

pnpd/nova

Tim Blechmann
tim@klingt.org

March 13, 2007

Introduction

The Dataflow Language

Implementation

Future

Dataflow Programming Languages

- ▶ 'max-like' languages: Max/MSP, Pure Data, jmax
- ▶ Labview
- ▶ vvvv
- ▶ ...

Dataflow Programming Languages

- ▶ nova is a 'max-like' programming language
- ▶ additional features
 - ▶ namespaces
 - ▶ powerful data structures
 - ▶ data encapsulation ('object orientation')
 - ▶ hierarchic bind system
- ▶ text-based and graphical syntax
- ▶ command line interpreter (controllable via OSC)

Motivation

- ▶ 'max paradigm' is very powerful & easy to learn for non-programmers (like most musicians)
- ▶ current implementations of the 'max paradigm' lack certain features
- ▶ Max/MSP is closed-source
- ▶ Pd is open source, but the codebase is not scaleable
- ▶ student project (software engineering lab, bachelor thesis ...)

The Patcher Language

- ▶ 'max-like' patcher language, describing the state of the system
- ▶ objects are placed onto a context (the 'canvas')
- ▶ contexts ('canvases') are nested, subpatches, abstractions
- ▶ graphical and text-based patcher language
- ▶ message/signal separation

Message Type System

- ▶ Bang
- ▶ Float
- ▶ Integer (exact, multiprecision)
- ▶ Symbol (hashed string)
- ▶ String
- ▶ List
- ▶ Pointer (for adding custom message types)

The Text-based Patcher Language

Hello World

```
{  
  signal = osc~<440>()  
  dac~(signal[0], signal[0])  
}
```

The Text-based Patcher Language

Hello World

```
{  
  signal = osc~<440>()  
  dac~(signal[0], signal[0])  
}
```

With explicit connections

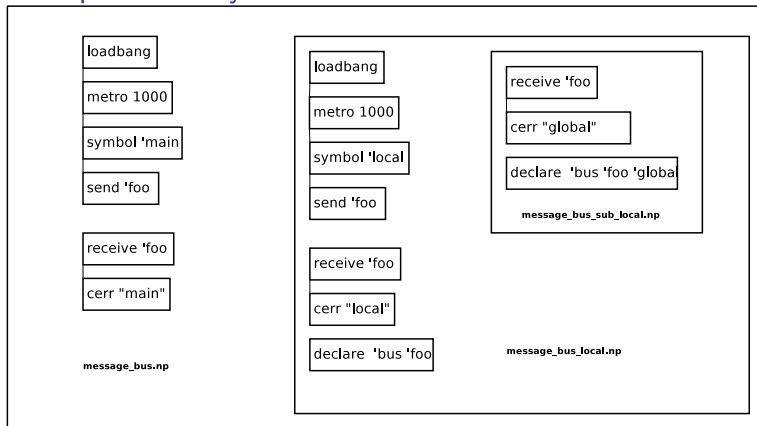
```
{  
  signal = osc~<440>()  
  out = dac~()  
  signal -> out  
  signal -> out[1]  
}
```

Encapsulation

- ▶ Patches can have `loadbang` or `endbang` objects ‘constructors’ or ‘destructors’
- ▶ Before `loadbang`s and after `endbang`s, no communication with other patches is allowed
- ▶ Incoming messages (e.g. from message busses or message inlets) will be queued

Object Bindings

Example hierarchy



Object Library

- ▶ Most objects with equivalents in the core language of Pd
- ▶ Many objects with equivalents in Max/MSP
- ▶ OSC network objects
- ▶ Python scripting objects
- ▶ It is easy to extend the object library from C++

Facts

Facts

- ▶ interpreter written in C++, patch-generator & gui in Python
- ▶ available as standalone interpreter, shared library & (in future) as Python module
- ▶ heavily based on the boost C++ libraries

Facts

Facts

- ▶ interpreter written in C++, patch-generator & gui in Python
- ▶ available as standalone interpreter, shared library & (in future) as Python module
- ▶ heavily based on the boost C++ libraries

Ambitions

- ▶ efficient dsp core
- ▶ avoid audio dropouts (even under situations of high load)

Dropout avoidance

- ▶ don't block the audio thread
- ▶ don't allocate (large chunks of) memory in the audio thread
- ▶ use lock-free data structures for synchronization
- ▶ but: messaging is done in the audio thread to ensure a tight message/audio interaction

Performance

- ▶ precompiled dsp chain (topological sorted dsp graph)
- ▶ simd (sse instruction set)
 - ▶ vector operations (add, multiply, copy, set...)
 - ▶ simple vectorizable operations (min/max/clip)
 - ▶ other vectorizable operations (abs, sign, denormal bashing)
 - ▶ accumulating operations (peak finding & power calculations)
- ▶ compile-time loop unrolling with template metaprogramming techniques
- ▶ special ugens for special situations (e.g. message- vs. signal-driven)

Benchmark

General purpose benchmark patch

- ▶ filtered noise (lowpass, highpass, bandpass)
- ▶ fm synth with linear envelope
- ▶ simple sample recorder / playback
- ▶ signal-driven sample playback (sample&hold and 4-point interpolation)
- ▶ delay line
- ▶ audio input & clipped audio output

Benchmark

General purpose benchmark patch

- ▶ filtered noise (lowpass, highpass, bandpass)
- ▶ fm synth with linear envelope
- ▶ simple sample recorder / playback
- ▶ signal-driven sample playback (sample&hold and 4-point interpolation)
- ▶ delay line
- ▶ audio input & clipped audio output

Benchmark results

(200000 CPU.CLK.UNHALT events, pentium-m 750)

Benchmark

General purpose benchmark patch

- ▶ filtered noise (lowpass, highpass, bandpass)
- ▶ fm synth with linear envelope
- ▶ simple sample recorder / playback
- ▶ signal-driven sample playback (sample&hold and 4-point interpolation)
- ▶ delay line
- ▶ audio input & clipped audio output

Benchmark results

(200000 CPU.CLK.UNHALT events, pentium-m 750)

- ▶ Pure Data: 27065 (Pd-0.40)
- ▶ nova: 11537 (recent svn)

Todo List: Must Have

- ▶ working graphical patcher
- ▶ extend the object library
- ▶ documentation of the patcher language & object library
- ▶ find some users
- ▶ testing, testing, testing . . .
- ▶ porting to x86_64
- ▶ porting to osx and win32

Todo List: Nice To Have

- ▶ distribute the dsp chain to several cpus
- ▶ running nova as plugin (ladspa/dssi/vst...)
- ▶ pd/max external compatibility layer