THE IXI LANG:
A SUPERCOLLIDER PARASITE FOR LIVE CODING

Thor Magnusson
ixi audio &
Faculty of Arts and Media
University of Brighton
Grand Parade, BN2 0JY, UK

ABSTRACT

This demo paper describes the rationale and design of the ixi lang, a live coding language built on top of SuperCollider. The paper explains why SuperCollider is used for this task, and reports on a survey conducted with users of the language. It concludes that simple and constrained systems can be useful in specific musical contexts, in particular when sketching or improvising, but that such systems can be limiting in the long run.

1. INTRODUCTION

In the summer of 2009, I had some free time where I decided to work on my C++ chops. When reading a book on C++ I got inspired by the section on operator overloading, which describes how any operator, method, or symbol can take on a new signification. A revisit to esolangs.org – a resource for esoteric languages, such as Whitespace and Brainfuck – prompted further intriguing ideas that resonated with the inspiring voodoo-like live coding systems of Alex McLean (yaxu.org/software). Much fun indeed, but I was not tempted by the prospect of writing my own live coding audio engine or a clock system that supports musical timing. Hence, I decided to build a language on top of SuperCollider by overloading its operators rather than C++'s, thus maintaining access to the expressive power of SuperCollider itself.

Live coding is becoming increasingly popular. A dedicated forum exists for practitioners (www.toplap.org) and various papers have been written with topics that range from general introductions [2][5], to live coding in specific systems [7][8][10], or live coding as artistic practice [6][9]. Key imperatives in live coding include projecting the performers' screens to the audience and to code from a relatively blank slate. It invites the audience to engage conceptually with what the performer is doing, which can involve sound synthesis, instrument design, algorithmic agent building, composition, and performance – all at the same time.

Live coding is used equally in music and graphics, and has inspired other practices such as live hardware hacking (as practiced by the New York based The Loud Objects) and live composing [4].[1] It introduces improvisational methods to computer based art practices. However, as intriguing as live coding performances can be at times (and we have indeed seen boxing challenges, choreography, group composition, audience interaction, artificial agents and more), they can also be immensely boring and dismissing. There is little fun in watching a stressed programmer designing algorithms for minutes before a simple sine oscillator is applied in the playback of a silly melody.

The ixi lang was intended to address this situation and possibly prevent such boredom. The goal was to be able to create a tune with rhythm and melody within a few seconds from the performance starting. The language should also be understandable to non-programmers who would be able to follow clearly the performer's train of thought. It was implemented in SuperCollider, enabling the use of all its available synth definitions, audio busses, sample buffers, and language constructs. In this paper I will introduce the design rationale of the ixi lang, its functionality, and report on a user survey conducted with its users.

2. DESIGN RATIONALE

A typical problem for the live coder is the high level of expertise required for such performance [6]. Very few performers are able to exhibit those skills without consistent dedication to practice [9]. Although I have long been fascinated by certain virtuosic live coders, it seemed to me that such incorporation of dexterity strives against the primary rationale of the mechanical computer; namely the automation of rote tasks and the augmentation of mental capacity. From this perspective, I attempted to design a musical live coding language that would free performers from having to think at the level of computer science, allowing them to engage directly with music through high-level representation of musical patterns. Most importantly, the language should be easily understandable by the audience who would be able to follow each step of the performance, given a little bit of imagination in terms of interpreting language features and functions.

The ixi lang is an interpreter built in SuperCollider, thus concomitantly gaining access to the underlying power of that environment. This is achieved by using a different key combination to evaluate the ixi lang code and sc lang code; thereby splitting the SuperCollider document into a development environment that supports two languages. However, unlike SuperCollider, the aim with the language was to provide a highly simple syntax with strong expressive constraints. As such, the system itself would become a compositional form. Here, constraints inherent in the language are seen as providing freedom from complexity, yet defining a large

[1] This was successfully explored by BIT20 Ensemble and The Bays in the Integra 2008 festival, Birmingham.

enough search space [1] for musicians other than the language author to explore and express themselves.

3. THE IXI LANG FUNCTIONALITY

The ixi lang has three modes of musical notation that can be generated and synchronised in real-time: melodic, percussive and concrète (sample based). These musical patterns are created in the form of identifiable agents whose performance can be adjusted through various methods (e.g., shifting notes, transposition, reversing, inversing, scrambling). Figure 1 shows a text document that serves as the code input window. A scale and a tuning can be chosen at the start. In this screenshot we see how an agent called "jarret" is created and assigned a piano instrument that plays the 7th, 1st, 5th and 3rd notes of the minor scale. The spaces represent silence. The next agent "jar2" gets a score that is played one octave lower and waits for 16 notes before playing again (-12!16). The square brackets represent the melodic mode where any synth definition in SuperCollider can be used as a source (replacing the "piano" synth definition). Below, "ali" and "hat" are in percussive mode (represented by the "|" sign). Here each character signifies a specific sound, either synthesized or sampled, that has been mapped to it.

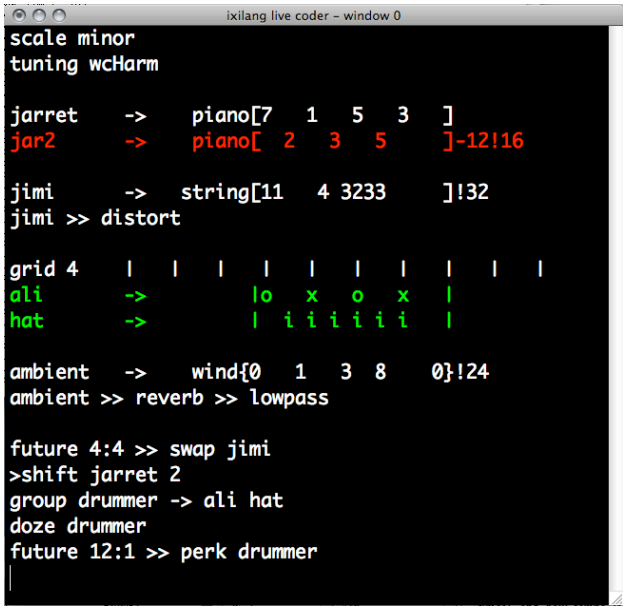

Figure 1. A screenshot of an ixi lang session

The output of the musical agents can also be routed to multiple effects and filters, such as "jimi" being routed through a distortion effect. We also see how actions are assigned to an agent in the future ("jimi" swaps its items every 4 seconds, four times). These methods on the agents are called "actions" and have the structure of verb-noun-adjective (as in >shift jarret 2, which stands for the operation of shifting the items in agent jarret 2 slots to the right). Initially the language had an object-method-argument (jarret.shift> 4) design structure, but was changed to the above in order to made the language more understandable to the layperson. Importantly, any change that is made onto an agent's score through code is updated in the text document. The code serves therefore both as an updated representation of the score and an instruction to the system's play mechanism.

The ixi lang clearly affords a certain limited set of musical activities. It provides a scaffold for externalising musical thinking and through its simplicity attempts to ease the live coder's cognitive load. As a live coding system it goes further than most common live coding environments in providing a simple, high-level platform for musical improvisation. This is at the cost of expression, as height obviously impedes freedom. However, as the system is written in SuperCollider, normal SC code can easily be written in the same document, thus tapping into the extensive scope of SuperCollider itself. A synth definition can be written in a performance, added to the server and immediately used in the ixi lang. For example the following synth definition:

```
SynthDef(\ixilangdemo, {arg out=0, freq=440;
        var signal, env;
        env = EnvGen.ar(Env.perc(0.001, 0.1), doneAction:2);
        signal = SinOsc.ar(freq, 0, env);
        Out.ar(out, signal!2);
}).add
```

could be used in an ixi lang session like this, where the agent "loki" is given a score playing the ixilangdemo instrument:

```
loki -> ixilangdemo[1  3 2 1  4  ]+12
```

Finally, all samples that are placed in the sounds folder of the ixi lang are constructed as a synth definition. A sound called "bird.aif" will become a synth definition called bird. This synth definition can then be played as bird[1 2 3 4] in the melodic mode, where the sound is pitch-shifted according to the scale values; in the percussive mode as |b b b b | (provided that the keymapping file has mapped "bird" to the 'b' character); and finally in the concrète mode we get bird{0 1 2 4 9 }, where the numbers signify the amplitude of the sound. The concrète mode was added latest, as I became tired of the strongly timed structure of ixi lang performances. I missed the texture available from sound file playback and this mode allows for sufficient control over recorded sounds.

4. USER FEEDBACK

Having developed the language, I found that I could use it for some of musical work and not other. For me, the ixi lang is enjoyable, but I am not fully satisfied by it. The temporal focus poses strong limitations and I tend to think music differently. This discontent prompted me to ask users to sign up for user evaluation survey when I released the language on the ixi audio website. When the survey was run there were 237 registered users (now 593), of which 23 responded to the questionnaire (a 10% participation is common in other surveys we have conducted). The survey lives here:

http://www.ixi-audio.net/ixilang

Although the survey is mainly qualitative, it includes a common user interface satisfaction questionnaire, as proposed by Chin et al. [3] where a Likert scale from 1 to 7 is used. The results can be seen in Table 1. It is clear that whilst users found ixi lang somewhat limiting, rigid and at times difficult, they also found it stimulating, wonderful and easy to understand:

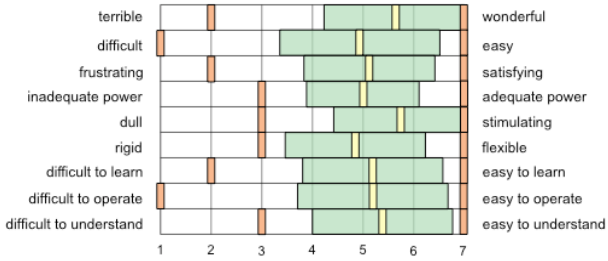

Table 1. User Interface Satisfaction. The yellow line is the mean, the green block represents the standard deviation, and the orange lines are the lowest and highest answers.

The survey was largely qualitative, prompting users to respond with discursive answers that describe overall experience, programming knowledge, and musical background. I was interested in learning about how users perceive the environment in the initial encounter, and here some responded:

Ixi lang is an application that helped me a lot in the development of rhythmic patterns in a way that is very simple and convenient, it is a useful tool that has much to be exploited.

The sheer speed and ease of setting up and reorganizing sequences leads to a very direct connection with the music, not at all like drawing MIDI notes on a DAW. It feels much more like an instrument.

The audience can immediately participate in the performance. The language is general and simple. At times funny to watch.

The obvious constraints as compared to, say, Ableton or Reason are quite welcome - a release from the paralysis of choice! Still, I would love to be able to become more proficient with SC so as to tailor the environment to my needs.

It encourages quick sketches in exploring polyrhythms and counterpoint, which is something that can be tedious in other digital environments. It's easy enough to develop semi-aleatoric arrangements as well, which is always fun. The sense of immediacy one feels using ixi lang can be a great catalyst for a new project when inspiration or motivation are in short supply.

Questions were asked about whether users felt constrained by using ixi lang, inspired, and/or prompting new ways of thinking.

Constriction is more important than freedom for art.

Very excited at first – the limitations became creative and the experience was immediate and very satisfying. However, I soon became frustrates with the limitations as well!

It feels like you're modifying someone's else software.

Wonderful to break free from the rigid time line approach.

It draws on the sequencer metaphor, but isn't constrained by set step sizes (typically 16 steps) of normal sequencers. The future function allows one to program changes to the scores and this gives ixi lang its programming, real-time character.

I like the idea of thinking of phrases as objects that can be manipulated by their name.

Users were asked if they had used ixiQuarks and how the ixi lang compares to that software. It was clear that users found the ixiQuarks more direct and embodied, even if primarily screen-based. Some noted the directness of engaging with sounds, such as in SoundScratcher, or the convenience in representing sounds as objects on a two dimensional plane that can be operated on.

ixiQuarks allow to imagine more "acousmatic" ideas, but the signal mapping need to be very concentrated. The rhythm conception of ixi lang gives other freedom (no worries about buses, order synths...) but the "concret" part of ixi lang is not so developed to imagine the "timeless" power of the "acousmatic sounds". But, this doesn't have to be necessarily the same program... Is two different ways to think.

Also a question was asked of what people would like to change in the language.

The only problem that I found was that although I realize that this software gives the user a "live coding" sessions, I felt it was too...live? Occasionally, I wished that there was some simple GUI button to silence certain loops and bring back on. Although everything would be up on the board for me to read, I was start to loose track of what was happening where.

There is no control over note duration. This is an unnecessary limitation.

In the first version of ixi lang, representation of note duration is missing and some users complain about this. This omission is partly deliberate as I was interested to see if people would note this strong limitation in the software. Indeed, only a couple of users did mention it, almost as if people take the tool on its own premise and start to think in terms of the tool, so much that one of the most basic musical parameters, such as the note length, is not thought about.

Although most participants stated that they enjoyed operating within the limitations of ixi lang, with comments such as, "Constriction is more important than freedom for art," this view was not shared entirely with the actual SuperCollider users that participated in the survey. Those who experimented with the language found it amusing and impressive, but used it only a few

times. They expressed that they would rather design their own systems and that the ixi lang does not particularly suit their way of thinking. This was expected; SuperCollider users are famous for wanting to design their own systems and not wanting to subscribe to other people's ways of doing things.

5. THE CODER AND THE MATRIX

In additions to improvements and a wider array of functionality in ixi lang, two important modes have been added since the above survey was conducted. The "coder" function opens a new coding window where each key stroke actually results in sounds, using the mapping between keys and samples, as discussed above in the section on the percussive mode. This presents a new mode in live coding where the actual keystrokes result in sound, removing the characteristic latency between having an idea of a new musical function and evaluating it. The keystrokes can be recorded and turned into agents with scores.

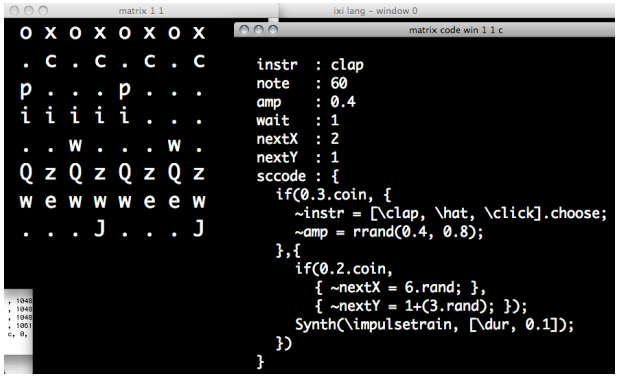

Figure 2. The ixi lang matrix. Each of the cells contain information like that displayed on the right. One can see the speed and the direction, but also the SuperCollider code which drastically strengthens the expressivity of this coding system.

The matrix is a system that mixes the declarative style of ixi lang and the procedural style of SuperCollider. Here agents run through a matrix where each cell is effectively a vector with specified speed and direction. However, each cell can contain SuperCollider code, thus accessing all the algorithmic power of that language, which was missing in ixi lang.

6. CONCLUSION

The ixi lang was devised to address specific problems common in live coding performance, such as slow and laborious build-up, incomprehensibility, and difficulty in making simple musical structures. It provides the performer with a very high-level language where musical structures can be set up in a matter of seconds using a syntax that is intuitive and easily understandable to audience. As such, I think the project has succeeded in fulfilling the original aims.

As an extension (or parasite) of SuperCollider, I hope that the ixi lang will contribute to the promotion of this programming environment. It integrates nicely with SuperCollider by using a different interpreter key combination, thus allowing for sc lang and ixi lang code to be run in the same document. Any pattern compatible synth definition in SuperCollider can be used and the language can be easily extended by users.

The ixi lang provides a syntax that affords certain musical compositions. It encourages users to explore tunings, rhythms, melodic and harmonic structures in a new representational mode. Users report on finding it a useful tool for exploration of certain musical structures and user friendly instrument for performance. However, the language is highly constrained, with limitations including lack of modularity and user customisation.

Aesthetically, I found the ixi lang basis in the use of patterns rather one-dimensional. This is an encouragement for me to explore further systems that support a more embodied musical composition, both hardware and screen-based, but also investigate patterns that afford flexibility in timing, e.g., swing.

7. REFERENCES

[1] Boden, M. A. *The Creative Mind: Myths and Mechanisms*. Wiedenfield and Nicholson, London, 1990.

[2] Brown, A. R. & Sorensen, A. "Interacting with Generative Music through Live Coding", *Contemporary Music Review*. Vol. 28 (1), 2009, pp. 17–29.

[3] Chin, J. P; Diehl, V. A. & Norman, K. L. "Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface", *ACM CHI 1988 Proceedings*, 1988.

[4] Clay A. & Freeman, J. (eds.) *Contemporary Music Review*. Vol. 29 (1). 2010.

[5] Collins, N.; McLean, A.; Rohrhuber, J. & Ward, A. "Live Coding Techniques for Laptop Performance", *Organised Sound* vol. 8 (3), 2003. pp. 321–30.

[6] Nilson, C. "Live Coding Practice", *Proceedings of NIME,* New York, 2007.

[7] Rohrhuber, J; de Campo, A. & Wieser, R. "Algorithms Today: Notes On Language Design for Just In Time Programming", *The ICMC 2005 Proceedings*, 2005.

[8] Sorensen, A. "Impromptu: An interactive programming environment for composition and performance", *ACMC 2005 Proceedings*, 2005.

[9] Sorensen, A & Brown, A. "aa-cell in Practice: An Approach to Musical Live Coding", *The ICMC Proceedings*, 2007.

[10] Wang G. & Cook P. "On-the-fly Programming: Using Code as an Expressive Musical Instrument", *The Proceedings of NIME*, 2004.