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

April 2014

Typeset in Palatino by TEX and LATEX 2ε.

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

Arrian Purcell
24 April 2014

To fluffy the cat.

Acknowledgements

Here is where you thank the people who helped you along the way…

Abstract

Live coding is a method of performance that presents audio and visual content to audiences through programming. Often ”showing the code” is 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

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

-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

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

1.3.1.1 Code Visualisation

Visuals based on Code Augmentation (eg. infographics, annotations, sparklines)
Visuals based on Abstraction (eg. scheme bricks, gource, code flow)
Domain Visualisations (eg. fluid source code views, indentation, class diagrams)

1.3.1.2 Music Visualisation

Generative Visualisations (eg. frequency wave, VLC/iTunes visualisations)
Associative or Emotive Visualisations (eg. video art, sampled video)
Domain Visualisations (eg. ableton, sheet music)

1.3.2 Old 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 'Self-illustrating phenomena'

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

Code Visualisation

//Generative Visualisations //-visualisations that respond to typing

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

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

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

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 -generates 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 -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

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)

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 'model based' visualisation 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 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.

2.1.1 Music vs Visualisation

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

2.2 Music Visualisation

2.2.1 Taxonomy

2.2.2 Live Performance

2.3 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...

2.3.1 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)

2.3.1.1 Design

Design...

2.3.1.2 Coding

Coding...

2.3.1.3 Comprehension

Comprehension...

2.3.2 Application of Live Coding to Software Engineering

The application of live coding to software engineering...

2.3.2.1 Dissemination of Code Understanding

2.3.2.2 Multidisciplinary Cohesion

2.3.2.3 Visualisation Framework

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 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.

3.6 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.)

3.7 Supplementary Observations - remove

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.

Chapter 4

Visualisation Experiment 1

4.1 Rationale

4.2 Procedure

4.3 Results

Chapter 5

Visualisation Experiment 2

5.1 Rationale

5.2 Procedure

5.3 Results

Chapter 6

Conclusion

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

Appendix A

Survey Results

Appendix B

Visualisations

Bibliography

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