DAVID ROWLAND - TRACKTION CORPORATION

pluginval

YEAH, BUT WHY VALIDATE PLUGINS?

TRACKTION DEVELOPERS PORTAL

- tracktion.com/develop
- "Tracktion Engine" now open source
  - Build anything from small app to full DAWs
  - github.com/tracktion/tracktion_engine
- "Verified by pluginval" scheme
  - Promoting best practices
- "Tracktion Presents" distribution services
  - We take care of web, marketing and licensing
  - You focus on code and features

NOTES

- Slides are numbered
- Slides/examples will be on GitHub

CONTENTS

- Landscape of developing plugins
- Common problems between hosts and plugins
- Improving confidence in releases using testing
- Introducing `pluginval` (motivation, history, architecture and use)
- Towards standards compliance…

PART 1 - DEVELOPING PLUGINS

LANDSCAPE OF PLUGIN DEVELOPMENT

- 3 OSes (Windows, macOS, Linux)
- 4 Plugin formats (VST3, VST, AU, AAX)
- 2 architectures 32/64-bit
- 12 major hosts [Wikipedia]
  - plus 12 smaller hosts
- iOS/AUv3?

“288 Environments to Test”

EARLY PLUGIN DEVELOPMENT

- Started with VST
- API of ~23 calls
- Encompassed
  - State of plugin (sample rate/block size)
  - Parameters
  - Programs
  - Your DSP
- Optional advanced features (busses, double precision etc.)

SIMPLE PLUGIN DEV WITH JUCE

- Cross-platform implementation of VST spec
- Implement `AudioProcessor`
- ~40 calls

TODAY

- AudioProcessor
- AudioProcessorParameter
- Busses (AudioChannelSet)
- AudioProcessorParameterGroup
- AudioPlayhead/ CurrentPositionInfo
- TrackProperties, CurveData etc.

FEATURES

- Parameter values, names, labels, text readouts, defaults, suffixes, steps, categories
- Processing, single/double precision, bypassed processing, suspension, real-time/rendering
- Playhead information (tempo, key, position, playing, recording etc.)
- Channel layout, bus enablement
- Latency, state restoration, program lists
- MPE compatibility
- Listeners, extra functions such as automation state, editors, scale factors, resizing, Hi-DPI, parameter control highlight info, host MIDI controller presence…

PER RELEASE

- ~100 features/calls
- ~28,800 things to test
  - “Complexity factor”

LET'S TALK ABOUT THREADS

THREADS

- Some plugin APIs don’t mention threads
- If they do, they don’t all agree on a threading model
- Assume the worst
- At least two threads (message, audio), sometimes more
- Complexity factor
  - x2 = 57,600
  - x10 = 288,000

“But it works in…”

–EVERYONE

SIMPLE THINGS

- Hosts not calling all methods
- Plugins not testing all methods
- E.g. `getParameterText` on ***all*** parameters to send to an external control surface

HOSTS DOING *"UNEXPECTED"* THINGS

- Especially if you're new to the world of plugin development
- `prepareToPlay` being called with block size `X`
- `processBlock` being called with `<X`
- Common for fine-grain automation or modulation
- Reduces parameter aliasing

PLUGINS DOING *"UNEXPECTED"* THINGS

- Block message thread
- Open modal registration/ authorisation windows
- Annoying
- Can deadlock

PLUGINS WRITING FILES ON INSTANTIATION

- Plugins installed to “C:\Program Files (x86)\Steinberg\VstPlugins”
- Try to write some data files needed by the plugin to their install location
- Fails on modern OSes
  - For extremely good reason
- “*Run as administrator*”
  - Just opened a huge security hole. Any compromised host or plugin now has complete control over your system

"DISAGREEMENT" IN THREADING

- Widest source of errors
- Difficult to track down
  - Can cause heap-corruptions that only manifest much later
  - Non-obvious in log files

"DISAGREEMENT" IN THREADING

- Plugins assume:
  - Calls will only be made from certain threads
  - Won't be made concurrently with other calls
  - E.g. host calling `processBlock` whilst a rotary in host is moved calling `setParameter`
- Hosts assume:
  - Methods called on timers
  - Delegates to the message thread
  - E.g. `getPlayHead` which should only be called from `processBlock`

REASONS

- Not always due to poor coding
- APIs across formats differ
- Difficult and time consuming to test every possible combination
- Remember **288,000**
- Plugin/host development historically been *anecdotal* and based on *precedent*
- Not a good ideal for conformity

WHAT'S THE SOLUTION?

- Assume the worst
- Test every possibility
- Use all the tools

- Every change or release, you have changed the system
- Need to re-test everything
- Aiming for confidence in release

HOW?

DON'T *ONLY* TEST WITH HUMANS

- No longer viable due to sheer size of landscape
- Humans catch different things to tools

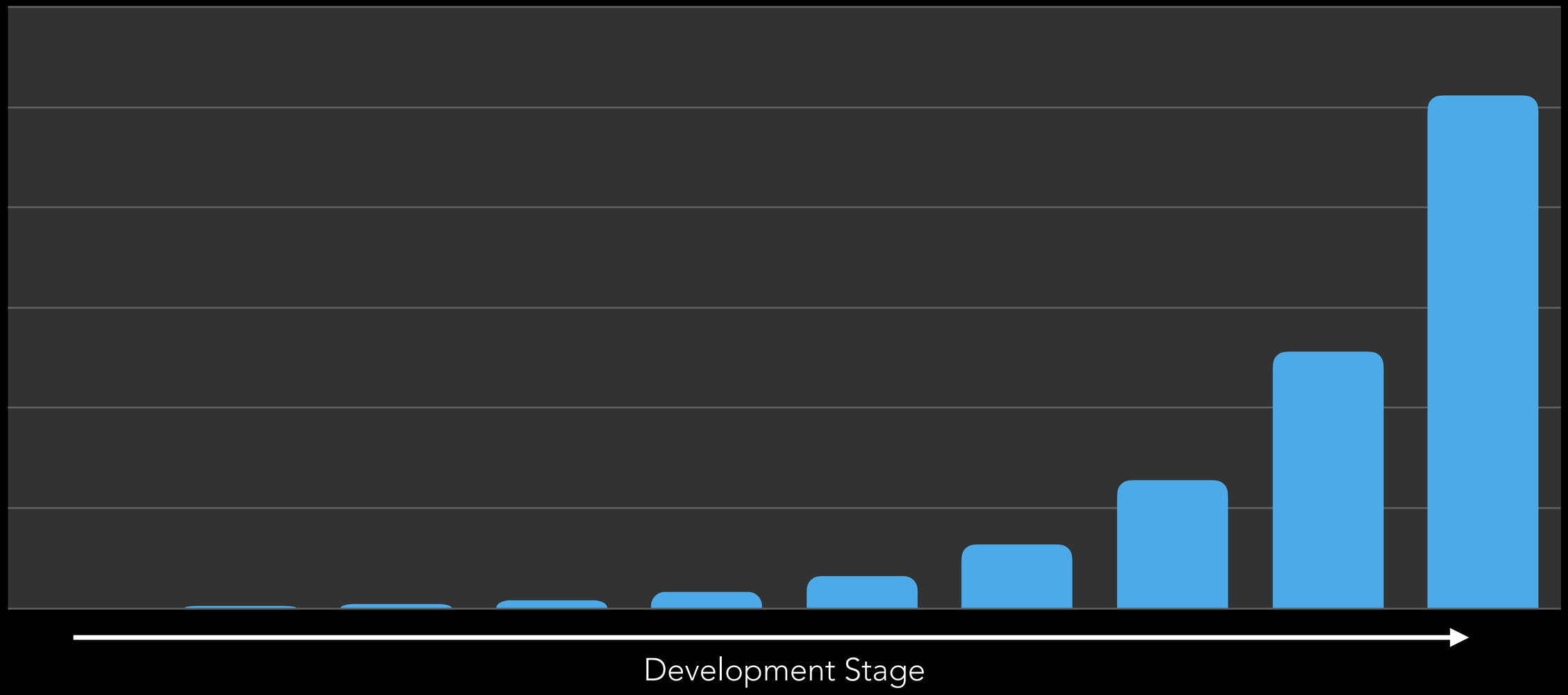
COST ASSOCIATED WITH EACH STAGE

- Time to identify
- Time to debug/fix
- Time to test
- Time to release

CATCH ISSUES EARLY

- As you type e.g. live issues
- Compiler warnings/errors (use -Werror to reject from remotes)
- Static analysers
- Run-time tests e.g. Asan, TSan, UBSan, Valgrind
- Unit tests (run these during initialisation)
- Automated tests e.g. loading projects/state from command line
- Validation tools `auval`, `AAXVal`, `vstscanner, pluginval`
- Fuzzers (can be part of validation tools)
- QA - targeted testing
- Private beta testers - usually good at testing their own workflows (making bug reporting easy is essential)
- Public beta testers - as above but less controlled
- Release (always revealing)
- Sale weekend - much more uptick and stresses new things such as download bandwidth
- New product release - similar to above but with little or no experience

COST OVER TIME



WHAT CAN WE DO?

- Improve the things we have control over
- E.g. tools and automated testing

PART 2 - pluginval

MOTIVATION

- Looking for an `auval`-like tool that could be used cross platform
- Can be easily made part of automated workflows and CI
- Open source to enable:
  - Community contributions
  - Teaching about the way hosts "see" and interact with plugins

HISTORY

- Started life as adding command-line options to Waveform to open sessions, run tests etc.
- Realised this was a niche, heavyweight solution that didn't isolate plugins
- Started building a standalone tool to replicate `auval` for all JUCE-supported formats
- Expanded beyond the tests `auval` offers
- Added UI so it can be used by QA, beta testers and end users

pluginval v0.1.4

Plugin List | Console

Name	Format	Category	Manufacturer	Description
#TAuto Filter	VST	Effect	Tracktion	0.1.2.5
#TBus Compressor	VST	Effect	Tracktion	0.1.2.5
#TChorus	VST	Effect	Tracktion	0.1.2.5
#TCompressor	VST	Effect	Tracktion	0.1.2.5
#TCrossover	VST	Effect	Tracktion	0.1.2.5
#TCrusher	VST	Effect	Tracktion	0.1.2.5
#TDelay Studio	VST	Effect	Tracktion	0.1.2.5
#TDucker	VST	Effect	Tracktion	0.1.2.5
#TEqualiser	VST	Effect	Tracktion	0.1.2.5
#TFlanger	VST	Effect	Tracktion	0.1.2.5
#TGate Expander	VST	Effect	Tracktion	#TGate/Expander – 0.1.2.5
#TLimiter	VST	Effect	Tracktion	0.1.2.5
#TPhaser	VST	Effect	Tracktion	0.1.2.5
#TReverber8	VST	Reverb	Tracktion	0.1.2.5
#TStereo Delay	VST	Effect	Tracktion	0.1.2.5
#TTape Delay	VST	Effect	Tracktion	0.1.2.5
AllocatingPluginDemo	AudioUnit	Effect	JUCE	1.0.0
AllocatingPluginDemo	VST3	Fx	JUCE	1.0.0
ArpeggiatorPlugin	VST	Effect	JUCE	1.0.0.0
ArpeggiatorPlugin	AudioUnit	MidiEffects	JUCE	1.0.0
BioTek	VST	Synth	Tracktion	1.6.4.0
BioTek 2	VST	Synth	Tracktion	
DSPModulePluginDemo	VST	Effect	JUCE	
FF Twin 2	AudioUnit	Synth	FabFilter	

Validate in process
Set timeout (30000ms)
Verbose logging
Show settings folder

Options... | Test Selected | Test All | Strictness Level | 10 | - | + | Options | Clear log | Save log

pluginval v0.1.4

Plugin List | Console

```
pluginval v0.1.4 - JUCE v5.3.2
Started validating: VST-#TAuto Filter-fa112ea5-4175746f
Random seed: 0x467dc54
Validation started: 5 Nov 2018 8:26:39pm


Testing plugin: VST-#TAuto Filter-fa112ea5-4175746f
------------------------------------------------------------
Starting test: pluginval / Open plugin (cold)...

Time taken to open plugin (cold): 202 ms
All tests completed successfully
------------------------------------------------------------
Starting test: pluginval / Open plugin (warm)...

Time taken to open plugin (warm): 8 ms
All tests completed successfully
------------------------------------------------------------
Starting test: pluginval / Plugin info...

Plugin name: #TAuto Filter
Alternative names: #TAuto Filter
SupportsDoublePrecision: no
Reported latency: 0
Reported taillength: 0

Time taken to run test: 0
All tests completed successfully
------------------------------------------------------------
Starting test: pluginval / Editor...

Time taken to open editor (cold): 52 ms
Time taken to open editor (warm): 4 ms

Time taken to run test: 57 ms
All tests completed successfully
------------------------------------------------------------
Starting test: pluginval / Audio processing...
Testing with sample rate [44100] and block size [64]
Testing with sample rate [44100] and block size [128]
Testing with sample rate [44100] and block size [256]
Testing with sample rate [44100] and block size [512]
Testing with sample rate [44100] and block size [1024]
Testing with sample rate [48000] and block size [64]
```

Test Selected | Test All | Strictness Level | 10 | - | + | Options | Clear log | Save log

WHAT DOES IT TEST?

- Plugin info
- Parameters
- Editor
- Plugin state saving and restoration
- Audio processing - multiple block sizes and sample rates
- Audio processing - sub blocks
- Automation
- Multi-threaded state & value setting
- Bus/channel layout info
- Memory allocations on the audio thread (`operator new`)
- Parameter fuzzing

"Sorry, we don't support…"

–ANY DEVELOPER

USE CASES - FOR PLUGIN DEVELOPERS

- Quick, easy to use validation
- Can be part of CI
- Improves confidence when making releases

USE CASES - FOR HOST DEVELOPERS

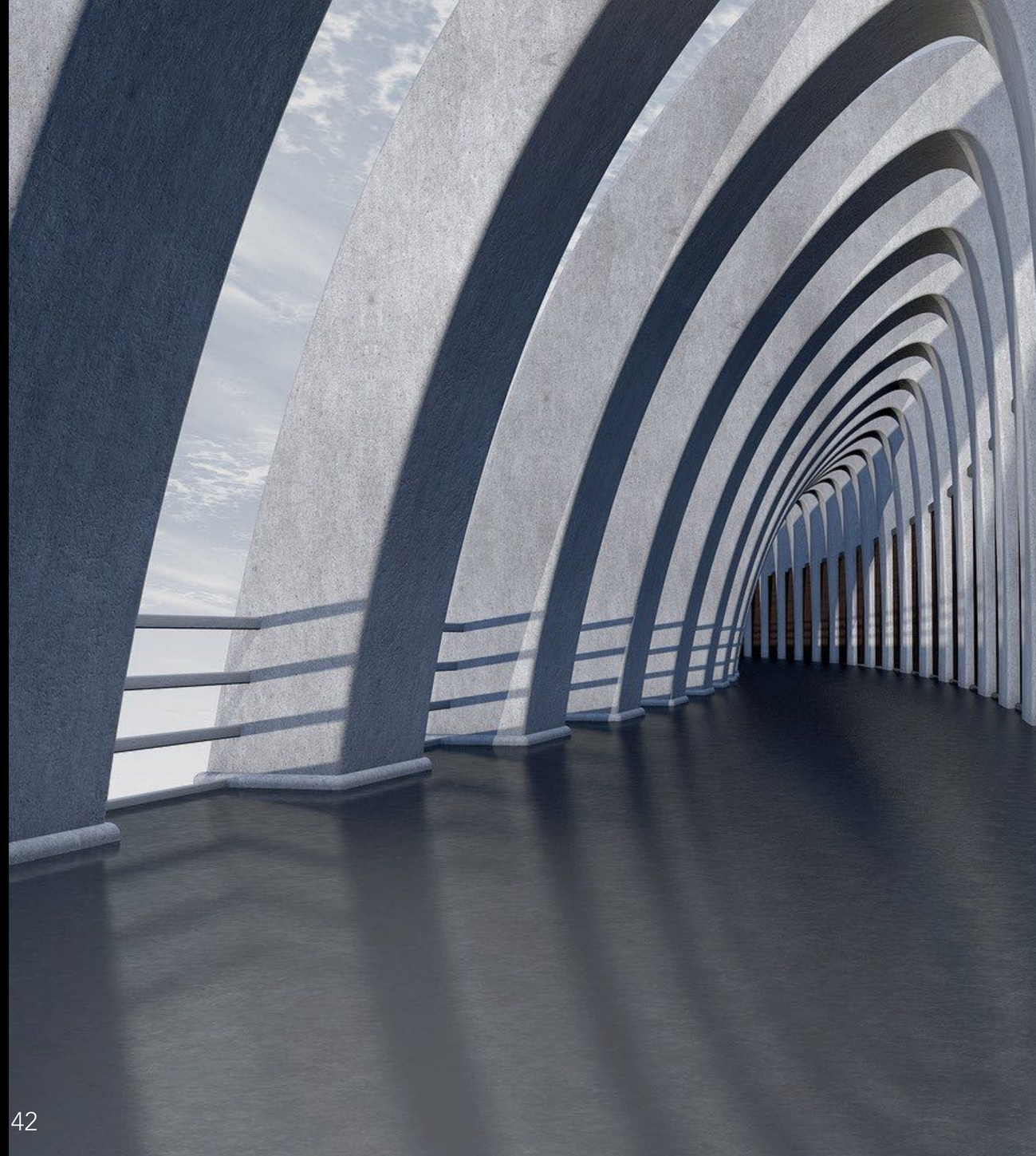
- Quick way to test compatibility with your JUCE based host
- If plugins fail validation, why?
  - Are there any quick workarounds or obvious bugs in the hosting code?
- Much quicker way of getting QA or users to test plugins
  - Reduces the likelihood having to go through the process of testing the plugin personally
  - Logs can be shared which can pinpoint issues
- Easier to share `pluginval` logs or tests with plugin developers
  - Open source improves transparency

USE CASES - FOR END USERS

- Makes reporting issues quicker
- Much simpler to look at a `pluginval` log file
- In general:
  - The more succinct the information you can provide…
  - The easier and quicker it will be to fix

ARCHITECTURE

- UI and command line app
- In and out of process validation
- Validate multiple plugins
- Based on JUCE `UnitTest` framework
- Easy to contribute tests



EXAMPLE TEST

```
//==============================================================================
struct PluginStateTest  : public PluginTest
{
    PluginStateTest()
        : PluginTest ("Plugin state", 2)
    {
    }

    void runTest (PluginTests& ut, AudioPluginInstance& instance) override
    {
        // Read state
        MemoryBlock originalState;
        instance.getStateInformation (originalState);

        // Set random parameter values
        for (auto parameter : getNonBypassAutomatableParameters (instance))
            parameter->setValue (ut.getRandom().nextFloat());

        // Restore original state
        instance.setStateInformation (originalState.getData(), (int) originalState.getSize());
    }
};

static PluginStateTest pluginStateTest;
```

DELIVERY/USE

- As source code (master/develop branches)
  - One line to clone from the terminal
  - Self contained (includes JUCE as a Git submodule)
  - Scripts to build and test the app (`/tests`)
- As versioned binaries
- As latest binaries

- Clone and build

```
$ git clone --recurse-submodule git@github.com:Tracktion/pluginval.git
$ cd pluginval/install
$ ./mac_build
```

- Or download latest

```
$ curl -L "https://github.com/Tracktion/pluginval/releases/download/latest_release/
pluginval_macOS.zip" -o pluginval.zip
$ unzip pluginval
```

- Validate plugin

```
$ cd ../bin/mac/pluginval.app/Contents/MacOS/
$ ./pluginval --strictness-level 10 --validate "/Library/Audio/Plug-Ins/VST3/MyPlugin.vst3"
```

```
pluginval v0.1.4 - JUCE v5.3.2
Started validating: VST-#TAuto Filter-fa112ea5-4175746f
Random seed: 0x318dc95
Validation started: 8 Nov 2018 12:28:36pm


Testing plugin: VST-#TAuto Filter-fa112ea5-4175746f
----------------------------------------------------------------
Starting test: pluginval / Open plugin (cold)...

Time taken to open plugin (cold): 193 ms
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Open plugin (warm)...

Time taken to open plugin (warm): 8 ms
All tests completed successfully

----------------------------------------------------------------
Starting test: pluginval / Background thread state...

Time taken to run test: 2 secs
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Basic bus...
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Listing available buses...
Inputs:
   Named layouts: Mono, Stereo
   Discrete layouts: Discrete #1, Discrete #2
Outputs:
   Named layouts: Mono, Stereo
   Discrete layouts: Discrete #1, Discrete #2
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Restoring default layout...

Time taken to run test: 0
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Editor stress...
Testing opening Editor with released processing
Testing opening Editor with 0 sample rate and block size

Time taken to run test: 9 ms
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Allocations during process...
Testing with sample rate [44100] and block size [64]
…
Testing with sample rate [96000] and block size [1024]

Time taken to run test: 10 ms
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Process called with a larger than prepared block size...
Preparing with sample rate [44100] and block size [64], processing with block size [128]
Preparing with sample rate [96000] and block size [1024], processing with block size [2048]

Time taken to run test: 21 ms
All tests completed successfully
----------------------------------------------------------------
Starting test: pluginval / Fuzz parameters...

Time taken to run test: 1 ms
All tests completed successfully

Finished validating: VST-#TAuto Filter-fa112ea5-4175746f
ALL TESTS PASSED
```

FUTURE DEVELOPMENT

- JUnit output formatting
- Javascript bindings - write tests in Javascript
- More bespoke testing
  - Provide with known input MIDI/audio and compare to known output
  - Custom Javascript tests with custom data files should make this a breeze

PLUGINVAL USERS

Delta-V Audio

tritik

tracktion

DIRECT APPROACH

GG AUDIO

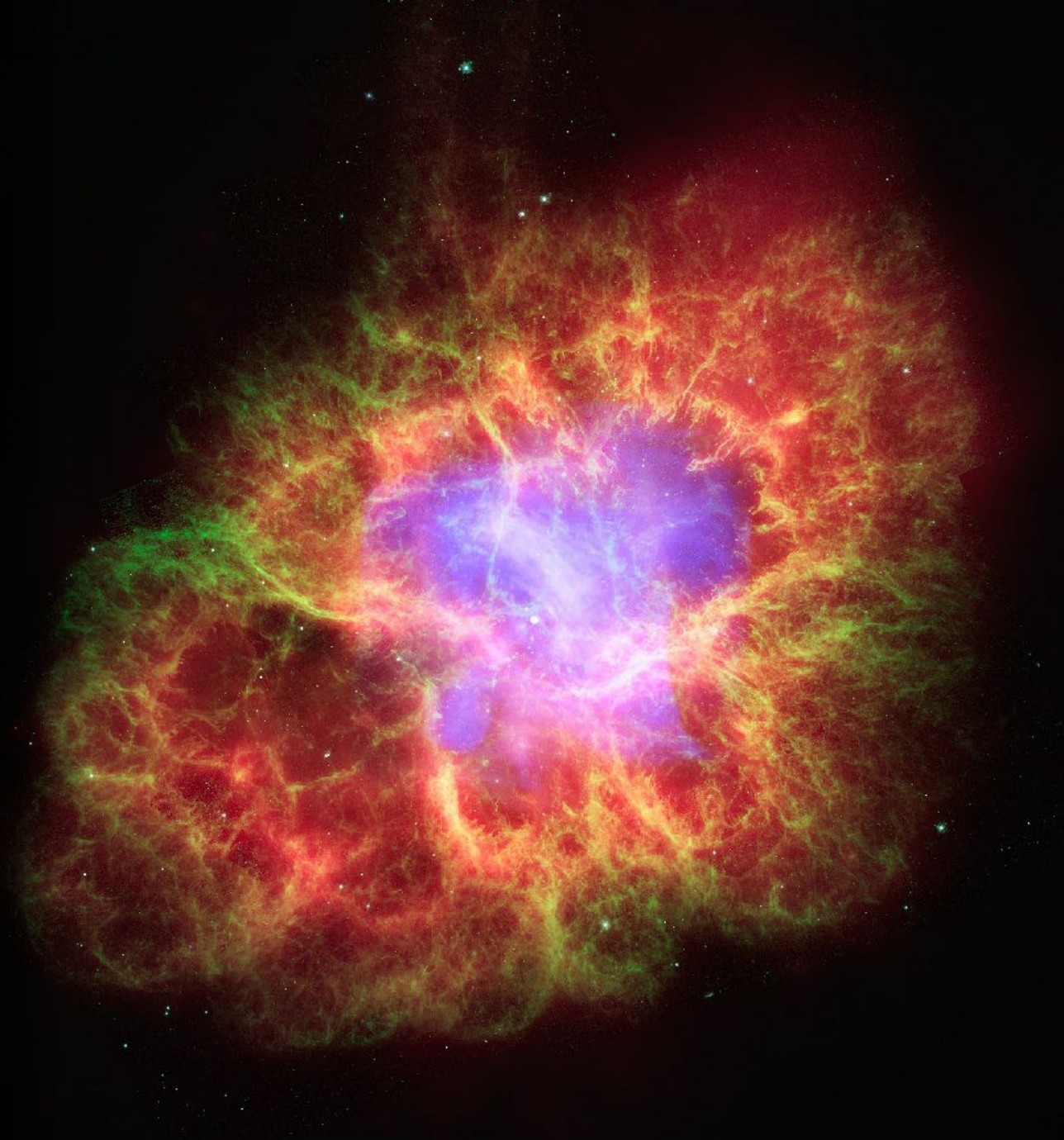
SOCALABS

ANNOUNCING "VERIFIED BY PLUGINVAL" SCHEME

- Send us a conforming log file and we'll put you on our website
- Links to conforming plugins

SUMMARY

- Developing plugins is hard
- Compounded by a complex landscape
- Catching bugs/issues early saves time
- Automating testing make this quicker and simpler
- `pluginval` can be added to your toolchain for extra test coverage
- Can be used to speed up the bug reporting process
- “Verified by pluginval” scheme promotes best practices



QUESTIONS?

- Presentation available on GitHub: https://github.com/drowaudio/presentations
- Twitter: @drowaudio