5 years of using SuperCollider in real-time interactive performances and installations
-
Retrospective analysis of *Schwelle*, *Chronotopia* and *JND/Semblance*

Marije A.J. Baalman

Overview

- Introduction
- Coding in the context of interactive performance
- The artistic projects
- Common techniques
- Software tools made public
- Conclusions

Why this paper?

5 years of writing code for professional artistic works

- Context of real-time interactive performance and installation
- Collaborative projects

Tension between on-the-fly solutions and general purpose tools

- Evaluation of approaches
- Identifying common problems...

... and solutions!

Coding in the context of interactive performance

Flexible, but robust and easy-to-start

Purpose of the code is not known from the start, but emerges during the process of artistic creation
- *the problem to solve keeps redefining itself*

Flexible, but robust and easy-to-start

Purpose of the code is not known from the start, but emerges during the process of artistic creation
- *the problem to solve keeps redefining itself*

Flexible system, capable of on-the-fly manipulation
- *livecoding*
- understanding of movement and interactions

Flexible, but robust and easy-to-start

Purpose of the code is not known from the start, but emerges during the process of artistic creation
- *the problem to solve keeps redefining itself*

Flexible system, capable of on-the-fly manipulation
- *livecoding*
- understanding of movement and interactions

Robust system for “show control”, but flexible to adapt to different theater/gallery/...
- and flexible for rehearsal

Flexible, but robust and easy-to-start

Purpose of the code is not known from the start, but emerges during the process of artistic creation
- *the problem to solve keeps redefining itself*

Flexible system, capable of on-the-fly manipulation
- *livecoding*
- understanding of movement and interactions

Robust system for “show control”, but flexible to adapt to different theater/gallery/...
- and flexible for rehearsal

Easy-start solution for installations

The artistic projects

The projects

Collaborations with artist/researcher Chris Salter
Two dance performances...
... and one installation

- Realtime sensor data
- Audio and data analysis
- Data exchange with other software/collaborators
- Show control
- Sound, vibration and light output

Schwelle

Schwelle is a theatrical performance that takes place between a solo dancer/actor (Michael Schumacher) and a “sensate room”

Interactive light design by Harry Smoak

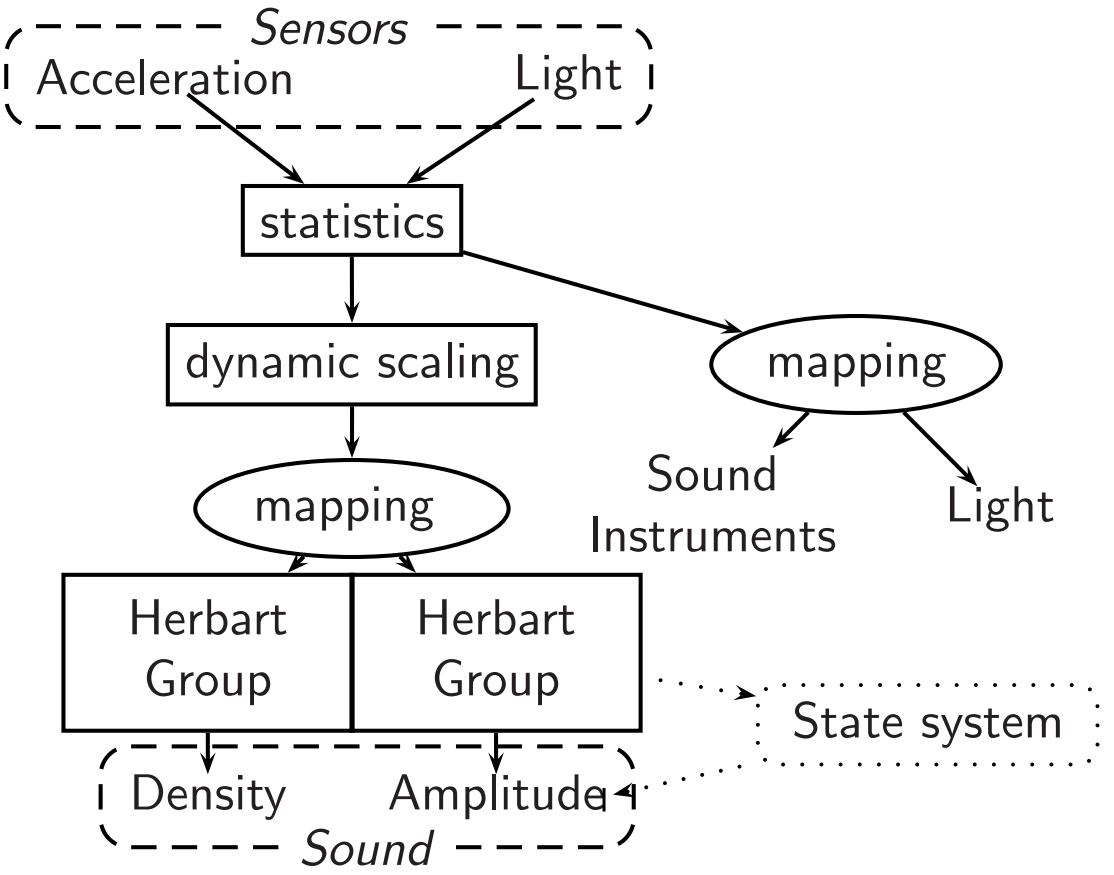
Schwelle

Schwelle is a theatrical performance that takes place between a solo dancer/actor (Michael Schumacher) and a “sensate room”

- Sensors on the body (acceleration) and in the room (light)
- *Adaptive audio and light scenography*
- Spatialisaton and submixing of audio
- Multi-layered data processing flow

Schwelle

Schwelle is a theatrical performance that takes place between a solo dancer/actor (Michael Schumacher) and a “sensate room”



Chronotopia

Chronotopia - a dance piece with the Attakkalari Centre for Movement (India), music from Matthias Duplessy, video from Christian Ziegler, interactive light installation controlled from SuperCollider

premiere at the Attakkalari India Dance Biennial 2009 (2 shows) and 2 other shows in India; German-Swedish tour in March 2010.

(video, performance March 2010, at the Mousonturm, Frankfurt, Germany)

Chronotopia

Chronotopia - a dance piece with the Attakkalari Centre for Movement (India), music from Matthias Duplessy, video from Christian Ziegler, interactive light installation controlled from SuperCollider

Hardware: light matrix and handheld lights - camera based videotracking

- 6 control boards for 6 CCFL lights based on XBee-Arduino
- 3 handheld CCFL lights with SenseStage MiniBee
- Wireless control over these lights

Chronotopia

Chronotopia - a dance piece with the Attakkalari Centre for Movement (India), music from Matthias Duplessy, video from Christian Ziegler, interactive light installation controlled from SuperCollider

- **Synths** outputting to controlrate buses
- *sclang* polling values to send to the wireless coordinator
- **Patterns** for sequencing

- Motion tracking
- Pitch, beat and amplitude tracking on soundtrack
- Data exchange of maximum amplitude and frame time with computer controlling interactive video

JND/Semblance

JND/Semblance - a one-person installation piece by Chris Salter, Marije Baalman and Harry Smoak, interactive sound, light and vibration controlled from SuperCollider

public preview at Empac, Troy, NY, March 3-7, 2010

ISEA in Essen, Germany, in August 2010

Today's Art in The Hague, The Netherlands, in September 2010

JND/Semblance

JND/Semblance - a one-person installation piece by Chris Salter, Marije Baalman and Harry Smoak, interactive sound, light and vibration controlled from SuperCollider

- 24 pressure sensing pads made of paper, data received wireless
- 12 speakers/vibrators
- Control over lights via DMX

JND/Semblance

JND/Semblance - a one-person installation piece by Chris Salter, Marije Baalman and Harry Smoak, interactive sound, light and vibration controlled from SuperCollider

- 24 pressure sensing pads made of paper, data received wireless
- 12 speakers/vibrators
- Control over lights via DMX

- New medium of tactile vibrations
- ... hard to separate from acoustic soundtrack
- Statistical analysis of sensor data
- ... mapping varying depending on section in the piece
- Spatialisation of sound over a 2 by 6 grid

Intermezzo - SuperCollider nomenclature

Briefly... SuperCollider

Two components communicating via OSC

- *sclang* — audio programming language
- *scsynth* — audio synthesis engine

UGen unit generator, or its representation in *sclang*.

SynthDef "blueprint" for a Synth, like an "instrument", consisting of a set of interconnected **UGens**.

Synth a running synthesis node on *scsynth*, created from a **SynthDef**; like a "voice".

Quark "packaged" set of *sclang* classes to extend the default class library of *SC3*.

SuperCollider can be found at
`http://supercollider.sourceforge.net`

Common techniques

Collecting sensor data

Hardware and protocols

- In *Schwelle*: Create USB devices — HID devices,
- ... and a WiiMote.
- In *Chronotopia*: MotionTracker — OSC input
- In *JND/Semblance*: Wireless, XBee based, sensing — Serial Port

Collecting sensor data

Hardware and protocols

- In *Schwelle*: Create USB devices — HID devices,
- ... and a WiiMote.
- In *Chronotopia*: MotionTracker — OSC input
- In *JND/Semblance*: Wireless, XBee based, sensing — Serial Port

- In *Schwelle*: abstraction between a class **SchwelleSensor** and classes gathering the HID data (one for Linux, one for OSX).
- Subclasses for different variants of **SchwelleSensor** — WiiMote backend, mixing sensors, dummy sensor...

In the other projects a generalised abstraction was used, the **SenseWorld DataNetwork**.

Processing sensor data

SchwelleSensor made use of the class **SensorData** to do statistical processing on the data. This was all in *sclang*.

Processing sensor data

SchwelleSensor made use of the class **SensorData** to do statistical processing on the data. This was all in *sclang*.

In the later projects, move of this processing to *scsynth* — efficient DSP algorithms.

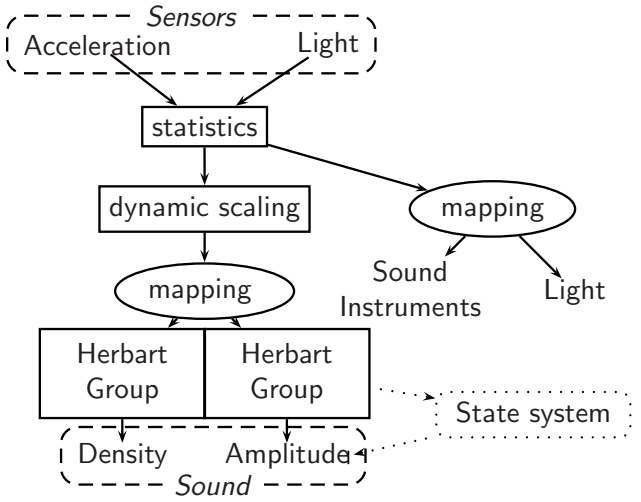
DataNetwork as a central hub for all datastreams.

Mapping sensor data

- Remapping value ranges
- Merging of data streams
- Extracting features
- Creating dynamical processes

Mapping sensor data

- Remapping value ranges
- Merging of data streams
- Extracting features
- Creating dynamical processes



- **SchwelleSensorSystem** managing interactions in dataflow path
- **DynamicScaleSystem** handling dynamic scaling
- **SchwelleHerbart** handling the Herbart system

All in *sclang*, with a lot of cross-referencing between classes.

Mapping sensor data

- Remapping value ranges
- Merging of data streams
- Extracting features
- Creating dynamical processes

In *Chronotopia* and *JND/Semblance* data processing centered around the **DataNetwork**.

Much more flexible

More processing takes place on *scsynth*

Not all algorithms from *Schwelle* have been ported to the **DataNetwork** yet.

Mapping sensor data

- Remapping value ranges
- Merging of data streams
- Extracting features
- Creating dynamical processes

For *JND/Semblance* start work on a Preset system

- Set parameters for specific **Synths**
- Mapping to specific datastreams from **DataNetwork**

Data exchange with other software

In all projects one of the collaborators was using *Max/MSP* to control lights or video.

Data exchange with other software

In all projects one of the collaborators was using *Max/MSP* to control lights or video.

In *Schwelle* custom OSC-namespace, defining addresses for each stream of data

Data exchange with other software

In all projects one of the collaborators was using *Max/MSP* to control lights or video.

In *Schwelle* custom OSC-namespace, defining addresses for each stream of data

In *Chronopia* and *JND/Semblance* use of the data sharing component of the **DataNetwork**

General approach

Data exchange with other software

In all projects one of the collaborators was using *Max/MSP* to control lights or video.

In *Schwelle* custom OSC-namespace, defining addresses for each stream of data

In *Chronopia* and *JND/Semblance* use of the data sharing component of the **DataNetwork**

General approach

For the DataNetwork see also the talk tomorrow on SenseStage

Managing synthesis processes

Within SuperCollider two main methods of creating and controlling **Synth** processes on the server:

- Direct instantiation and controlling parameters either manually or automated with tasks.
- Using the **Pattern** sequencing library — high level mechanisms

Managing synthesis processes

In *Chronopia*, extensive use of the **Pattern** library with some direct **Synth** instances, mapping parameters to control buses.

Managing synthesis processes

In *Chronopia*, extensive use of the **Pattern** library with some direct **Synth** instances, mapping parameters to control buses.

For *Schwelle*:

- Class **SchwelleInstrument**, taking care of creating and controlling of **Synths**, and submixing the output
- Various subclasses dealing with different types of **Synths** or ways of control.
  - Using **Buffers**
  - Using audio input
  - Specific mappings to sensor data
  - Clouds of **Synths**
- Graphical user interface for starting and stopping synths, and controlling the volume.

Managing synthesis processes

In *Chronopia*, extensive use of the **Pattern** library with some direct **Synth** instances, mapping parameters to control buses.

For *Schwelle*:

- Class **SchwelleInstrument**, taking care of creating and controlling of **Synths**, and submixing the output
- Various subclasses dealing with different types of **Synths** or ways of control.
- Graphical user interface for starting and stopping synths, and controlling the volume.

For *JND/Semblance*:

- Central **JNDEngine**, handling all synths
- **JNDSynth** — control over settings and connections to the **DataNetwork**.
- Graphical user interface for starting and stopping synths, and manipulating settings, and controlling the volume.

Spatialisation methods

In *Schwelle*:

- **SchwelleSurround** class providing different spatialisation methods, implemented in **SynthDefs**
- Routing of **Synth** outputs to a spatialisation **Synth**

Spatialisation methods

In *Schwelle*:

- **SchwelleSurround** class providing different spatialisation methods, implemented in **SynthDefs**
- Routing of **Synth** outputs to a spatialisation **Synth**

In *Chronotopia*: Matrix of outputs — need for a Panner UGen to deal with this.

— At the time of creation only **PanAz** available.

— Now also **PanX**, which is more suitable for dealing with a row of speakers.

- Direct output to a specific channel
- Use of **PanX** in the **SynthDef**

Spatialisation methods

In *Schwelle*:

- **SchwelleSurround** class providing different spatialisation methods, implemented in **SynthDefs**
- Routing of **Synth** outputs to a spatialisation **Synth**

In *Chronotopia*: Matrix of outputs — need for a Panner UGen to deal with this.

- Direct output to a specific channel
- Use of **PanX** in the **SynthDef**

In *JND/Semblance*:

- Again extensive use of **PanX**
- Definition of a signal function, stored in **JNDSignalLib**
- Dynamical creation of **JNDSynthDefs** (using **SynthDef**::wrap) with different spatialisation methods
- All **JNDSynthDefs** stored in a separate **SynthDescLib**, which can be browsed from a GUI

Show control

Certain things need to happen at specific times — *Cues*

Show control

Certain things need to happen at specific times — *Cues*

In *Schwelle*

- cues often linked to performer's movements on stage — improvisation
- no absolute timing, sometimes relative timing — **ShowTimer**
- allocation (preparing an event) and freeing resources (cleaning up)

Text file with code and comments — *code as interface*

Show control

Certain things need to happen at specific times — *Cues*

In *Schwelle*

- cues often linked to performer's movements on stage — improvisation
- no absolute timing, sometimes relative timing — **ShowTimer**
- allocation (preparing an event) and freeing resources (cleaning up)

Text file with code and comments — *code as interface*

In *Chronotopia* — strict timing according to sound track — no improvisation
— **CueList** executing functions at a specific frame time

Show control

Certain things need to happen at specific times — *Cues*

In *Schwelle*

- cues often linked to performer's movements on stage — improvisation
- no absolute timing, sometimes relative timing — **ShowTimer**
- allocation (preparing an event) and freeing resources (cleaning up)

Text file with code and comments — *code as interface*

In *Chronotopia* — strict timing according to sound track — no improvisation
— **CueList** executing functions at a specific frame time

In *JND/Semblance* — initially using **Tdefs**, one for each movement.
— now moving towards a hierarchical **CueList**.

Show Control (2)

- Skipping back and forth, during rehearsals
- Quick editing of cues
- Manual vs. timed execution
- Preparation and cleanup of cues

Which cues are still having an effect at what time?

Summary

- Capturing, processing and sharing data is being consolidated in the **SenseWorld DataNetwork** framework
- *JND/Semblance* **SynthDef** creation and handling of **Synths** and presets is moving toward a *composition framework* interacting with the **DataNetwork**
- Show control is still an issue to solve

Software tools made public

SuperCollider extensions

- *sclang:* GeneralHID - cross platform abstraction to access HID devices
- *sclang:* WiiMote - access to the WiiMote
- *Quarks:* SenseWorld, — DataNetwork, and — MiniBee
- *Quark:* DMX - control over theatrical lights
- *UGen:* PanX - non-wrapping N-channel panner

Standalone programs

wiiosc Access the WiiMote and send OSC-messages; based on *libcwiid* and *liblo*.

motiontrackosc Access a camera image, detect motion, and send out information about areas where motion occurred; based on *OpenCV* and *liblo*.

Conclusion

Conclusion

- Interactive live performance = a challenging and exciting context for coding
- *SuperCollider* is certainly a suitable choice of language for this purpose

Conclusion

- Interactive live performance = a challenging and exciting context for coding
- *SuperCollider* is certainly a suitable choice of language for this purpose
- Creating tools as problems are encountered/invented
- Ad-hoc solutions for one performance — solid tools for subsequent works
- Publication of tools for use by other artists

Conclusion

- Interactive live performance = a challenging and exciting context for coding
- *SuperCollider* is certainly a suitable choice of language for this purpose
- Creating tools as problems are encountered/invented
- Ad-hoc solutions for one performance — solid tools for subsequent works
- Publication of tools for use by other artists

Some insights into:

- the creative process of working with code in artistic projects,
- and the specific challenges in this context.

Acknowledgements

- Chris Salter — *keeping me busy with all these projects :)*
- Harry Smoak — *initial development for data sharing...*
- Alberto de Campo — *many nice SC-coding sessions*
- Josh Parmenter — *writing **PanX***

Website

http://www.nescivi.nl

http://sensestage.hexagram.ca

Software available under the GNU/(L)GPL