Understanding through Code Visualisation

Arrian Purcell

A subthesis submitted in partial fulfillment of the degree of
Bachelor of Software Engineering (Honours) at
The Department of Computer Science
Australian National University

July 2014

Typeset in Palatino by $\TeX$ and $\LaTeX\,2_\varepsilon$.

Except where otherwise indicated, this thesis is my own original work.

Arrian Purcell
12 July 2014

To ...

Acknowledgements

Ben/Henry

Office

Andrew

Work?

Family/Friends

...

Abstract

Live coding is a method of performance that presents audio and visual content to audiences through programming. Often s̈howing the codeïs a fundamental part of the performance in order to retain the attention of the audience and provide a measure of authenticity.

Currently missing within the research within live coding is a visualisation of the code that represents the artists intent. Previous visualisation techniques present an abstract and often disjoint representation from the associated code. Missing within this context is a formal analysis of how to best represent the artist's intent visually and a formal analysis of the target audience.

Contents

Chapter 1

Introduction

———————— Questions to ask (thesis post-mortem):

What was your thesis about? looking at code visualisations What did you find out? there was a demonstrable shift in understanding when using visualisation x... so what?

————————-

didactic vs aesthetic industry vs art

The purpose of software engineering is to bring together ideas from many fields into one - developing good software that is fit for purpose and fit for use. The systems engineering perspective of software engineering. However, one area where software engineering has had little influence and little influence has been taken from is the area of art.

Often put on separate ends of the spectrum... unless you are building software for multimedia practice - even then though...

Software engineering practice and art are often considered opposing ends of the spectrum.

This project will examine the relationship between aesthetics and educational aspects of code visualisiatons applied to two fields - an industry based application and a new media art application.

Questions: (1) What about the multimedia arts domain? What about the application of aesthetics and consideration for design within the space of software engineering? Both are widely accepted areas

(2) Why don't we even have the most simple of tools to visualise changing code structure? Too complex? Not helpful? Why must code structures be designed statically?

(3) Where is the future of programming? It is esentially the same as it was when it was first created. (Focus on higher abstractions in some fields? Move towards less abstraction...) Yet today it claims to be interactive/responsive etc...

——— Initial ideas/ ideas from the first presentation:

-code is often difficult to quickly understand -some observers may lack the experience to understand the software or the programming process

Additionally, how we program does not achieve the goals we set out to achieve (from $https : //www.youtube.com/watch?v = 1f13TTu_X9k$, taken from presentation reimagining programming languages): -programming is unobservable (looking at the system through a keyhole) -programming is indirect (no direct feedback) -programming is incidentally complex (complexity not inherent in the problem that we need to solve)

-how can we improve source code comprehension? -how can we aid understanding of the programming process? -better yet, how can we better communicate the programmers intention?

-techniques such as modelling or code documentation arent dynamic or flexible -dont allow for close to realtime understanding -an effective technique is the use of visualisations -it would be valuable to use visualisations as a means to communicate the programmers intention

1.1 Summary - remove

-this thesis will explore code visualisations -specifically, it will investigate visuals within the combination of the domains of software and music -will be using Iive coding as a platform and case study for this (will discuss later) -will develop and test code visualisations on audiences with audiences of varied levels of experience with programming, addressing code comprehension

1.2 Background

1.2.1 Live Coding

-live coding is a platform for bridging these two domain visualisations -what is live coding? -method of programming in front of an audience for artistic or informative purposes -the live coder displays their screen to an audience, showing their code as they are working on it building a functional program -makes use of interactive programming environments -program running while changes are being made -often focusses on improvisation - the programmer often has to think on their feet

-what does live coding achieve? -gives the audience insight into the programming process - ill be taking advantage of this

1.3 Theoretical Framework

1.3.1 Taxonomy

-Goal: categorising existing visualisations -Gaps in existing models: elaboration of dynamic software visualisation taxonomies, taxonomy of music visualisation

-High level features: -Shape -Size -Orientation -Dimensionality -Colour -Rethinking Visualisation: A High Level Taxonomy' discusses lower level taxonomy including -spacial relationships, numeric trends, patterns, connectivity and filtering

-distinction between scientific visualisation and information visualisation (Infovis discussed in Rethinking Visualization: A High-Level Taxonomy) -Information visualisation vs Scientific visualisation - infovis when spacial representation is chosen, scivis when spacial representation is given -Taxonomy developed within this article consists of discrete, continuous vs display attributes (eg. given, constrained, chosen) per the design model -Discussed in A Principled Taxonomy of Software Visualisation (1993) -Myers (1986) classifies using level of abstraction vs level of animation. (Visual Programming, Programming by Example A Taxonomy) Also uses static, dynamic vs code, data. Minimal discussion of dynamic visualisations; no elaboration.

-Most effective visualisation technique might be S̈elf-illustrating phenomena¨

Both code and music present a wide variety of visualisation techniques. These techniques will be summarised below.

1.3.1.1 Code Visualisation

Code Augmentation Visuals based on Code Augmentation (eg. infographics, annotations, sparklines)
Visuals based on Code Augmentation -infographics -annotations (visual code annotations for cyberphysical programming) -sparklines (Visual Monitoring of Numeric Variables Embedded in Source Code) -etc

Code Abstraction Visuals based on Abstraction (eg. scheme bricks, gource, code flow)

Visuals based on Abstraction -scheme bricks -gource -code flow -etc

Software Domain Visualisations Domain Visualisations (eg. fluid source code views, indentation, class diagrams)

Domain Visualisations -understood by the domain, not necessarily useful for observers -eg. eclipse, visual studio code displays, auto updating class diagrams etc -fluid source code views (Fluid Source Code Views for Just In-Time Comprehension) -class diagrams -debugging -tracing

1.3.1.2 Music Visualisation

-music is similar to software in a number of ways -often has standardised notation -expression may diverge from notation -visual representation is by default static -can be visualised using dynamic methods -understanding can augmented with visualisation techniques

Generative Visualisations Generative Visualisations (eg. frequency wave, VLC/iTunes visualisations)

Generative Visualisations -g̈enerates animated imagery based on a piece of music-̈eg. change with loudness and frequency spectrum -VLC, iTunes etc. -visualisations that respond to music

Associative or Emotive Visualisations Associative or Emotive Visualisations (eg. video art, sampled video)

Associative or Emotive Visualisations -includes areas such as synaesthesia (Movies from Music) -eg. video art, sampled video with sampled music -extension/exploration of Emotion-based Music Visualisation using Photos -Classifications including: sublime, sad, touching, easy, light, happy, exciting, grand

Music Domain Visualisations Domain Visualisations (eg. ableton, sheet music)

Domain Visualisations -understood by the domain, not necessarily useful for observers (A Visualisation of Music) -could include music creation tools, for example abelton etc. -graphic representations that have one-to-one mapping -direct visualisation -eg. sheet music (again, A Visualisation of Music)

1.3.2 Design

-NOTE: make it clear that this is not intended to be a visual programming language environment. The intention of this project is to assist in code comprehension

1.3.2.1 Cognitive Dimensions of Notation

$http : //en.wikipedia.org/wiki/Cognitive_dimensions$

1.3.2.2 Visual Primitives

structural units include line, shape, color, form, motion, texture, pattern, direction, orientation, scale, angle, space and proportion.

deutsch limit (max 50 visual elements onscreen)

Chapter 2

Literature Review

Research Project - Article Summaries

A principled approach to developing new languages for live coding

- Focusses on musicians entering the field of live coding
- Discuss domain specific languages (e.g. spreadsheets in accounting)
- Approaching live coding as a way to extend the musician rather than the programmer becoming a musician
- Interfacing with external hardware
- Cognitive ergonomics of language design
- Declarative constraint propagation
- Direct manipulation over indirect manipulation allows audience to perceive relationship between action and effect
- Use supercollider as the live coding platform
- Critical technical practice

Algorithms as Scores: Coding Live Music

- considers live coding as a new branch of musical score
- Kadinsky and Klee representing synchronic process (painting) as diachronic process (music)
- graphical representations of music
- graphical scores as special representation of an algorithm
- Claudia Molitors 3D Score Series - engaging with score
- Basically describes the history and modern live coding practice

An Approach to Musical Live Coding

- aa-cell performances
- does remapping a function to a random function produce measurable results
- Overview of the live coding environment and practice

Visual Music Instrument

- synsthetic composition, computational expression and the dynamics of performance are important research axes
- History of live visual performances
- painted composition is closer to a single musical instance than it is to musical composition.
- music is abstract, visuals are moving that direction too
- Visual Music Instrument Design

A Principled Taxonomy of Software Visualisation

- Discusses visualisation of algorithm
- level of abstraction vs level of animation
- aspect vs abstractness vs animation vs automation
- data vs code
- static vs animated
- No demonstrable gains from software visualisation seen
- Very old article

A Model-Based Visualisation Taxonomy

- scientific visualisation vs information visualisation
- model based visualisation taxonomy divides groups into continuous and discrete models
- continuous model divides into 3 dimensions including dependent variables, data type, and number of independent variables
- discrete model divides into connected and unconnected data types

A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualisation

- shows taxonomy of glyph placement strategies

- not immediately relevant

A Taxonomy of Program Visualisation Systems

- Scope (Code, Data state, Control state, Behaviour)
- Abstraction Level (Direct representation, Structural representation, synthesised representation)
- Specification method (Predefinition, Annotation, Declaration, Manipulation)
- Interface (Simple objects, Composite objects, visual events, dimensionality, multiple worlds, control interaction, image interaction)
- Presentation (Analytical, Explanatory, Orchestration)

Improvising Synesthesia

- introduction of the term comprovisation
- has been no visual creative process in which the artistic process is available to the audience
- improvisation of visual art

Live Coding Towards Computational Creativity

- describes what live coding is and potential future directions in terms of computational creativity
- includes live coder survey (http://doc.gold.ac.uk/ma503am/writing/icccx/

Painterly Interface for Audiovisual Performance

- Describes the history of audio visual performance (incl. Castels Ocular Harpsichord, Thomas Wilfreds Clavilux, Oskar Fischingers Lumigraph, Charles Dockums MobilColor Projector,)

AVVX - A Vector Graphics Tool for Audiovisual Performances

- Survey for ease of use and utility of the AVVX engine

Content-based Mood Classification for Photos and Music

- Variety of emotion classifications:
  - Thayers model: stress vs energy
  - Russells model: pleasantness vs alertness
  - Tellegen-Watson-Clark model: positive affect vs negative affect
  - Reisenzeins model: pleasantness vs alertness
- Classified images and music as: aggressive, euphoric, melancholic, calm

- Found that a combination of dimensional models and category-based models provided the most useful results

Dimensions in Program Synthesis

- Three dimensions in program synthesis: User intent, search space (expressiveness vs efficiency), search technique (eg. brute-force)

Dimensions of Software Architecture for Program Understanding

- Three dimensions of software architecture that affect user involvement: level of abstraction, degree of domain specific knowledge, degree of automation

Gathering Audience Feedback on an Audiovisual Performance

- Modes of engagement: Perceptive, interpretive and reflective

The Programming Language as a Musical Instrument

- Discusses differences between software engineering and live coding as a musical practice
- Utilitarian design focus helps live coders see beyond the narrow focus of live coding performance itself and see the underlying software engineering focus including requirements analysis, design, reuse, debugging, maintenance etc.

Rethinking Visualization: A High-Level Taxonomy

- Old method of categorisation: Scientific vs Information (incl. factors such as scientific vs non
- scientific, physical vs abstract, spacialisation given vs specialisation chosen)
- Introduce ḿodel basedv́isualisation techniques
- our main objective is to provide insight into how different research areas relate, not to provide guidelines for visualisation design."
- Terminology - Object of study, Data, Design model, User model (see image in article for relationship)
- Taxonomy developed within this article consists of discrete, continuous vs display attributes (eg. given, constrained, chosen) per the design model
- Continuous model visualisation is broken down according to the number if independent and dependent variables and the type of the dependent variables (incl. scale, vector and tensor)
- Discrete model visualisations are broken down into wether the data structure or data values are visualised
- Article discusses lower level taxonomy including - spacial relationships, numeric trends, patterns, connectivity, filtering

Heuristics for Information Visualization Evaluation

- suggests that between 3 and 5 heuristics for evaluation would be enough

2.1 Outline

Historic Programming Approach (history of coding - eg. communication of an idea into code) Programming for the Modern World (discussion of cyberphysical programming, live coding, collaboration etc...) Historic Visualisation Approach (history of visualisations) Visualisations as Communication (discussion of code visualisations)

2.2 Live Coding

(focus on developing a narrative concerning what needs to be done within live coding to achieve the software engineering goals and what needs to be done to develop successful visualisations within the field of live coding)

Live coding describes the process of exposing the programming process to a live audience. -talk more about what live coding is

Live coding history... . -talk more about history of live coding

There exists much discussion within the live coding research body (eg. ...) about the potential for live visual manipulation and examination of the current progress within the field to achieve this.

2.2.1 Music vs Visualisation

In addition, there has been a move towards manipulation of the visuals in synchronisation with the ....

2.3 Music Visualisation

2.3.1 Taxonomy

2.3.2 Live Performance

2.4 Software Engineering Practice

As the field of live coding develops, the relevance of both the application of software engineering practice to the field and the relevance of live coding to the field of software engineering has become highly apparent...

2.4.1 Application of Software Engineering to Live Coding

The application of software engineering to live coding... ([Blackwell and Collins 2005] paper incl. requirements analysis, design, coding, project management, reuse, debugging, documentation, comprehension and maintenance)

2.4.1.1 Design

Design...

2.4.1.2 Coding

Coding...

2.4.1.3 Comprehension

Comprehension...

2.4.2 Application of Live Coding to Software Engineering

The application of live coding to software engineering...

2.4.2.1 Dissemination of Code Understanding

2.4.2.2 Multidisciplinary Cohesion

2.4.2.3 Visualisation Framework

–– The Early History of Software Visualization (Baecker, Ronald chapter of book) discusses the purpose of software visualisation as enhancing program representation, presentation and appearance. Discusses further that programmers have always used visualisation tools (eg. diagrams) to aid in illustrating program function, structure and process. States further that any of typography, symbols, images, diagrams and animation can present information more concisely than formal/natural programming languages.

The Early History of Software Visualisation (Baecker) gives the major ́threadśof activity in the development of software visualsation as (1) the presentation of source code, (2) representation of data structures, (3) animation of program behaviour and (4) systems for software visualisation and (5) animation of concurrency. ———

Exposing the Programming Process discusses the focus of visualsations on the execution of the program rather than the development of the program.

Found demonstrating the following concepts useful for teaching programming: - use of an IDE - incremental development - testing - refactoring - error handling - use of online documentation - model-based programming

——— Seven Basic Principles of Software Engineering (Boehm, 1983) discusses the following principles: (1) manage using a phased life-cycle plan (2) perform continuous validation (3) maintain disciplined product control (4) use modern programming practices (5) maintain clear accountability for results (6) use better and fewer people (7) maintain a commitement to improve the process

—————

-"A person understands a program because he is able to relate the structures of the program and its environment to his conceptual knowledge about the world." - Program Understanding and the Concept Assignment Problem, Biggerstaff et al

-"A person understands a program when he is able to explain the program, its structure, its behavior, its effects on its operational context, and its relationships to its application domain in terms that are qualitatively different from the tokens used to construct the source code of the program." - Program Understanding...

———–

Toward a Perceptual Science of Multidimensional Data Visualization: Bertin and Beyond discusses that info visualisations can be constructed by mapping one data dimension to brightness, colors or orientations, however, it goes on to say that in practice this does not work very well. Ḧumans can only perceive a limited number of data dimensions.̈

Discusses fundamental graphic primitives/procedures as invariants, components, correspondences and marks. Components define variational concepts. Invariants relate components. Marks (three types/ ïmplantationsïncl. point, line and area) are graphical representations on the visualisation that represent invariants. Correspondences are the relationships between and among components.

Discusses that there are two F̈unctionally different classes of visual variableïncluding planar and retinal variables. Planar includes any spatial information and retinal variables include things such as size, color, shape, orientation, texture and brightness (which can use marks to be displayed).

(Check source) Bertin states that it is only possible to effectively map three graphic variables including two planar dimensions and one retinal dimension.

Bertin also did not discuss motion within his variables. Investigation of this in the context of temporal visualisation would be enlightening.

——— Cleveland and Mcgill accuracy vs genericity. Hierarchy of effective visualisation elements in terms of the accuracy they provide or how they allow generalisation.

——— Model: Quadrant model... Didactic vs Aesthetic (Intent) and Generic vs Specific (Coverage)

Chapter 3

Survey

3.1 Purpose

The purpose of this interview was to gain insight into the audiences current understanding and enjoyment of the live coding process. Additionally, the relationship between enjoyment and understanding was to be examined. It was hoped that the examination of these factors would further inform the development of visualisations within live coding.

3.2 Hypothesis

Describe what you think will happen.

3.3 Materials

-Computer with Extempore -Live coding performance venue

List special materials you used.

3.4 Method

Survey questions were distributed following a live coding performance. Both an online and paper copy were distributed.

Write a step by step description of what you actually did, identifying the different variables and how you controlled them. Describe what things you changed (variables you manipulated).

3.5 Observations

Overall, the live coding performance went smoothly. The room layout was perhaps not optimal for the performance and the display screen was very dim leaving some parts of the source code unreadable. Additionally, the projection surface was not flat further reducing readability.

An estimated 20 people were in attendance at the start of the performance. This grew to an estimated 30 people by the end of the performance, over a time period of about 20 minutes.

The logistics of handing out paper surveys did not suit the venue layout, however most people had the ability to access the survey through their phone. Online survey attrition was about four people, though the accuracy and reason behind this can not be determined from the data. As suggested by Henry, it may be easier just to hand out the QR code on a piece of paper after the performance.

Henry also noted that the other performer used his visuals to tell a story and this story may or may not relate to the music being played.

3.6 Results

A total of thirteen survey responses were received. Of these, 77% regularly listen to music and 54% perform regularly. 38% of the respondents have high exposure to programming through work, study or their hobbies, as opposed to 31% who have no experience with it. Of the respondents, 69% had never been to a live coding performance before.

Enjoyment was measured according to the relative change in enjoyment through the performance from the beginning to the end. 46% of survey respondents had high enjoyment throughout the performance. The results for enjoyment are summarised in Table 1. The results suggest an overall high level of enjoyment of the performance. No respondents chose low enjoyment throughout the performance.

Dimension	Flat	High	Low	High to Low	Low to High	Unsure
Count	2	6	0	2	1	2

Table 1 : Enjoyment through the performance

Similarly, understanding was measured according to the relative change in understanding through the performance from the beginning to the end. 31% of survey respondents had no change to understanding through the performance The results for understanding are summarised in Table 2. Overall, understanding is spread out more than enjoyment with only 15% suggesting that they could understand the relationship between the visuals and the music throughout the performance. There is no statistically significant relationship ($p > .05$) between music listening habits and understanding nor is there a statistically significant relationship ($p > .05$) between coding experience and understanding.

Dimension	Flat	High	Low	High to Low	Low to High	Unsure
Count	4 2	0	2	3	2	

Table 2 : Understanding through the performance

The relationship between enjoyment and understanding can be seen in Figure 1. Notably, three respondents who had high enjoyment throughout the performance were the only respondents who had a pattern of low to high understanding. However, the relationship between enjoyment and understanding is not statistically significant

($p > .05$).

69% of respondents stated that the visuals provided a sense of liveness to the performance. The remained 31% stated that they had no effect on their sense of liveness. There were no responses stating that the visuals negatively impacted the sense of liveness.

In terms of confusion, 38% suggested that no aspects of the visuals were confusing, though 31% did not respond to the question.

Supplementary observations of the performance are available in Appendix B.

1.Using all your senses, collect measurable, quantitative raw data and describe what you observed in written form. 2. Reorganise raw data into tables and graphs if you can. 3. Don't forget to describe what these charts or graphs tell us! 4. Pictures, drawings, or even movies of what you observed would help people understand what you observed.

3.7 Discussion

1.Based on your observations, what do you think you have learned? In other words, make inferences based on your observations. 2.Compare actual results to your hypothesis and describe why there may have been differences. 3.Identify possible sources of errors or problems in the design of the experiment and try to suggest changes that might be made next time this experiment is done. 4.What have experts learned about this topic? (Refer to books or magazines.)

Chapter 4

Follow-Up Interview

4.1 Purpose

The purpose of the follow-up interviews was to gain an in-depth view of what some of the audience believed they would like to understand more about the performance.

Additionally, it was anticipated that some audiences may prefer not to understand more about the technical details of the performance but would be more interested in focussing on the music throughout the performance. The extent of this sentiment was to be examined.

4.2 Method

Two questions were asked of three audience members a number of days after a performance. These two questions were: What did you únderstandábout what was going on with the code being projected? In particular, what did you understand about the relationship between the code and the music? Would would you like to understand more about the code in order to enjoy the performance more?

4.3 Results and Discussion

A total of three responses were received. interviews are available in Appendix A.

All three respondents referred to the initial stage of developing loops to be played and two of these referred to the silence preceding the music suggesting an understanding within the initial stages of the performance. Two interviewees mentioned that they recognised changes to the code and changes to the music separately but could not see the link between the two.

All three interviewees suggested that it would be nice to know a little about the code or the concepts but it is not essential and may even be detrimental to enjoying the music. One interviewee mentioned switching between observing the code and listening to the music. This transition was described as świtching offśuggesting that the code takes mental effort to understand whereas the music does not.

Chapter 5

Visualisation Experiment 1

5.1 Rationale

5.2 Procedure

5.3 Results

Chapter 6

Visualisation Experiment 2

6.1 Rationale

6.2 Procedure

6.3 Results

Chapter 7

Conclusion

Future work: type of language may affect understanding. Imperitive vs functional etc.

Appendix A

Survey Results

Appendix B

Follow-Up Interviews

B.1 Respondent 1

1) What did you ünderstandäbout what was going on with the code being projected? In particular, what did you understand about the relationship between the code and the music?

The code sets up a set of nested loops which are then modified by the composer in real time. This immediately leads to the danger of repetitive loops. It may be useful to have rhythms of 4 or 8 bar repetition as in Africa. Actually, this musical form lends itself to that type of rhythm and music. As soon as an organ comes in I am reminded of Mike Oldfield and am anxious that someone will say slightly distorted guitar. IN jazz one improvises on standards so there is a very strong form (AABA etc) which, in general, is respected allowing the audience to deduce where they are in the piece. I had the feeling that the present way the code is used limited the music

2) Would would you like to understand more about the code in order to enjoy the performance more?

Yes, I think that if the audience were told what was happening or the ideas behind the constructs then I would be happier. Compared to jazz it is not note by note improvisation so an explanation of the limits and advantages would be useful. Respondent 2

1) What did you ünderstandäbout what was going on with the code being projected? In particular, what did you understand about the relationship between the code and the music?

In the beginning, I could tell from the silence and the live coding that it was being build, and sound by sound line by line was being added to as the piece grew. When Ben went back in the code to change beats or melodies, I could tell something was being changed but wasn't tracking what or how.

2) Would would you like to understand more about the code in order to enjoy the performance more?

I feel like I already understood a rudimentary amount [...] which was enough to enjoy it. I feel like if I had more knowledge about code I would focus on that to the detriment of the music; and if I knew more about music then I may have focused on that to the detriment of my attention on the code. Considering my education, if I had not had the exposure to code and music through [...], then some basic rudimentary knowledge of code would have been good. Respondent 3

1) What did you ünderstandäbout what was going on with the code being projected? In particular, what did you understand about the relationship between the code and the music?

I understood that the music was being made from scratch and this was evident in the long silence before any sound is heard. I understand what sounds are being made based on the code names and that some of the numbers represent timing, volume and pitch. I still don't quite understand when the code is r̈eadyänd starts working to make music. It has something to do with the highlighting the text, but that also confuses me. (I understand that most of the music is stored in the program as s̈ound bitesöf real instruments, but sounds can also be made from scratch as mathematical wave functions - but I don't think I would know this if Ben hadn't have told me). Sometimes the coder scrolls up and down the screen to much and I get lost, I don't have a big picture of what all the code looks like.

2) Would you like to understand more about the code in order to enjoy the performance more?

In some ways I would like to understand a little more about the code. It would be nice to have a director's commentary of what's going on behind the scenes, just so I can follow along with the changes that I can hear in the music as they are occurring. But I think I more enjoy just listening to the music, knowing broadly that a livecoder is manipulating code to make the sounds that I hear. I don't often like reading the code for the whole performance, maybe for a few minutes at a time, but then I like to switch off and just focus on what the musician is playing. I more often like to listen to the music and guess what the livecoder has done to make that changes (which is kind of backwards). I wouldn't mind having more understanding of the code on hand, but I probably wouldn't use the details of it during the whole performance every time.

Appendix C

Visualisations

Bibliography

BLACKWELL, A. AND COLLINS, N. 2005. The programming language as a musical instrument. . . . *of PPIG05 (Psychology of Programming . . .* , 1–11. (p. 9)