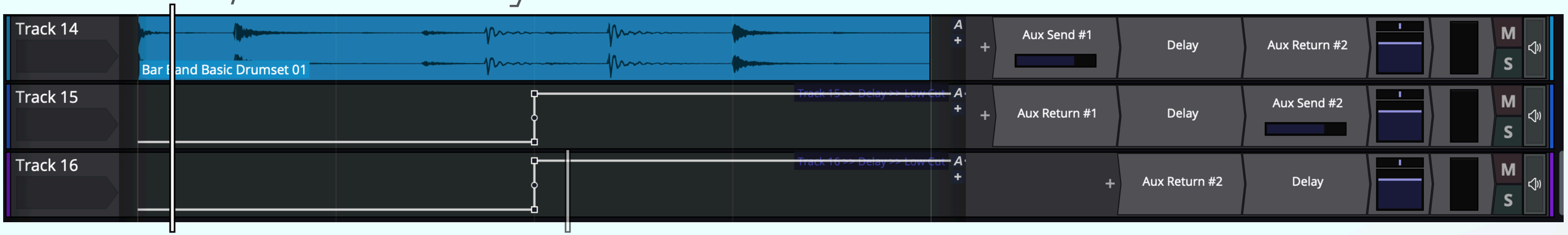
PDC/Latency

PDC/Latency

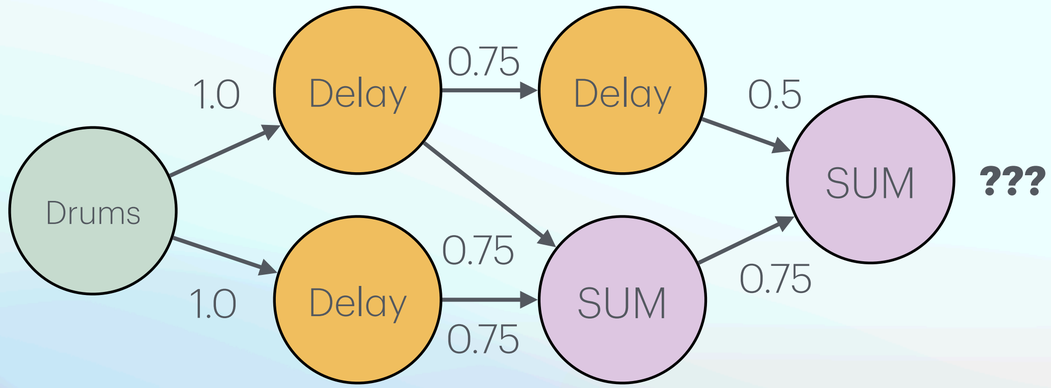
D: 1,411 £423k

- Plugins can introduce latency
- Leads to offsets in the times that subsequent nodes are processing
- Compensate for these delays
- Sync with live input recording
- Ensure plugins read the correct automation values
- Ensure plugins receive the correct timeline timestamps
- Extra complicated via aux busses
- Extra extra complicated when bypassing

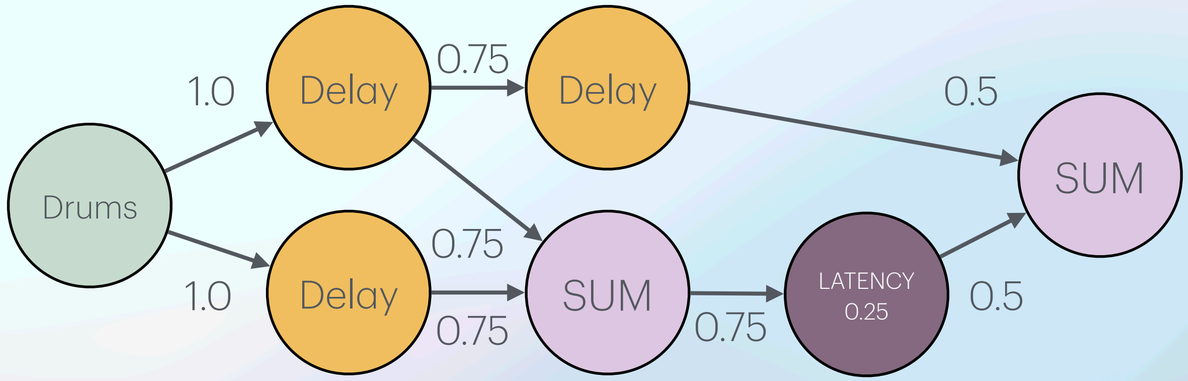
PDC/Latency



T = 1s



T = 1s



Rendering

Rendering

D: 1,475 £443k

- Bouncing down an audio/MIDI file
- Detach from audio device
- Flush plugins
- Reset plugins
- Which tracks to include?
  - Plugin side chain inputs?
  - Group/folder/submix destinations?
  - Aux returns?

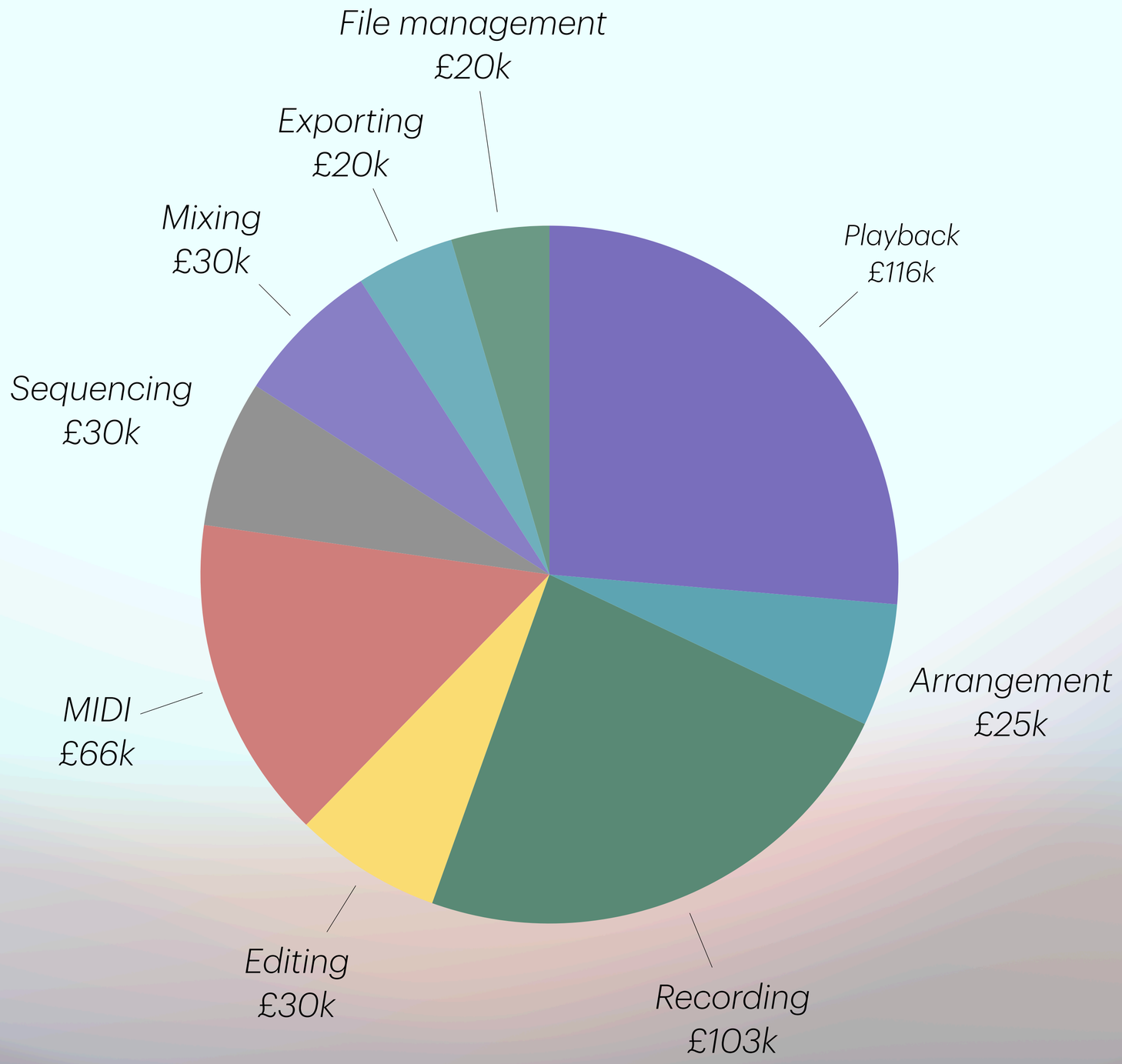
- File format: MIDI, wav, aiff, ogg, flac, mp3
- Sample rate/bit depth
- Normalisation (Peak/RMS)
- Plugin tails
- Silence trimming
- Dithering
- Real-time

Review 4

D: 1,475 £443k

- Time/beats (£5k)
- Read audio/write files (£130k)
- Data model (£45k)
  - State save/load, undo/redo, UI hooks
- Playback graph (£109k)
- Record, sequence and playback MIDI (£96k)
- Host plugins (with PDC) (£38k)
- Export audio/MIDI (£20k)

- Playback **(£116k)**
- Arrangement **(£25k)**
- Recording **(£103k)**
- Editing **(£30k)**
- MIDI **(£66k)**
- Sequencing **(£30k)**
- Mixing **(£30k)**
- Exporting **(£20k)**
- File management **(£20k)**


File management
£20k
Exporting
£20k
Mixing
£30k
Sequencing
£30k
MIDI
£66k
Editing
£30k
Recording
£103k
Arrangement
£25k
Playback
£116k

Missing

D: 4,500 £1.3m

- Product design
- UI development
- Testing/QA
- Website
- Customer support
- AI

AI

AI is a tool, not a feature

- Stem separation
- Vocal/instrument swapping
- EQ/compression/mixing/mastering effects (often need access to multiple tracks)
- Generative AI (MIDI/drums/vocals/music from text)

AI

- Probably need the engine framework for it to work on
  - Access to files/audio streams
  - Rendering
  - Inserting to model

“Artists want to be creative and they want to have control over the creative process.

For TikToks, or podcast music soundtracks, this throw-away generative music might work, but for artists and producers with their own unique identity, it’s hard to see the one-button style solution working.

What’s more likely, is that generative AI replaces steps within the music-making process, rather than replacing creativity entirely.”

Declan McGlynn, DJ Mag

Alternative titles

- “How to crowbar in as many references of my past talks as possible”
- “Behind the scenes of a DAW engine”
- “How to petition your boss for a pay rise”

Why you Shouldn’t Write a DAW (engine)

Another approach?

- Use Tracktion Engine
  - E.g. company with <$2M revenue
  - $150 per seat per month
  - Team of 5
  - = ~$365k in
  - = $9,000 li
  - = **2.5%** of w

Drive & utils: Tracktion Engine
(Cardinal Audio Developers Meetup - May 2019)


Bass
Lead
Synth
FX
Track 6
Track 7
Track 8
Pattern Bass 1 (...
Pattern Bass 1 (...
Pattern Bass 3 (F...
Track Bass 4
Pattern Bass 6 (F...
Lead 1
Lead 2
Lead 3
Lead 4
Lead 5
Synth 1
Synth 1
Synth 1
Synth 1
FX 1
FX 1
FX 1
Scene
Scene
Scene
Scene
Scene
Scene
Scene
Scene
Multi Sampler
Volume & Pan Plugin
Level Meter
Multi Sampler
Volume & Pan Plugin
Level Meter
Multi Sampler
Volume & Pan Plugin
Level Meter
Multi Sampler
Volume & Pan Plugin
Level Meter
Volume & Pan Plugin
Level Meter
Volume & Pan Plugin
Level Meter
Multi Sampler
Volume & Pan Plugin
Level Meter
M
S
4
0
12
24
36
48

Another approach?

- Partner with Tracktion?

innovation

Lex Dromgoole, CEO of Bronze

innovation

Why you shouldn't write a DAW

Build something new

David Rowland

𝕏 @drowaudio

Slides/video:

https://drowaudio.github.io/presentations

tracktion code:

github.com/Tracktion/tracktion_engine