Ingen: A Meta-Modular Plugin Environment

David Robillard

School of Computer Science
Carleton University

April 10, 2015

Introduction	Design	Applications	The Glorious Future
●○○	○○	○○	○○
○○○○	○○○○○○	○○○	

Overview

"Ingen"

- Nobody? Nothing?
- That company from Jurassic Park?
- . . .
- Instrument Generator, "engine"

What is Ingen?

- ▶ Ingen is a modular synth, effects processor, mixer, router, MIDI processor, cheese grater, chain degreaser. . .
- ▶ . . . a "modular"
- ▶ Designed around LV2 plugins
- ▶ Usable in many different contexts

Introduction	Design	Applications	The Glorious Future
○○●	○○	○○	○○
○○○○	○○○○○○	○○○	

Overview

Features

- ▶ Polyphonic
- ▶ Recursive (graphs within graphs)
- ▶ Many data types (including events like MIDI)
- ▶ Strict client/server architecture
  - ▶ Real-time editable
  - ▶ Flexible deployment

Philosophy

- ▶ Do one thing and do it well
- ▶ Integrate with the surrounding environment
- ▶ Make use of existing facilities

Existing Facilities?

- Plugins!
- Internals Considered Harmful

Surrounding Environment

In the Lignux audio world, there are several choices:

- ▶ JACK application
- ▶ LV2 plugin
- ▶ Network service
- ▶ Physical device (e.g. MOD)

Philausophy

- ▶ Do one thing and do it well and do it with real-time safety
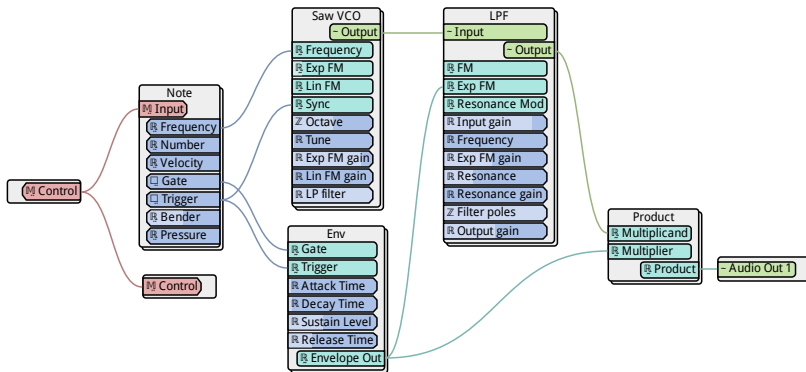- ▶ Keep GUIs at arm's length

Terminology

Ingen contains:

- Graphs
- Blocks
- Ports
- Arcs

Introduction
000
0000

Design
○●
○○○○○○

Applications
○○
○○○

The Glorious Future
○○

Terminology

A Graph

Data Model

Everything is described in a simple data model:

- ▶ All objects have a path, e.g. /amp/gain
- ▶ Objects are a dictionary of <u>properties</u> (key:value)
- ▶ Inherently extensible: keys can be added without breakage

Introduction	Design	Applications	The Glorious Future
ooo	oo	oo	oo
oooo	oeoooo	ooo	

Data Model / Protocol

Data Model Example

A snippet of an Ingen graph:

```
</osc>
  a              ingen:Block ;
  lv2:prototype  <urn:someplugin> ;
  ingen:canvasX  42.0 ;
  ingen:canvasY  24.0 .
```

Introduction	Design	Applications	The Glorious Future
ooo	oo	oo	oo
oooo	oo●ooo	ooo	

Data Model / Protocol

Protocol

Ingen is controlled by manipulating the model:

- ▶ Changes are based on property manipulation
- ▶ Everything done with a few generic methods: get, set, put, . . .
    - ▶ Vaguely HTTP-like
- ▶ No special "commands" for specific objects
    - ▶ No: block.move_to(42, 0)
    - ▶ Instead: set block.x $= 42$, block.y $= 0$

Introduction
000
0000

Design
00
000●00

Applications
00
000

The Glorious Future
00

Data Model / Protocol

Simple Message Example

Moving a block on the canvas:

```
[
  a               patch:Set ;
  patch:subject   </osc> ;
  patch:property  ingen:canvasX ;
  patch:value     42.0 ;
]
```

Introduction	Design	Applications	The Glorious Future
000	00	00	00
0000	000000	000	

Data Model / Protocol

Slightly Less Simple Message Example

To add, or <u>put</u>, a new block:

```
[
  a             patch:Put ;
  patch:subject </osc> ;
  patch:body [
    a             ingen:Block ;
    lv2:prototype <urn:someplugin> ;
    ingen:canvasX 42.0 ;
    ingen:canvasY 24.0 ;
  ]
]
```

Introduction	Design	Applications	The Glorious Future
000	00	00	00
0000	000000●	000	

Data Model / Protocol

Who Cares?

- ▶ Elegant correspondence between protocol and data model
- ▶ Identical syntax used on the wire and in saved files
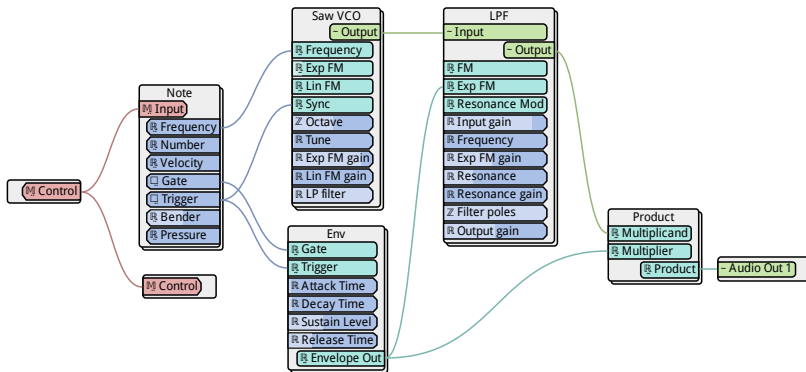- ▶ Conceptually simple

Introduction	Design	Applications	The Glorious Future
000	00	●0	00
0000	000000	000	

Uses

No, really... who other than you cares?

- ► :(
- ► Allows flexible deployment
- ► Ingen as server, network controlled from a different machine
  - ► Dump text protocol for a plain text log of activity
- ► Run Ingen as an LV2 plugin, control via LV2 ports
  - ► Protocol in binary (LV2 atoms, native `float`, etc)
  - ► No shady underhanded communication
  - ► GUI talks to engine via ports like any LV2 plugin
- ► Extensible, properties can be added freely
- ► A few methods are capable of everything

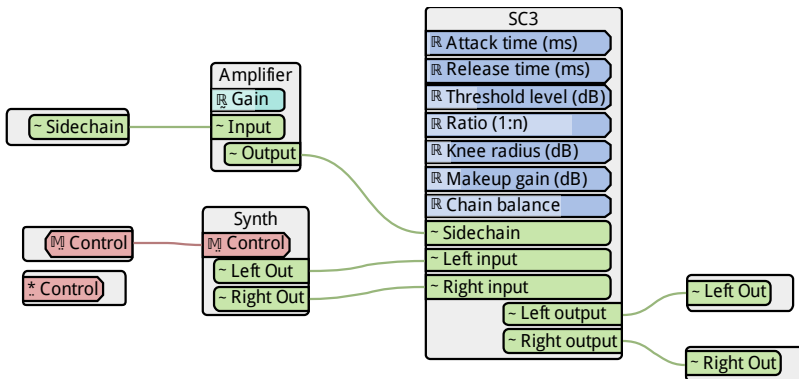| Introduction	Design	Applications	The Glorious Future
000	00	0●	00
0000	000000	000	

Uses

Scenarios

- ▶ Need to put some non-linear in another host (e.g. Ardour mixer strip)
- ▶ Remote-controllable "glue" for a headless JACK box
- ▶ Want to publish a plugin but can't or don't want to code
  - ▶ Ingen graphs are LV2 plugins
  - ▶ . . . literally. No special export, no compilation; the one and only save format is LV2 compatible
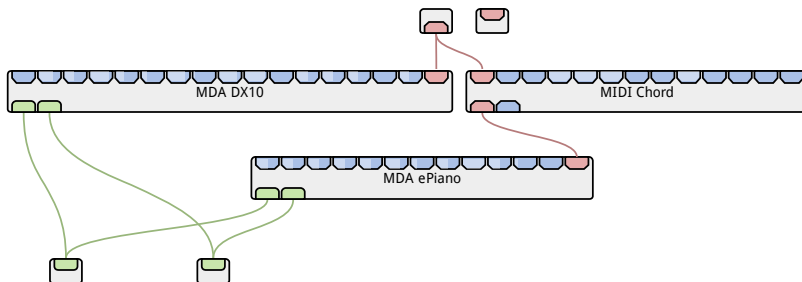 - ▶ Save to LV2_PATH (typically ~/.lv2) and graphs will be visible in any LV2 host

Introduction	Design	**Applications**	The Glorious Future
000	00	00	00
0000	000000	●00	

Graphs Whose Key Feature Is Fitting On One Slide

A Synth

Introduction
000
0000

Design
00
000000

Applications
00
0●0

The Glorious Future
00

Graphs Whose Key Feature Is Fitting On One Slide

Flexible I/O (Sidechains)

Introduction
ooo
oooo

Design
oo
oooooo

Applications
oo
oo●

The Glorious Future
oo

Graphs Whose Key Feature Is Fitting On One Slide

MIDI Filtering

Introduction	Design	Applications	The Glorious Future
000	00	00	●○
0000	000000	000	

Things drobilla would rather be doing than what he gets paid to do

Mundane Improvements

- ▶ Performance / memory consumption improvements
- ▶ Better state/preset support
- ▶ Stability, more rigorous testing
- ▶ GUI polish

Introduction
000
0000

Design
00
000000

Applications
00
000

The Glorious Future
0●

Things drobilla would rather be doing than what he gets paid to do

Whiz-Bang

- ▶ Control panel building (presentation mode)
- ▶ Novel presentation of patching canvas? (Pies?)
- ▶ Integrated online patch repository (MOD?)
- ▶ Dynamic ports when running as LV2 plugin
- ▶ Design plugin suite for message-based programming
- ▶ Make it as simple as possible for users to publish "plugins"