Reflection in Pure Data

IOhannes m zmölnig
zmoelnig@iem.at

Institute of Electronic Music and Acoustics
University of Music and Dramatic Arts
Graz, Austria

Overview

- motivation
- reflection (super-short)
- self-examination/modification in Pd
- iemguts
  - design ideas
  - examples
- uses
- outlook

IOhannes m zmölnig
zmoelnig@iem.at

Motivation

- agent-like system
  - esp. in visual programming languages
- live coding
  - adding “funny” (easily comprehensible) features
    - moving objects
  - coding with the computer as “opponent/partner”

IOhannes m zmölnig
zmoelnig@iem.at

Reflection

- “process of reasoning about and/or acting upon oneself” [Demer/Malenfant 1995]
- Self examination
  - examine run-time behaviour
- Self modification
  - changing running program
- Self replication

Definitions

- object
  - Pd-object (“rectangles”)

- abstraction
  - external object written in Pd
- canvas
  - a Pd window where you can put objects
  - patch, sub-patch, abstraction

Reflection and Pd

- Self modification
  - interpreter and IO (GUI, file,...) communicate via Pd-messages
  - → generate messages to control the interpreter
- Self examination
  - few constructs to query the interpreter

Self-examination in Pd

- *some* functionality that allows examination by user
  - e.g. numberboxes
- *few* objects that allow patch to examine itself
  - [cputime]
  - [realtime]
- *message domain only!*
  - signal-domain usually has constant load (less problematic)

IOhannes m zmölnig
zmoelnig@iem.at

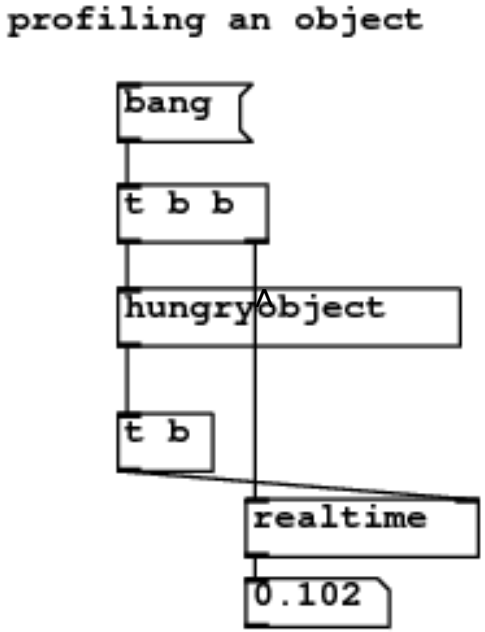
[cputime]

- featured in “Load Meter” (*top* for Pd interpreter)
- turn on/off patches depending on the available processing power
  - only takes Pd-process into account
  - estimation!
  - could be interesting during performance
- stress-tests
  - generating cpu-load in the process
  - interesting during development (stability tests)

IOhannes m zmölnig
zmoelnig@iem.at

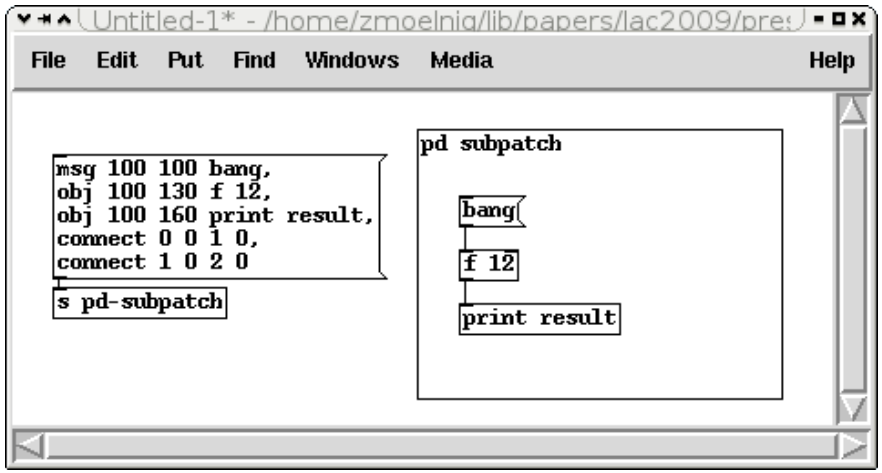
[realtime]

- confusion about “logical” vs “real” time
- measuring the “real” (system) time elapsed between two events
- profiling!
- performing?



Self-modification in Pd

- “dynamic patching”
  - messages to canvas



- exactly the same happens when reading a patch from file
  - additive (restore from file)

```
#X msg 100 100 bang;
#X obj 100 120 print;
#X connect 0 0 1 0;
```

Self modification continued

- more complex self-modification
  - mimicking user-interaction
  - sending e.g. mouse events

    `click 5 5 0 0 0, mouseup 200 200 0`
  - requires high knowledge of the (visual representation of the) patch

IOhannes m zmölnig
zmoelnig@iem.at

Woes of dynamic patching

- indices
  - manually keeping track of indices
  - indices can change on object-deletion
- limited functionality
  - adding made easy
  - programmatic deletion of specific objects complicated
- user-interaction
  - concurrent (virtual) mouse pointers
  - ...

Mitigation of dynamic patching

- “dyn~” (Thomas Grill)
  - sandbox “canvas”
  - extended set of methods
    - object “labels” rather than indices
    - object deletion
  - top-down approach

IOhannes m zmölnig
zmoelnig@iem.at

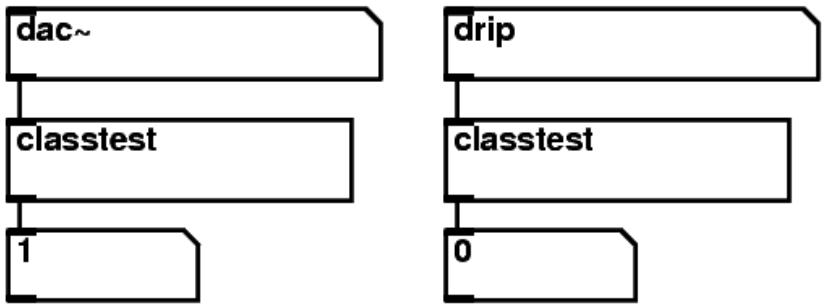
iemguts

- making dynamic patching easier
- exposing internals of Pd on the patch-level
  - functionality (not) available through *public* “C” API
- bottom-up approach
  - `$self` metaphor
- global state

IOhannes m zmölnig
zmoelnig@iem.at

global state of interpreter

- [dspstate] would be nice :-(
- [classtest]
  - test whether an objectclass is loaded into Pd



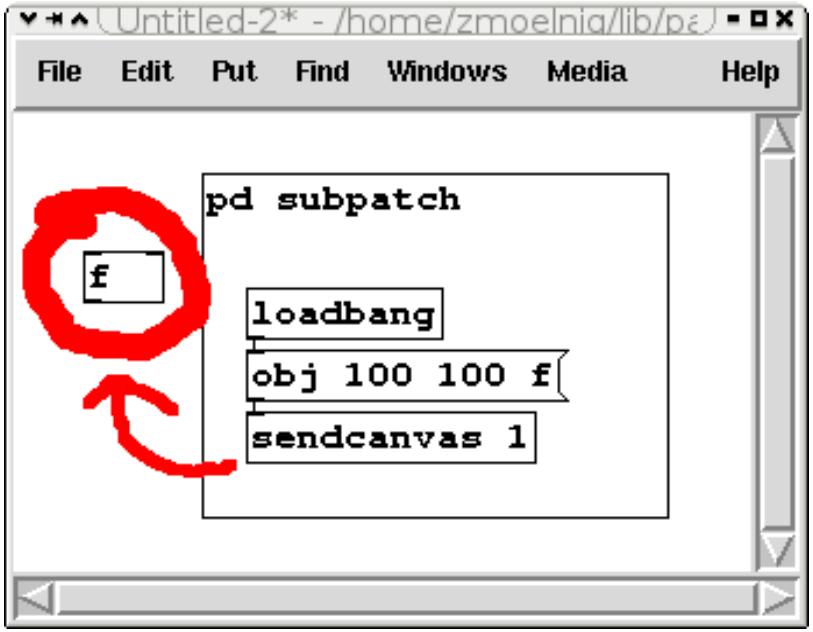
self

- provide information about abstraction itself *only*
- depth
  - iemguts-objects work on abstractions they live in (or parents thereof)
  - [canvasdollarzero 2]
    - information on “grandparent” canvas

IOhannes m zmölnig
zmoelnig@iem.at

talking to oneself

- [sendcanvas]
  - communication with self (or rather: self's parent)



more selfishness

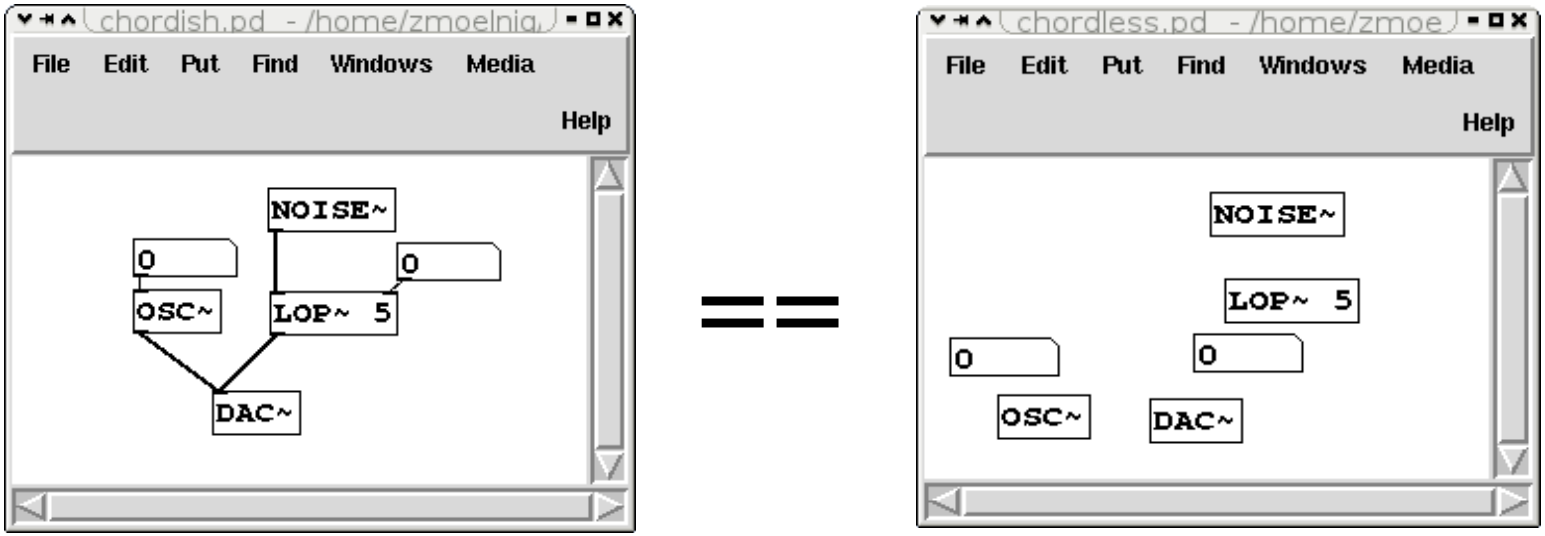
- neighbours
  - information about “neighbouring” abstractions have to be queried from neighbours
  - implementation in plain Pd
- how to deal with *3rd party* objects?
  - esp. *internals*

position in space

- Pd is a *visual* language
- objects relate to each other in space
  - not quite true in language semantics
- [canvasposition]
  - query position of self
  - change position of self

chordless patching

- *reacTable* like interfaces



IOhannes m zmölnig
zmoelnig@iem.at

connections

- objects explicitly relate to each other through connections
- [canvasindex]
  - query the index of self
  - allows dynamic construction of "`connect <ob0> 0 <ob1> 0`" messages

connections (cont.)

- no awareness of what connects or what is connected to self (functional approach)
- [canvasconnections]
  - number of inlets/outlets
  - type (signal vs message)
  - existing connections
- automatic patcher

IOhannes m zmölnig
zmoelnig@iem.at

deletion

- usually, Pd is not very stable when objects try to delete themselves
- [canvasdelete]
  - allows safe self-destruction
    - objects with limited lifetime
    - temporary objects
- TODO
  - intercept user-triggered deletion process

IOhannes m zmölnig
zmoelnig@iem.at

persistency

- allow objects to save their current state
- [canvasargs]
  - settable arguments for next duplication/save
  - objects have to ensure themselves, that all relevant parameters are stored
  - persistency hooks into Pd's “save” mechanism
    - simple
    - single-level (only abstractions in saved patch are affected)

IOhannes m zmölnig
zmoelnig@iem.at

mutation

- objectargs
  - [canvasargs]
- entire objects
  - [canvasname]
    - versioning system
    - ...

IOhannes m zmölnig
zmoelnig@iem.at

use cases

- live-coding performances
  - “live” (moving) patches
  - autoconnecting
- patching helpers
  - Luke Iannini: [templater]

IOhannes m zmölnig
zmoelnig@iem.at

Future

- directly querying/setting information of 3rd party objects on the canvas
  - [canvas...] → [canvasobject...]
  - query index resp. name/args
  - connections
- deletion-hook
- ...

Thank you

https://pure-data.svn.sourceforge.net/
/svnroot/pure-data/trunk/externals/iem/iemguts

IOhannes m zmölnig
zmoelnig@iem.at