CHUCK RACKS
Text-based Music Programming for the Digital Audio Workstation

Jordan Hochenbaum
Spencer Salazar
Rodrigo Sena

California Institute of the Arts
Valencia, CA

ICMC 2017 | Shanghai, China
Chuck Racks

Chuck Racks is an audio plugin for working with the ChucK programming language.

Bridges the world of the DAW with the world of text-based music programming.

Highlights:
- Host tempo synchronization
- Audio processing + synthesis
- MIDI input + output
- Parameter automation
Motivation

Digital audio workstation software has greatly expanded access to computer music technology, with concomitant limitations.
Motivation

Computer music programming languages offer sonic/compositional possibilities not available in DAWs.

DAWs offer distinctive compositional and sonic features.
Motivation

Computer music programming languages offer sonic/compositional possibilities not available in DAWs (algorithm, process, sound design)

DAWs offer distinctive compositional and sonic features
Motivation

Computer music programming languages offer sonic/compositional possibilities not available in DAWs (algorithm, process, sound design)

DAWs offer distinctive compositional and sonic features (timeline, graphical notation, audio tracking, plugin ecosystem)
Applications

Music composition and production
Audio algorithm prototyping
Computer music education
Background

ChucK, miniAudicle, Audicle
Background

Max for Live

Community “devices”
Background

Reaktor, Kontakt

block flow + digital signal processing

sampling + scripting
Background

ixi lang, Gibber, Sonic Pi

textual/visual paradigms
for symbolic music representation

d = Drums("xoxo");
Chuck Racks

• VST/AudioUnit plugin

• Embedded full ChucK compiler and virtual machine

• Host integration features exposed to ChucK in PluginHost and PluginParameter API
Host Integration

Tempo synchronization
Audio processing
MIDI Input + Output
Parameter Automation
Tempo Synchronization

Chuck Racks automatically syncs with host tempo and transport state

Chuck Event system
Reacts to changes in transport state and tempo
Tempo Synchronization

PluginHost.getTempo();

PluginHost.bar() => now;
PluginHost.half() => now;
PluginHost.quarter() => now;
PluginHost.eighth() => now;
PluginHost.sixteenth() => now;

PluginHost.isPlaying();
PluginHost.posInBeat();
PluginHost.posInBar();

// etc...
Audio Processing

ChucK dac and adc unit generators

Synthesis, analysis, and audio processing programs function with no changes
Audio Processing

Processes synthesized or pre-recorded audio from host

Can be further processed by other plugins (compression, EQ, effects, etc.)

Can be executed live or bounced to a recording
MIDI Input + Output

Process MIDI from the host
MIDI sequence, input device, generator, etc.

Generate and send MIDI back to the host
MIDI Input + Output

MIDI-controlled synthesis

Algorithmic composition

Algorithmic parameter mapping (e.g. control change)
MIDI Input + Output

PluginHost.onMidi() => now;

MidiMsg msg;
PluginHost.recvMidi(msg);

PluginHost.sendMidi(msg);
Parameter Automation

- Arbitrary user-created parameters
- Mapped to musical parameters within code
- Controlled by host facilities: knobs, MIDI controls, automation
Parameter Automation

- Graphical control of musical parameters over time
- Not typically available in text-based music programming!
Parameter Automation

PluginParameters.getValue("name")
Technical Details

JUCE-based plugin architecture
multiple frameworks/platforms
(VST, AU, RTAS, AAX, etc.,
Mac, Windows, Linux)

libchuck embeddable ChucK engine
(limited to single instance)
Example: Synthesizer

```plaintext
TriOsc s => ADSR adsr => dac;
MidiMsg msg;

while(true)
{
    PluginHost.onMidi() => now;
    while(PluginHost.recvMidi(msg))
    {
        if(midi.data1 == 144)
        {
            midi.data2 => Std.mtof => s.freq;
            1 => adsr.keyOn;
        }
        else if(msg.data2 == 128)
        {
            1 => adsr.keyOff;
        }
    }
}
```
Example: Audio Effect

```plaintext
adc => Gain g => dac;
SinOsc s => g;
3 => g.op; // multiply

while(true)
{
    Math.sin(now/second*0.15*2*pi)*4000 => s.freq;
    5::ms => now;
}
```
Example: Audio Effect with Param

```plaintext
adc => Gain g => dac;
SinOsc s => g;
3 => g.op; // multiply

while(true)
{
    PluginParameters.getValue("freq")*4000 => s.freq;
    5::ms => now;
}
```
Discussion and Future Work

Graphical/timeline representations in text-based music programming are interesting.

Musical practices of hybrid DAW/programming?

Community building and packaging.
Conclusion

Chuck Racks is an audio plugin for working with the ChucK programming language.

Bridges the world of the DAW with the world of text-based music programming:

- Host tempo synchronization
- Audio processing + synthesis
- MIDI input + output
- Parameter automation
Thanks!

http://mtiid.calarts.edu/projects/software/chuck-racks/
https://github.com/mtiid/chuck-racks

Spencer Salazar
ssalazar@calarts.edu

Questions?