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 2ε.

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

Arrian Purcell
30 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

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 History

For most of its history source code has been displayed as simple text due to the expressiveness of this format and despite its inefficiencies. It is only recently, due to ever increasing programming language complexity and increasing computational power, that code annotations and syntax highlighting have become more commonplace. Nevertheless, these visual enhancements rarely provide information beyond the basic grammar of the language they are intended to augment. The limitations of this approach are becoming ever more apparent as programming languages and interactive programming environments move towards the need for real-time comprehension and a need to understand the source code within the context of a running program.

The problem of understanding code within a running environment is complex. The fact that programming concepts are fundamentally different to human understandable concepts [**?**] presents challenges in communicating the complex nature and intention of the program. The fundamental complexity lies within the translation from code to the brain; the programming concepts to the human understandable concepts.

The human brain is highly proficient in pattern recognition [**?**] and there is evidence to suggest that visualisations can take advantage of this proficiency to enhance understanding [**?**]. It would be informative to investigate this translation using visual representations to translate and assist in the understanding of program concepts.

1.2 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

1.3 Background

1.3.1 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

1.4 Theoretical Framework

1.4.1 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.

1.4.1.1 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

1.4.1.2 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 -ğenerates 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)

1.4.2 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

1.4.2.1 Cognitive Dimensions of Notation

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

1.4.2.2 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)

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 ́threadsóf 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. ̈Humans 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 Ẅunctionally 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)

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.)

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

Two code visualisations have been analysed in a live coding context to determine effective presentational and educational features. Two visualisations were evaluated including a visualisation targeting aesthetic appeal and a visualisation with a more didactic approach. The goal was to determine the usability, differences and desirability of the two approaches to further inform future live coding visualisations.

The set of didactic visualisations predominantly focussed on the relationship between the live coding active processes and their behaviour. The visualisations prominently displaying the names of the active functions with visual indication of the number of functions running and their callback time. Bright colours and solid shapes were used to ensure constant visibility and communicate the intention of the underlying code. Overall, four visualisations were presented with each introduced depending on the number of active functions. It was predicted that taking a more educational approach would see a reduction in audience confusion through the performance.

The set of aesthetic visualisations focussed less on the programmatic aspects of the live coding performance, rather intending to provide additional visual interest to the projected code thereby prolonging attention. More variety was used in visual structure and colour. Again, four visualisations were presented, varying the visualisation based on the number of active functions. It was predicted that focussing on the aesthetic nature of the visualisations would assist in audience retention and result in a consistency of interest through the performance.

5.1 Method

Two sets of visualisations were presented to an audience during a live coding performance and surveys were distributed. The performance was run as two ten minute improvised songs with each song demonstrating one of the two experimental visualisation conditions, either the didactic condition or the aesthetic condition. The experiment was run twice, with two separate participant groups, swapping the order in which the two sets of visualisations were presented to the audience.

The audience members completed a survey (see Appendix B) consisting of four sections over the course of the performance. These sections included a demographic information section, opinion of the first visualisation, opinion of the second visualisation and a section investigating the audience's overall opinion of the performance. The sections regarding the visualisations predominantly focussed on the enjoyment and understanding related to the visualisation while the final

section focussed on eliciting potential improvements.

After explaining to the participants the structure of the experiment and allowing the participants to complete the initial demographic section of a survey, the live coder began the first performance utilising one of the two visualisations. Following this first set, the second section of the survey was conducted. A second set was then played using the alternative visualisation. Following this second set, the same survey questions were administered again, again asking the participants specifics about their enjoyment and understanding related to the specific visualisation demonstrated. A final survey question was then asked relating to their opinion regarding the whole performance and their suggested improvements.

To further inform the outcomes of the experiment a video-cued recall interview was conducted following the performance with the live coder. Two of the performances were examined critically and further discussion concerning the advantages and limitations of the visualisations were examined.

5.2 Participants

A total of 41 participants took part in the study. Over the two performances, 19 participants observed the first performance and 22 participants observed the second performance. The demographic makeup of the audiences was similar.

66% of the participants stated that they were male (see Figure C.1b) and most participants were aged between 18 and 32 (76%, see Figure C.1a). As the study was conducted within the Computer Science Department, a large proportion of the participants were experienced with programming with 90% having current or previous experience with it (see Figure C.2a). Nevertheless, only 15% of participants had previous experience with any of the Lisp style of languages (see Figure C.2b), the style used within the performance.

Of the participants, 68% stated that they listened to a large amount of music (see Figure C.3a) though only about 15% of participants stated that they played an instrument or sung regularly (see Figure C.3b). Only 22% of participants had seen a live coding performance before.

5.3 Results

Understanding and enjoyment were evaluated against the didactic and aesthetic visualisations. Results and statistical analysis of the differences between the aesthetic and didactic conditions follows. Note that for the following statistical analysis a significance level of 0.05 was used with the chi-squared test for independence.

5.3.1 Understanding

Overall, 37% participants stated specifically that the didactic visualisations helped them to understand the code, whereas 12% participants stated that the aesthetic visualisations assisted in

understanding the code.

H_0: There is no difference between the aesthetic visualisations and didactic visualisations in terms of understanding.
H_1: There is a difference between the two visualisations in terms of understanding.

A significant difference between the visualisations effect on understanding was found ($\chi^2 = 7.1986, df = 2, p = 0.02734$).

5.3.2 Enjoyment

Overall, for both visualisations, a large proportion ($> 50\%$) of the participants stated that the visualisations helped their enjoyment of the performance. Of the participants, 76% stated that the aesthetic visualisations helped their enjoyment compared to 56% participants that stated the didactic visualisations helped their enjoyment.

H_0: There is no difference between the aesthetic visualisations and didactic visualisations in terms of enjoyment.
H_1: There is a difference between the two visualisations in terms of enjoyment.

No significant difference between the two visualisations effect on enjoyment was found ($\chi^2 = 3.7733, df = 2, p = 0.1516$).

5.3.3 Change in Understanding

Participants were asked to rate their understanding during the beginning, middle and end of the performance.

During the didactic and aesthetic performances, 49% and 44% of the participants respectively stated that their understanding remained the same throughout the performance.

During the didactic performance, 10% of the audience had an understanding that tended downwards (eg. high to low) compared to 20% of the audience during the aesthetic performance meaning fewer audience members had a reduction in understanding through the didactic performance.

5.3.4 Change in Enjoyment

Participants were asked to rate their enjoyment during the beginning middle and end of the performance.

During the didactic performance, 15% of the audience stated that enjoyment increased during the beginning remaining steady for the remainder whereas 24% of the audience said the same during the aesthetic performance.

During the didactic performance, 22% of the audience stated that their enjoyment decreased during the start of the performance whereas only 2% stated the same for the aesthetic performance.

Approximately 30% of the audience during both the aesthetic and didactic performances stated that their enjoyment remained steady throughout.

See Figure **??** and Figure **??** for the distribution of changes to enjoyment between the aesthetic and didactic visualisations.

5.3.5 Liveness

Participants were asked to discuss how each visualisation influenced or impacted the liveness of the performance. Concepts identified included positive and negative valence towards the didactic visualisations, positive and negative valence towards the aesthetic visualisations and the relevance of visual source code.
within the discussion included the positive and negative aspects of the didactic visualisations, the positive and negative aspects of the aesthetic visualisations, source code discussion and statements indicating an understanding between the visuals and the source code.

Overall, 54% of the audience indicated negative valence towards the didactic visualisation regarding liveness referencing a variety of reasons including that the "visuals only responded to what was typed", that the "musical forms didn't occur at the most expected times" and that the visualisations "perhaps made the performance seem too polished".

On the other hand, 34% of the audience indicated positive valence towards the didactic visualisations suggesting that "it was easier to follow this visualisation than the code", "the visualisations clearly showed the changes being made to the code" and that these visualisations "helped more with communicating that the performance was live".

40% of the audience had negative valence towards the aesthetic visualisation its effect on the sense of liveness. Reasons cited include that the "influence is not clear" between the code and the visuals, that "the visualisation did not make much sense" and that the "visualisations did nothing to suggest that the performance was live".

32% had positive valence towards the aesthetic visualisation. A variety of the response included that these visuals were "less dominating and more complementary" and that "the visualisations helped to show when a piece of code started working".

A relatively large proportion (28%) of the audience discussed the importance of the source code to the sense of liveness of the performance such as that the "code showed what the musician was doing physically". A further 5% of the audience demonstrated a deeper understanding of the live coding process such as "changing values produced changes in tone and speed of the music pitch".

5.3.6 Improvements

The audience was asked to indicate possible improvements to the visualisations. 40% of the audience suggested improvements that indicate a desire to see a better relationship between the visualisations and the music, with suggestions such as matching the visualisations to musical rhythms or pitch. For example, one audience member stated that the visualisations should "perhaps have a stronger correlation with hush tones and defined shapes, baselines with wide and soft shapes with animations that follow the beat more consistently".

Other members of the audience stated that they would like to see a better relationship between the code and the visuals. For example, one audience member stated that "it would be really cool once you edit a code line and then activate it for it to morph into a visualisation for as long as this line of code is active". Another audience member stated that "it would be better to not just relate the function to the sound but relate every beat to the sound or to the code responsible for that beat".

Other suggestions included increasing the readability of source code, increased immediacy of actions, increasing uniformity between the visualisations and the use of images to assist with the understanding of high level concepts.

5.3.7 Follow-Up Interview

A follow-up interview was conducted with the live coder to further examine the visualisation approach taken and the overall usability of the visualisations. See Appendix D for the full interview transcript.

5.4 Discussion

Overall enjoyment of the visualisations was high. This was the case for both the aesthetic and didactic visualisations.

The enjoyment of the aesthetic visualisation was higher and generally increased during the earlier stages of the performances suggesting that the aesthetic visualisations held the audience's attention consistently for a longer period (see Figure **??**).

The didactic visualisation had a more constant decrease in enjoyment throughout the performance. Audience suggested improvements offer some insight with some stating that the visualisations were competing with the projected code. One audience member stated that they "found them distracting" and that they "preferred just to read the code". Given that more than half of the audience stated that both visualisations contributed positively to enjoyment, there is indication that the didactic visualisations contributed more to audience fatigue through the performance with decreased effectiveness during the later stages of the performances.

As expected, the didactic visualisations had a clear educational benefit over the aesthetic visualisations, contributing to increased understanding throughout the performance. Nevertheless, features of the visualisation still confused the audience. For example, one audience member stated

that "it felt harder to understand the code" during the didactic performance than the aesthetic performance. Similarly, a number of audience members discussed the need for a better relationship between the visualisation and the music.

There was indication that the experience of the live coder and the experience of the audience was fundamentally different. For example, many members of the audience stated that they drifted between focussing on the music, focussing on the visualisations and focussing on the code. However, the live coder stated in the interview that his focus was dedicated purely to the code and the music rarely drifting. In one particular section of the interview, the live coder stated: "I definitely wasn't paying attention to them on the day. In fact I tuned them out as best I can because I am just trying to focus on the code". This conflicted with some members of the audience. For example, one stated that "you could see the code being written and the visualisations helped to show when a piece of code started working".

The strategy for audience members for either understanding or enjoying the performance proved to be different to that of the live coder. Regarding distractions, one audience member stated that "the visualisations were interesting but distracting". In comparison, when asked if the visualisations were distracting the live coder stated: "Ah, no. In general I'm just so focussed on the code". This difference may be due to the varying experience levels or the pressure of the situation.

5.5 Conclusion

An application of the visualisation of code to live coding has been examined. Results indicate the importance of providing visuals that complement the live coding performance. Visualisations targeted at the education of the audience showed a qualitative increase in understanding throughout the performance whereas visualisations targeted at the aesthetics of the music and performance indicated audience retention and a reduced fatigue effect over the didactic visualisations.

There was indication throughout audience feedback and the follow-up interview that the visualisations had potential. Desirable features within both visualisations could be further utilised building on the ideas developed including reacting instantly to code changes within the visualisations and providing a more effective visualisation of the relationship of code to music. These developments will provide a baseline for future visualisations.

One question to come from this study is the need to understand why the audience and the coder report such different experiences. Differences between the interview with the live coder and the audience feedback indicated a division between the goals of the audience and the coder.

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

Follow-Up Interviews

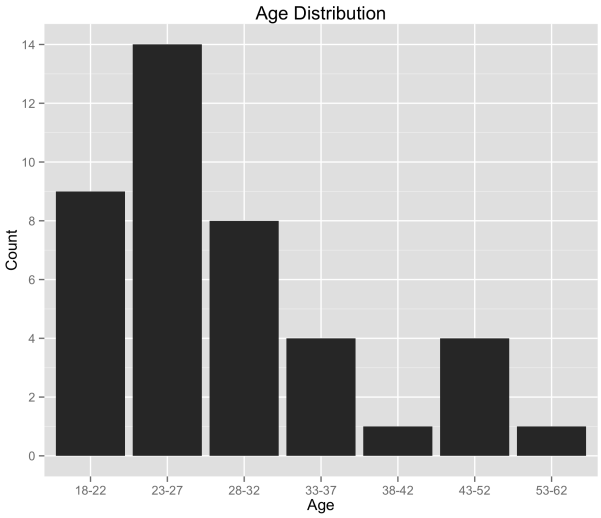
B.1 Respondent 1

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 un-

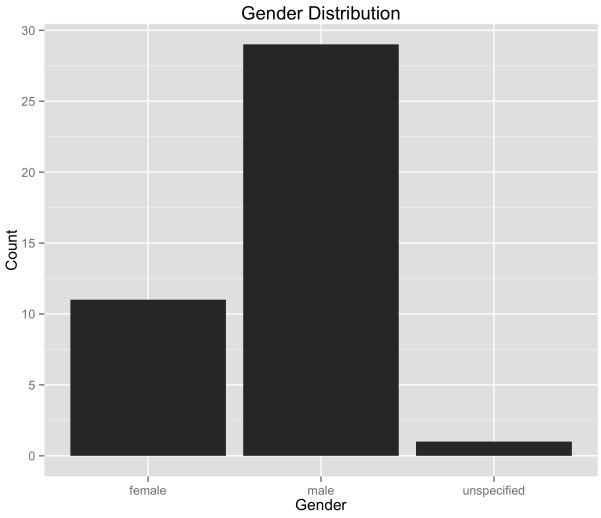
derstand when the code is ẗreadyänd starts working to make music. It has something to do with the highlighting the text, but that also confuses me. (I understand that most of the music is stored in the program as s̈ound bitesöf real instruments, but sounds can also be made from scratch as mathematical 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.

Visualisations

Appendix A - Survey Results

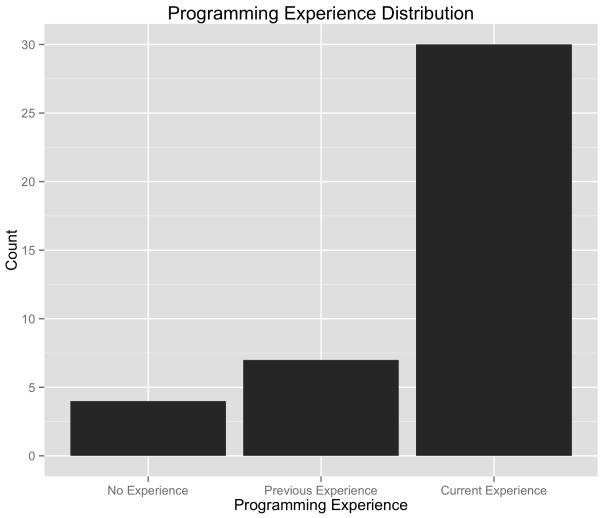


(a) Age Distribution

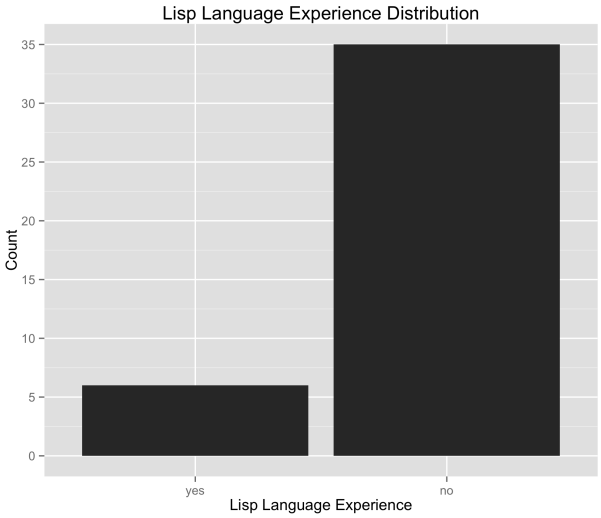


(b) Gender Distribution

Figure C.1: Basic Demographics

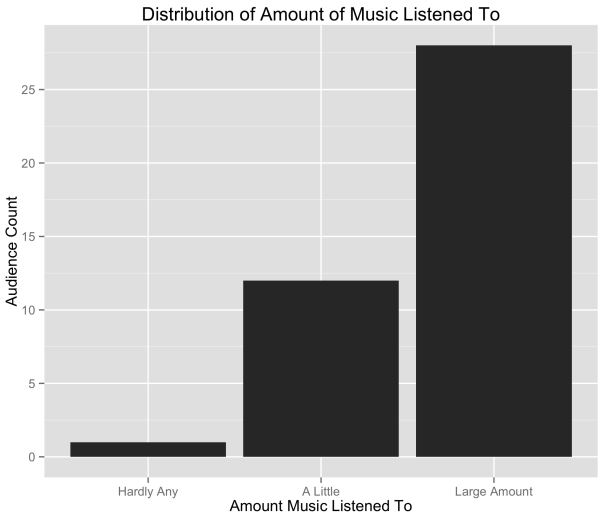


(a) Programming Experience Distribution

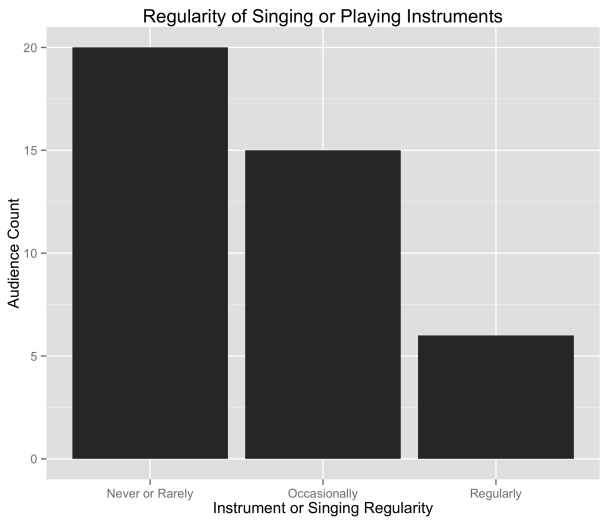


(b) Lisp Experience Distribution

Figure C.2: Programming Demographics



(a) Listen to Music Regularity Distribution



(b) Playing Instrument or Singing Regularity Distribution

Figure C.3: Musical Demographics

Appendix B - Survey

Part A

Age?
18-22
23-27
28-32
33-37
38-42
43-52
53-62
63+

Gender? _____

How many live coding performances have you been to?
This is my first one
I have been to one or two
More than two. Please indicate the approximate number: _____

How much music do you regularly listen to?
Hardly any
A little
A large amount

Do you play an instrument or sing?

No - I would not consider myself a musician or singer
Occasionally
Yes - I play or sing regularly

How much experience do you have with programming?
No experience
Some experience
I currently program for my study/hobby/work

Do you have much experience with the Lisp family of programming languages? (e.g. Scheme, Lisp, Clojure, Racket)
Yes
No
I dont know what youre talking about

Part B and C (repeated for the two performances)

How would you rate your levels of enjoyment during the beginning, middle and end phases of this performance?
Circle one alternative for each phase - interpret beginning, middle and end as you wish:

Beginning: Low Medium High

Middle: Low Medium High

End: Low Medium High

Did the projected visualisations help with your enjoyment of the code?
Yes
No
No opinion

How would you rate your understanding of what the code was doing during each phase? Circle one alternative for each phase - interpret beginning, middle and end as you wish:

Beginning: Low Medium High

Middle: Low Medium High

End: Low Medium High

Did the projected visualisations help with your understanding of the code?
Yes
No

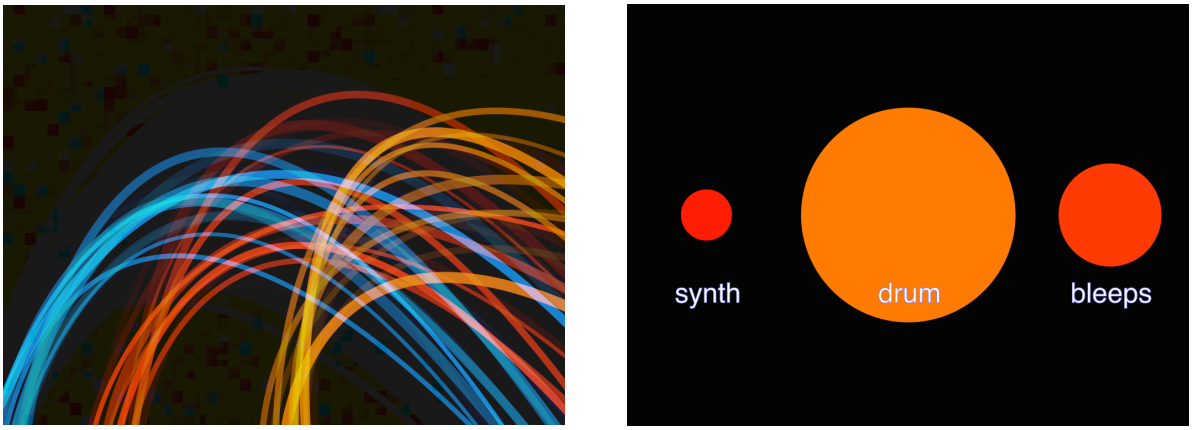
No opinion

This was a live performance! Did the projected code and visualisations help communicate the feeling that the performance was live? If so, how?

Part D

Do you have any suggestions for how the visualisations for either performance might be improved? (Continue answer on the back of this sheet if you wish)

Appendix C - Visualisations



(a) Aesthetic Visualisation (b) Didactic Visualisation

Figure C.4: Visualisation Examples

Appendix D - Interview Transcript

Aesthetic Visualisation

Ben: So the first little synth thing in this I knew pretty much exactly what I wanted to do. I was going to do random pitches at 16th notes. I think I probably did this synth thing exactly the same in other performances.

Arrian: Is this a common technique you use?

B: Yeah, well. This was not even prepared specifically for this performance but I've done stuff like this in the past. This is a pretty common trick I use for getting up and running early.

I think the bass is playing at this point but I can't hear on these speakers.

A: How much planning went into this performance?

B: This performance was pretty safe. It was pretty simple and pretty preplanned. What I would say is that the form of all of the instruments are pretty standard for my live coding.

So for the bass to go through a list of pitches like that... I do that alot.

For the drums I play with some sort of modulation of the sample slot, of the kit... I do that alot.

I probably choose slightly different parameters each time through but, yeah, none of these things are really adventurous by the standards of my live coding.

A: Why is that?

B: Honestly, the main reason is that these things work well and I've tried other algorithms and they don't sound as good as these simpler algorithmic structures with judicious choice of parameters.

Though I feel that as an artist sure, as a programmer this is not so interesting because the algorithms are pretty safe.

There are a lot of people in computer music today that use fancy algorithms, they use cellular automata, they use generative algorithms but I think it works better if you use... I've had more artistic success using these simple algorithms and then just using my musical experience and intelligence to select good parameters...

A: Did you plan the performance around the visuals?

B: To be honest, I think in the second piece I tried to limit the callback rate because I knew that some of the visuals in the didactic setup worked better with a longer callback rate.

A: Do you think the visualisations held you back?

B: No, it certainly didn't hold me back. In some ways it is nice to have constraints.

I'm now going up an octave. It gives it a harder edge...

It is interesting that I'm very conscious of the hypermeter. I'm just grooving along and it just makes sense to evaluate in time.

A: Did you find this visualisation distracting?

B: Ah, no. In general I'm just so focussed on the code.

This little bit is adventurous. This drum bit I didn't know exactly what samples were in those slots. Before I was just guessing some numbers and picking numbers I thought might be good.

This bit is certainly more intense. This is more european house. I don't actually know if it is european house but I'm sure there is some name for this genre.

This end bit is a bit new. I hadn't planned to end it like that.

A: Did it occur by chance?

B: Not so much by chance. I got to the end and wasn't really sure how I would finish this. This is true making it up.

A: Were you happy with the performance?

B: Yeah.

I'm actually going diatonically out of the scale I was using for the whole piece.

Yeah, I hadn't planned to finish like that but... yeah, yeah, I'm reasonably happy with that.

B: I think in terms of the surprising stuff. There were no surprises when I started or added each new instrument. I knew pretty much exactly what I was going to do. I might not have had the exact parameters in mind. I would have put maybe a 60 instead of a 70. Even when I didn't have an exact number in mind I would have had an approximate number in mind... loud vs soft.

Once stuff is going then I think all bets are off and I at least don't really think through what I'm going to do after that. Generally I'll go back and start messing with stuff. In that case I went back and messed with the synth.

I did a bit of interesting stuff with the drums. I went from a more groovy and pretty standard drum beat to a heightened beat which definitely changed the mood of the piece.

I'm not unhappy with that. I'd probably do something different next time just because you do something different every time but that was one of the things that surprised me.

In general I was pretty happy with it. I think it is a good sound palette. The drums grooved pretty well which is an important thing. The bass line was pretty cool, though I couldn't hear it in that recording.

A: In terms of the visualisations, do you think they added anything to the piece?

B: I think they added something. I think they are just ambience. It is definitely cool to have that stuff going on that is a little visually interesting but I wasn't paying attention to them even then. I definitely wasn't paying attention to them on the day. In fact I tuned them out as best I can because I am just trying to focus on the code.

Like I was saying before there were times where it was hard to see the code underneath the visualisations.

A: Was this during the aesthetic or didactic performance?

B: I think it was more the didactic and I think it was the text in the didactic ones and not in the aesthetic.

I definitely like the visuals. They definitely add something and they don't take anything away even though I wasn't paying attention to them. I still see the text as the main thing but the visuals are gravy and that was nice gravy.

A: The audience was reasonably computer literature. Do you think they would have preferred to focus on the code, the music or the visuals?

B: That's a good question. I don't know. I'm really curious.

Focus is a funny thing. You rarely explicitly go "alright, I'm going to focus on blah and focus on blah". I think you drift. Probably at different points they were paying attention to each.

I think I'm a bad judge of what people pay attention to because I pay attention to completely different stuff when I watch it. What I pay attention is probably completely unhelpful in terms of an indicator for what even a computer literate audience member would be paying attention to.

Didactic Visualisation

Ben: Now this one starts at a slower tempo. Half the speed... 60bpm vs 120bpm.

Arrian: Was this due to the nature of the visualisation?

B: No, that was just to be different. They both work fine with the visualisations. In fact the visualisation pretty much work with whatever tempo.

So this one is a slower starting one. I still get it going fairly quickly.

A: Did the visualisations get in the way here?

B: It wasn't too bad because it's not over the top of where I was trying to work.

This is one of the things that I did have the visuals in mind when I put together this thing. Initially you have the fast spinning visuals. I knew that I was going to slow this one down and go for two bars of eight beat long sustained chords. I knew that that would look cool as a slowly rotating thing.

In fact I stuffed it up there. It wasn't so much a typo as I did a tricky thing where I tried to have a couple of overlapping temporal recursions and filter out only the fast one keeping the slow one going. But that relies on changing the code once you've got it to the state you want.

I think in general with these parameters I was just messing around. I knew the general form.

I've got a couple of polyrhythms. I really like this bit. I think it works well.

A: Despite the timing issue with the visualisations?

B: Timing is off by half. We knew that was a problem. I still think it works pretty well. I think this one has real potential but it is disappointing that they didn't sync up.

This one is just grooving. I quite like the beat in this one.

A: Did the visuals affect your ability to see here?

B: I can't remember to be honest. It wasn't a big problem to be honest. It probably only happened once through all four pieces.

A: Were you tuning out these visuals during the performance?

B: Even when I'm watching it I'm tuning out the visuals but definitely during the performance.

I don't think this was planned. It is a fairly standard part of my live coding toolbox. I'm just changing the pitch. So it's to do with the bass. Quick ones and then long ones.

Then I changed the offset of the chords and the chords would come in staggered. I don't know how well this one worked in the end. I like bits of it but...

This one I think the stuff that is kind of up beat and is really groovy is easier to do than this.

This bit was disappointing. I had a cool ending in mind and the I stuffed it up here. I forgot to put the tick symbol.

A: Did you manage to pull off the cool ending in the second performance?

B: No, I tried to do the same thing and stuffed it up in the exact same way. That was interesting and frustrating. I had a cool ending that I just thought of that day where I was going to do some harmonic organ-y stuff, take it through the circle of fifths and do an interesting chord progression but really kind of draw one out for a slow finish. I just forgot to quote that symbol when I went from the minor to the major.

If I had other instruments covering me I could of started it again but since everything had died if I started it again it would have been really obvious so I decided in the moment that that is where I would finish it whereas I had one more minute planned. I started to go down a path where I had a minute more of material to finish it off but then just dogged it. Frustratingly I did not quite the same mistake but a similar sort of mistake in the second one. It's kind of really rare... obviously I make typos but I don't tend to make them in that way.

A: Was there some reason for the mistakes?

B: Not really.

A: Chance?

B: Yeah, just chance. Just life.

A: Was the main goal of this performance to entertain the audience... beyond the research?

B: Yeah, I think so. I always want the people to enjoy themselves. I tried to keep them pretty short. I think every little set was under ten minutes. If you're going to do a live coding set longer than ten minutes it needs to be bloody good. So in general I try to stay under ten minutes.

It's interesting, some of my earlier videos are longer than ten minutes and I watch them now and I think 'this drags on'. I'm much better at it now and I'm much better at making things happen quickly. I'm especially better at getting stuff up and running, partially because I'm an emacs guru and have all the snippet magic to make that happen but also you just learn the little extempore tricks and the general tricks for getting things up and running.

For example in the aesthetic set, there was probably stuff going after only 10 seconds. For this one there was probably stuff up before 30 seconds. I reckon you've got to get something up before 30 seconds.

A: Was boredom getting to you?

B: Not really. Certainly not in a big way. By the fourth I was like "I'm done, this has been a lot of live coding, I'm sort of out of ideas".

It's not even fair to say 'out of ideas'. I had that cool idea about how I was going to finish it that I dogged both times. It's just exhausting. It really does take a lot of concentration.

I was done at the end and I was pretty happy it was done. I enjoyed it, I had a good time but I was glad it was done.

Bibliography

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