DAVID ROWLAND, JULES STORER - TRACKTION CORPORATION

Tracktion Engine

PART 1 - HISTORY

TRACKTION DAW HISTORY

- Jules released Tracktion 1 DAW in 2002
- Sold to Mackie in 2003
- Spun out the JUCE framework from this in 2004
- From 2008 Mackie stopped releasing updates
- In 2013 the newly formed Tracktion Software Corporation acquired Tracktion and started releasing yearly updates
- In 2017 “Tracktion DAW” was renamed to “Waveform” to disambiguate it from the company name and its other products

TRACKTION ENGINE HISTORY

- 2012 - T3: Code was around 180K LOC
- 2013 - T4: Audio engine refactor
- 2015 - T6: Data model rewrite to use ValueTrees
- 2016 - T7: Separated the engine from the app
- 2018 - W8: Open source the engine, a lot of refactoring
    - Current Engine is about 120K LOC (and app code ~160K) so we've increased the original code by ~65% and quadrupled the functionality
- 2019 - W10: Released Waveform 10, the first version to use the open source version of the Engine

PART 2 - BACKGROUND

WHAT TRACKTION ENGINE IS

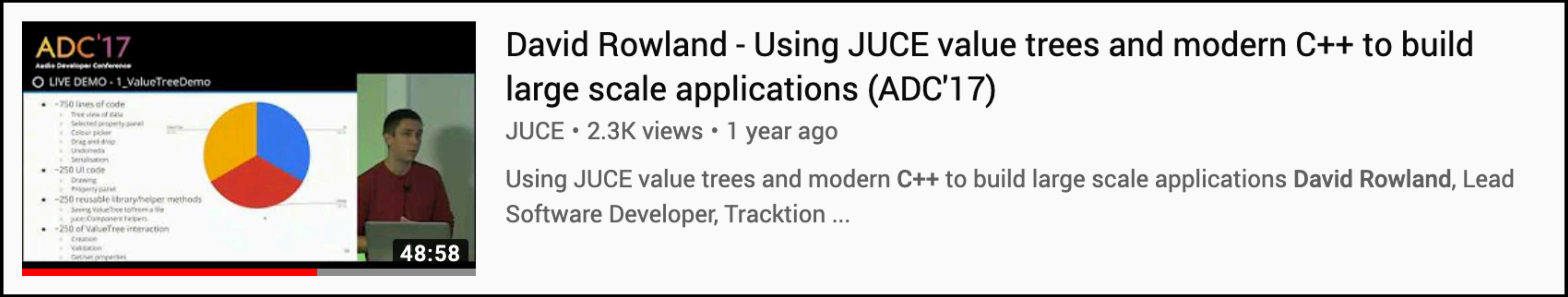
- Framework (JUCE module format)
- Document specification
- Set of classes to manipulate the document
- Audio graph and playback engine
- Utilities and helper methods

WHAT TRACKTION ENGINE ISN'T

- [*Not*] Complete application (excluding demos)
- [*Not*] UI framework
- [*Not*] Part of JUCE

PART 3 - MODEL

PRIMERS



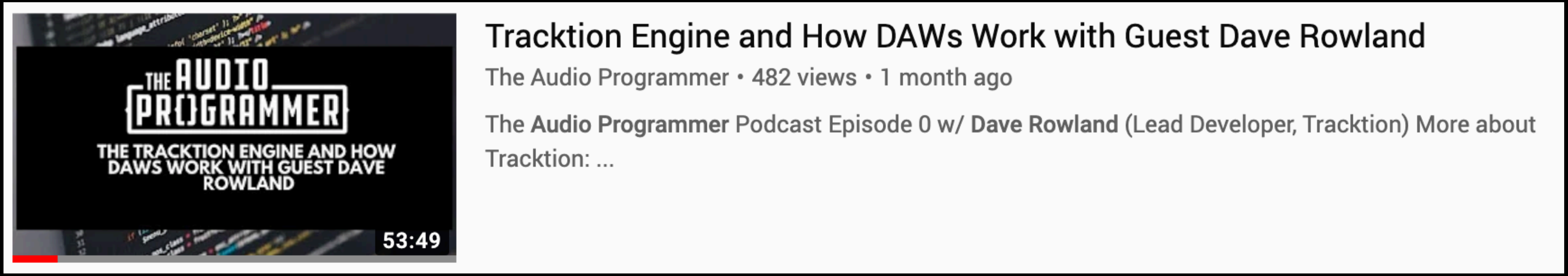
David Rowland - Using JUCE value trees and modern C++ to build large scale applications (ADC'17)

JUCE • 2.3K views • 1 year ago

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

- https://www.youtube.com/watch?v=3IaMjH5lBEY
- Discusses ValueTrees and MVC the way Tracktion Engine handles its model
- Introduces the ValueTreeObjectList
- The live demo is actually a Tracktion Edit file

PRIMERS


THE AUDIO PROGRAMMER
THE TRACKTION ENGINE AND HOW DAWS WORK WITH GUEST DAVE ROWLAND
53:49

Tracktion Engine and How DAWs Work with Guest Dave Rowland

The Audio Programmer • 482 views • 1 month ago

The **Audio Programmer** Podcast Episode 0 w/ **Dave Rowland** (Lead Developer, Tracktion) More about Tracktion: ...

- https://www.youtube.com/watch?v=M-BB-CCqdIc (also Spotify and Apple)
- Discusses:
    - Engine contents and concepts
    - How DAWs work in general

MODEL SPECIFICATION

- ValueTree structure which can be observed or mutated
- Concrete classes which give:
  - Type safety, validation, performance

MODEL SPECIFICATION

- *Project*
    - *Audio/MIDI files etc.*
    - Edit
        - Tracks (audio/MIDI, folder, marker, tempo, chord)
            - Clips (audio, MIDI, step, marker etc.)
                - MIDI lists, step sequences, takes, comps etc.
            - Plugins (vol/pan, aux send/return, external plugins etc.)
                - Automation
            - Modifiers (LFOs, breakpoint envelope, MIDI mapper etc.)
            - Macros parameters

CLIP EXAMPLE

- XML structure
- Clip class

```
<MIDICLIP id="1232" name="Brake" start="58.615326" length="0.461538"
              colour="7090ee91" volDb="0">
    <SEQUENCE name="Combinator 10" channelNumber="1" ver="1">
        <NOTE p="50" v="96" b="0.0" l="0.0625"/>
        <NOTE p="50" v="96" b="0.125" l="0.125"/>
    </SEQUENCE>
    <QUANTISATION type="(none)" amount="1"/>
    <GROOVE current=""/>
</MIDICLIP>
```

```
class MidiClip  : public Clip
{
public:
    //==============================================================================
    MidiClip (const juce::ValueTree&, EditItemID, ClipTrack&);
    ~MidiClip();

    AudioTrack* getAudioTrack() const;

    //==============================================================================
    AudioNode* createAudioNode (const CreateAudioNodeParams&) override;

    MidiList& getSequence() const noexcept;
    MidiList& getSequenceLooped();

    MidiChannel getMidiChannel() const;
    void setMidiChannel (MidiChannel newChannel);

    //==============================================================================
    QuantisationType& getQuantisation() const noexcept;
    void setQuantisation (const QuantisationType& newType);

    juce::String getGrooveTemplate() const noexcept;
    void setGrooveTemplate (const juce::String& templateName);
```

EXTENDING THE ENGINE

- Extend your app by adding your own ValueTree properties
- Extend by registering built-in plugins
- Customisable behaviours

PART 4 - AUDIO PIPELINE

AUDIO PIPELINE

- Edit
  - EditPlaybackContext
    - OutputDevice
      - AudioNode*

AUDIO PIPELINE

```
class AudioNode
{
public:
    //==============================================================================
    AudioNode();
    virtual ~AudioNode();

    //==============================================================================
   /** tells the node to initialise itself ready for playing from the given time.
        This call may be made more than once before releaseAudioNodeResources() is called
    */
    virtual void prepareAudioNodeToPlay (const PlaybackInitialisationInfo&) = 0;

    /** tells the node that play has stopped, and it can free up anything it no longer needs. */
    virtual void releaseAudioNodeResources() = 0;

    //==============================================================================
    // called before renderOver/Adding, to allow prefetching, etc
    virtual void prepareForNextBlock (const AudioRenderContext&)   {}
    virtual void renderOver (const AudioRenderContext&) = 0;
```

AUDIO PIPELINE

```
struct AudioRenderContext
{
    //==============================================================================
    inline AudioRenderContext (PlayHead&, EditTimeRange,
                               juce::AudioBuffer<float>*,
                               const juce::AudioChannelSet&,
                               int bufferStart, int bufferSize,
                               MidiMessageArray*, double midiOffset,
                               int continuityFlags, bool rendering) noexcept;

    //==============================================================================
    /** The playhead provides information about current time, tempo etc at the block
        being rendered.
    */
    PlayHead& playhead;

    /** The time window which needs to be rendered into the current block.
        This is a monotonically increasing window, even if playback is paused. To find
        out what section of the edit needs to be rendered, Playhead provides conversion
        methods such as Playhead::streamTimeToEditWindow() or getEditTime()
    */
    EditTimeRange streamTime;

    /** The target audio buffer which needs to be filled.
        This may be nullptr if no audio is being processed.
    */
    juce::AudioBuffer<float>* destBuffer;

    /** A description of the type of channels in each of the channels in destBuffer. */
    juce::AudioChannelSet destBufferChannels;

    /** A buffer of MIDI events to process.
        This may be nullptr if no MIDI is being sent
    */
    MidiMessageArray* bufferForMidiMessages;

    //==============================================================================
    bool isContiguousWithPreviousBlock() const noexcept;
    bool isFirstBlockOfLoop() const noexcept;
    bool isLastBlockOfLoop() const noexcept;
    bool didPlayheadJump() const noexcept;

    //==============================================================================
    /** Returns the section of the edit that needs to be rendered by this block. */
    PlayHead::EditTimeWindow getEditTime() const;
```

UTILITIES

- Audio thumbnails
- Scratch buffers
- Lock-free audio queues
- Dithering
- Band-limited oscillators
- Crash tracing
- ValueTree utilities
- Automation curves
- Musicality

GETTING STARTED

- Clone the repo and recurse the submodules:

```
$ git clone --recurse-submodules https://github.com/Tracktion/tracktion_engine.git
```

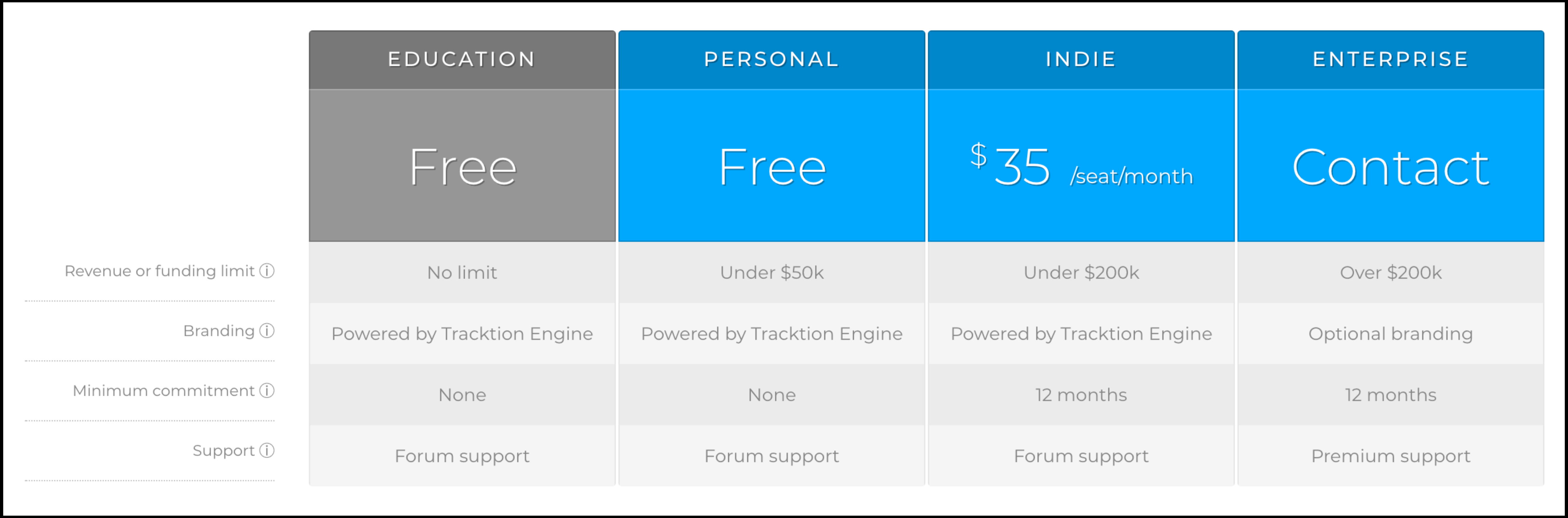
- Example projects are located in `/examples`

```
$ cd tests/mac
$ ./generate_examples
```

- Start with `PitchAndTimeDemo` or `StepSequencerDemo`

PART 5 - DEMOS

LICENSING



	EDUCATION	PERSONAL	INDIE	ENTERPRISE
	Free	Free	$35 /seat/month	Contact
Revenue or funding limit ⓘ	No limit	Under $50k	Under $200k	Over $200k
Branding ⓘ	Powered by Tracktion Engine	Powered by Tracktion Engine	Powered by Tracktion Engine	Optional branding
Minimum commitment ⓘ	None	None	12 months	12 months
Support ⓘ	Forum support	Forum support	Forum support	Premium support

LINKS

- Presentation available on GitHub:
  https://github.com/drowaudio/presentations
- Tracktion Engine GitHub:
  https://www.github.com/Tracktion/tracktion_engine
- Tracktion Engine Website:
  https://www.tracktion.com/develop/tracktion-engine
- Twitter:
  @drowaudio

BONUS SLIDES

AUDIO PIPELINE

- AudioNode (played by an OutputDevice)
    - -> PlayheadAudioNode
      -> MixerAudioNode
      -> TrackMutingAudioNode
      -> PluginAudioNode -> PluginAudioNode
      -> CombiningAudioNode
        -> WaveAudioNode, MIDIAudioNode