Graphic Score Grammars for End-Users

Alistair G. Stead
University of Cambridge
Computer Laboratory
Cambridge, UK
ags46@cam.ac.uk

Alan F. Blackwell
University of Cambridge
Computer Laboratory
Cambridge, UK
alan.blackwell@cl.cam.ac.uk

Samuel Aaron
University of Cambridge
Computer Laboratory
Cambridge, UK
sja55@cam.ac.uk

ABSTRACT

We describe a system that allows non-programmers to specify the grammar for a novel graphic score notation of their own design, defining performance notations suitable for drawing in live situations on a surface such as a whiteboard. The score can be interpreted via the camera of a smartphone, interactively scanned over the whiteboard to control the parameters of synthesisers implemented in Overtone. The visual grammar of the score, and its correspondence to the sound parameters, can be defined by the user with a simple visual condition-action language. This language can be edited on the touchscreen of an Android phone, allowing the grammar to be modified live in performance situations. Interactive scanning of the score is visible to the audience as a performance interface, with a colour classifier and visual feature recogniser causing the grammar-specified events to be sent using OSC messages via Wi-Fi from the hand-held smartphone to an audio workstation.

Keywords

Graphic Notation, Disposable Notation, Live Coding, Computer Vision, Mobile Music

1. INTRODUCTION

Graphic scores were a regular feature of contemporary composition and performance practice from the 1950s onward. The intention in graphic scores is both to circumvent the limitations of common music notation, and to allow more interpretive freedom to the performer (e.g. offering generative rules or systematic constraints within which improvisation can be directed or coordinated). A key element in the use of graphic scores is that the 'grammar' of the score – the syntax and semantics of its visual elements – is invented by the composer, as part of the creative process. The invention of new notations allows both increased structural abstraction, and conceptual release from the analytic systems associated with conventional notations. In this respect, use of graphic scores resembles those practices in creative computer science research, where the first step in a project is to invent one's own programming language, after which it will be possible to explore design problems from a new conceptual perspective.

The starting point for our work was to create a digital performance interface that incorporated both elements of performance practice with graphic scores: the ability for the composer to define their own syntax and semantics; and the ability for a performer to interpret the score freely, in ways that can be appreciated by an audience. Ideally, we would like the composer and performer to be the same individual, creating a new digital artform that combines elements of composition and improvisation, as in Live Coding [1]. We wanted to enable both of these elements via a digital channel, with an interactive programming language used to define the syntax of the score, and an interactive machine vision system used to recognise and 'perform' it, under user control on a sufficiently large scale that it can be recognised by an audience watching a stage performance.

The use of computer vision as a component within musical performance systems has been commonplace for many years -(e.g. BigEye MIDI controller [12] or CV.jit for Max/MSP + Jitter [7]). Many of these systems have been oriented toward detecting the location, pose or gestures of the performer as a real-time control device that can be used on stage or in an installation. In these applications, the main challenges are background subtraction and object tracking – challenges that are likely to become somewhat reduced as depth cameras and fast body part recognition of the type used in the Xbox Kinect become ubiquitous. However, there have also been some experiments in using live computer vision to produce musical sounds directly from recognition of sketched drawings, as in the case of McLean's Acid Sketching [6]. Such systems can rely on simple and robust approaches to 2D sketch interpretation.

Despite machine vision systems being developed specifically for simplicity and use by artist-programmers, many are still relatively complex to create and configure, requiring technical training and understanding of the underlying infrastructure. Our goal was to create a machine vision system that could be used to define novel visual grammars, and to interpret them sufficiently quickly and robustly that they could be created by a musician with no prior programming experience, possibly even on stage in front of an audience. The scenario we imagined was that the graphic score could be drawn on a physical surface at large scale – for example on a whiteboard – so that an audience could see the components of the notation and gain an impression of the planned structure of the piece. The performer would then use a hand-held camera to scan over parts of this notation, modifying the musical output in response to his or her camera-movement gestures. If the camera is mounted on a touchscreen device such as a smartphone, then the touchscreen itself could be used to 'program' the grammar that interprets the score, and even modify that grammar while on stage, without having to interrupt the performance to move to a desk and sit behind a laptop.

2. BACKGROUND

Musical notation is typically used for communication and reuse/reference. Both humans and computer systems need an interpretation model to read and understand it. Quick human comprehension of notation usually takes years to learn. The notation system typically needs to be a broad 'one fits all' solution to be worth the learning effort required. Computers are capable of learning more quickly and therefore do not necessarily face this problem.

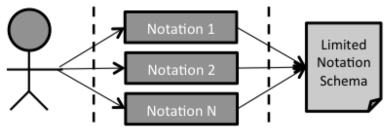

Figure 1: Current Limited Notation Model

We believe that the ability to create a disposable notation is extremely powerful and only now possible due to advances in notational schema definition. Existing systems such as Acid Sketch and reacTable usually use fixed notation schemas; they require reasonable learning and are fundamentally limited to the scope of the schema (see Figure 1). Grammar specification allows construction of notation that is appropriate for the piece the performer intended. By referring to the notation as 'disposable', we mean a fast construction (within minutes) and a minimal cost in effort.

2.1 Live Coding

Live coding is a growing approach to computer music performance [2] in which performers use dynamic programming environments to describe sound output. Properties of sound and algorithms modifying sound parameters are edited by the performer in response to audience reaction, other performers, or evolving artistic intention. At present, live coding generally involves programming on laptops. As demonstrated by the reacTable system, a performer physically moving over a large notation space is likely to provide more audience entertainment or energy than simply using a projected screen or laptop. Furthermore, the use of a simple notation, rather than complex programming structures, may reduce audience information overload and provide a simpler visualisation from which to infer mappings between the sound and notation.

There have been recent efforts to control musical parameters using smartphones [13]; in particular, using data from users' interaction with a touch screen, accelerometer and microphone. Mobile camera data has been used to control MIDI parameters in systems such as CaMus [9]. Whist it is not possible to 'program' with these systems, it is clear there is unrealised potential for live coding environments on smartphones, with collaborative performance, mobility, and sensory interaction being just a few potential benefits.

2.2 Grammar Specification

Our grammar specification tool was intended to fulfill several design requirements: be powerful enough to express a wide range of grammars, simple enough for non-programmers to understand; use flexible syntax; encourage experimentation; and be fast enough to use in a live coding scenario. The small screen of a smartphone renders text-based languages unsuitable for programming. Direct manipulation is one approach to reduce cognitive load, making the tool itself "disappear" [10]. A visual language is suitable for our purposes as semantics can be encoded in visual objects and users can directly manipulate them using touch. Visual languages exist in many forms [4]. Holz and Baudisch [5] claim that users have a touch accuracy of 95% on targets of over 8.6mm. We intend to use visual objects of this size or greater; an iconic language fulfills our requirements. We draw from the simple concept of condition-action rules, as used in popular end-user programming systems such as Agentsheets [8].

3. ARCHITECTURE

Figure 2 shows the relationship between the 'main menu', and the three stages. The menu is intended to allow users to move quickly between stages.

3.1 Conceptual Architecture

The system architecture, conceptually, is composed of three successive stages: ink capture, grammar specification and performance. The 'ink capture' and 'performance' activities both use computer vision techniques to detect coloured blobs on a lightly coloured background (e.g. a whiteboard).

Figure 2 shows how musicians interact with the notation system. After specifying a grammar, users are able to draw desired notation and scan the smartphone over it to perform. During performance, they can continue writing notation, direct another performer to add extra notation, or reuse existing notation by scanning the phone over it.

Users can specify notation grammar using a grid, on which they place tiles. The tiles are moved onto the grid from a 'palette', which contains the set of available tiles. When a tile is directly next to another, it is connected and forms part of a rule. Tiles can be connected in any order or shape (see Figure 3b). Performance only takes place with a valid grid. A valid rule has the following syntax: one or more 'identifier' tiles; one or more 'trigger' tiles; one 'value' tile; and one or more 'property' tiles. Flexibility in the ordering and shape of rules allows users to customise their grid and focus on semantics rather than syntax.

The 'identifier' tiles represent unique OSC streams, of which there can be several. One or more streams can be affected by the same rule. The 'trigger' tiles are conditions that have to be met for the rule to execute (e.g. a red blob is in view). If multiple triggers are used in a single rule, all triggers must be satisfied. The 'value' tiles represent a number that can be applied to a 'property' of the audio output. Currently, 'value' tiles can either be a single value, derived from area, or an envelope over time.

For example, refer to the bottom left rule in Figure 3b, which has a single tile of each type: An 'identifier', referring to stream alpha; a 'trigger' , which executes the rule if the camera view contains a grey blob; a 'value' tile, which is a single value somewhere in the middle of the scale; and a 'property' tile representing pitch. The result of the rule is that when the grey blob is in the camera view, the system plays a middle note on stream α.

Parameters of tiles can be changed by double tapping on a tile. A dialog will appear, allowing modification of properties specific to that tile type, such as colour, value or time. Figure 3a shows the settings dialog for the 'value' tile, which represents a percentage or a piano note. A 'trigger' tile can be modified to refer to a particular blob colour.

3.2 Technical Architecture

The application makes use of Java and C++ via the Android Native SDK, allowing us to use OpenCV, a comprehensive and widely available vision library. The vision system detects, recognises and tracks coloured blobs by using simple white balancing, background removal and image segmentation techniques. Colour data collected during the first stage is used to train a classifier, which recognises new, unseen notation (blobs) during performance. Notation is

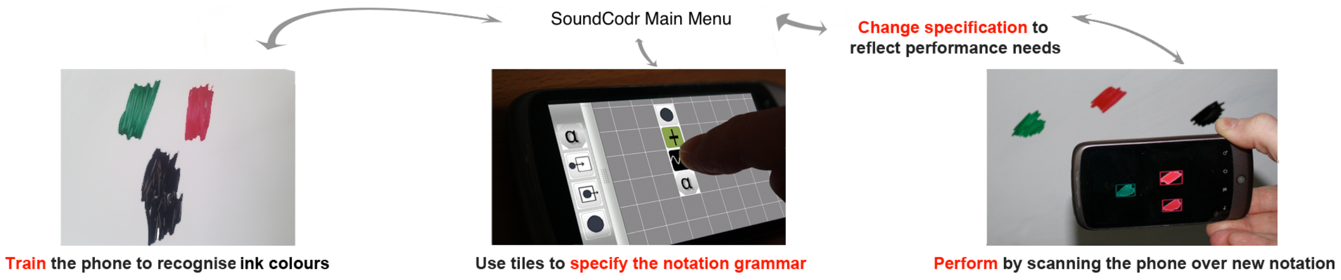
SoundCodr Main Menu
Change specification to reflect performance needs
Train the phone to recognise ink colours
Use tiles to specify the notation grammar
Perform by scanning the phone over new notation

Figure 2: Conceptual Architecture

tracked to allow the system to trigger appropriate events. We use a frame-by-frame comparison of blobs, together with a probability model to measure the likelihood of these events (e.g. enter, exit). The events are then propagated to the grammar interpreter.

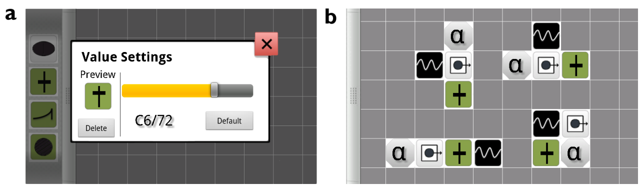

Figure 3: a. Settings Dialog for a 'value' tile. b. The same rule in four different possible positions.

3.2.1 *Open Sound Control Messages*

During performance, each frame is processed by the vision subsystem, which then returns the state of the set of known blobs (entered, exited, visible) and their properties (area, orientation, position etc.) The appropriate OSC messages are then dispatched to the remote server via UDP. A remote server is used so that audio can be produced at very high quality using external speakers. The use of Wi-Fi allows the phone to be moved a great distance from the remote machine without limiting movement. Currently, there are two property tiles (pitch and volume), and therefore two types of OSC message required:

```
/set-volume {Identifier, Time, Value%}
/set-frequency {Identifier, Time, Note}
```

For instant changes, the time parameter is 0, and overrides any current changes for the targeted synth. If the value is an envelope (a change over time), pitch changes at a linear rate. We use piano notes for our purposes (e.g. D3), rather than frequencies, to increase usability for musical novices and to allow construction of chords and melodies for experts.

Our audio server was implemented using Overtone [1], an open source audio toolkit with a strong focus on collaborative programmable music. Overtone combines SuperCollider's sophisticated synthesis capabilities with Clojure's linguistic flexibility, advanced concurrency semantics and full Java interoperability. On initialisation, the audio server defines and triggers a number of synths. An OSC server listens for incoming control messages, which are then dispatched to functions that modulate the current note and volume of the specified synth. Clojure's immutable datastructures and lock-free mutable references allow the audio server to coordinate an arbitrary number of concurrent OSC clients whilst avoiding classic concurrency problems such as deadlock and blocking writers.

Table 1: Experimentation Period Metrics

Participants	No. of Sessions	Max No. of Rules
P1	2	4
P2	1	2
P3	2	3
P4	1	3
P5	1	3
P6	1	3
P7	4	2
P8	1	7

4. EVALUATIVE STUDY

Our system is intended for users without substantial musical experience. Ideally, we would like there to be little usability difference between experienced digital music performers and non-musicians. We also do not expect programming experience to affect the competence of users due to the unconventional style of programming. To determine the results of these hypotheses, we carried out an evaluative user study. We discuss the study briefly – for more information see [11]

4.1 Design

The study was between-subjects, using two groups: digital music performers and non-digital music performers. The two groups of 4 participants were given 4 tasks, followed by 15 minutes of free experimentation. Sessions were recorded, with participants being encouraged to mention any problems. Grammar specification time; performance time; the largest rule size in each performance; the number of rules constructed in each performance; and the number of colours used in each performance were also recorded.

Participants were supplied with a whiteboard and several coloured markers. A short tutorial was given to familiarise participants with concepts in the application. Each task focused on a separate type of tile, allowed usability evaluation of tiles in isolation and reducing learning effects.

After personal experimentation participants were given a Cognitive Dimensions of Notation Questionnaire [3], in which they were asked questions relating to grammar specification. Each question is directly related to a CD – a design principle for programming languages. The questionnaire data was cross-referenced with the recorded audio to strengthen the validity of our conclusions.

4.2 Discussion of Study

Group 1 (P1-P4) are digital music performers, with experience with systems such as Supercollider, Pure Data, Chuck and Max MSP. Group 2 (P5-P8) have no similar system experience. The level of programming experience varies between basic and intermediate. Half of all users had smartphone experience; this varied across both groups.

Due to the nature of the study, statistical analysis of qualitative data from the questionnaire would be of limited value. However, metrics recorded during the exploratory experiments can be used to make some observations of the amount of expression and experimentation the language encourages. Table 1 shows that participants from both groups were similar in the number of rules used. P8 (a non-digital performer) was very confident, using 7 rules, twice the mode of the other participants. Table 1 shows that both groups collectively had equal numbers of sessions, although P7 was very confident, carrying out 4 experimental sessions (using different sets of rules).

4.3 Questionnaire Results

We obtained interesting qualitative results from the questionnaire, and verbally during the experiments: P2 described the screen as being *'small'* and later mentioned double tapping was difficult and that tiles *'don't fall accurately'*, even though the tile drag-and-drop system highlights the destination square in yellow. This participant also pressed several external buttons by mistake, indicating a lack of smartphone experience, which may be a barrier to using the system. Conversely, P7 also had no smartphone experience, but became familiar with the system very quickly and had the most sessions during the experimentation period.

P4 stated that it is *'not always easy [to compare or combine tiles] as the screen only has a finite area'*. It is possible for rules to extend off the grid. Flexible ordering and positioning of tiles was intended to address this. It was recognised that tiles with values are difficult to compare; to address this, we use live dynamic tiles, which update to reflect their value. When asked about dependency, participants were not aware it was possible to override one rule with another. A particularly interesting CD is 'progressive evaluation' – checking work done so far. Most participants found this easy by tapping 'back' and then 'perform'. P2 found it *'not so easy; you need to change modes which is a little cumbersome'*, although this may have been due to unfamiliarity with the phone's buttons.

4.4 Group Comparison

Our qualitative feedback suggested that most participants found the system easy to use after reading through the initial tutorial. The majority of problems reported were received from P2, who reported having issues using smartphones. Table 1 shows that both groups performed similarly, with P7, a non digital music performer, experimenting the most. It is possible that digital music performers were constructing more complex performances and were therefore limited to fewer performances.

5. CONCLUSIONS & FUTURE WORK

We have described SoundCodr, a system allowing construction of graphic score grammars on a smartphone. The creation of the system was motivated by research into end-user programming and live coding. The system reaches the goal of allowing users to define and perform their own graphical scores. Our evaluative user study confirms that the system is of interest to musicians and non-musicians alike, and gives users a new and innovative method of music creation. We found that users embraced the ability to progressively evaluate their grammar by moving between performance and grammar specification.

Our study indicates that there is little difference in usability for digital musicians and normal users. The qualitative feedback was positive, and included comments such as: *'[using tiles] is very straight forward; building blocks structure is simple to learn and holds potential'*, and *'it is possible to have an end product in a reasonable time, as the structure of the notation is fairly simple'*. The study has given valuable feedback for future improvements.

At present, the system uses two property tiles (volume and pitch). The system was built for extensibility and would benefit from the ability to manipulate other musical properties such as timbre. It would also be interesting to manipulate a variety of different synths. Although the study focused on one controlling device per system, the audio server is capable of handling multiple devices concurrently sending OSC commands. There are potential 'values' of the notation that could be used in the future (e.g. the distance between blobs, relative proportion and circularity). We would like to enable local audio synthesis, which would allow experimentation in casual contexts, away from the laboratory, stage or studio. Our system demonstrates the vast potential of disposable notation in musical contexts; there is still much work to be done.

6. REFERENCES

[1] S. Aaron, A. F. Blackwell, R. Hoadley, and T. Regan. A principled approach to developing new languages for live coding. In *Proceedings of New Interfaces for Musical Expression*, pages 381–386. Nime, May 2011.

[2] A. F. Blackwell and N. Collins. The programming language as a musical instrument. In *Proceedings of the Psychology of Programming Interest Group*, pages 120–130. PPIG, June 2005.

[3] A. F. Blackwell and T. R. Green. A cognitive dimensions questionnaire optimised for users. In *Proceedings of the Psychology of Programming Interest Group*, pages 137–154. PPIG, April 2000.

[4] Y. Engelhardt. *The Language of Graphics - A Framework for the Analysis of Syntax and Meaning in Maps, Charts and Diagrams.* PhD thesis, University of Amsterdam, The Netherlands, 2002.

[5] C. Holz and P. Baudisch. Understanding touch. In *Proceedings of Human factors in Computing Systems*, pages 2501–2510. CHI, May 2011.

[6] A. McLean. Acid sketching. http://yaxu.org/acid-sketching/, November 2009.

[7] J. M. Pelletier. Cv.jit (software library for max/msp computer vision). In *IAMAS, Japan*, 2005.

[8] A. Repenning. *Agentsheets: A Tool for Building Domain-Oriented Dynamic, Visual Environments.* PhD thesis, University of Colorado at Boulder, Colorado, USA, 1993.

[9] M. Rohs, G. Essl, and M. Roth. Camus: Live music performance using camera phones and visual grid tracking. In *Proceedings of New Interfaces for Musical Expression*, pages 31–36. NIME, June 2006.

[10] B. Shneiderman. *Sparks of Innovation in Human-Computer Interaction.* Ablex Pub. Co, New York, New York, 1993.

[11] A. G. Stead. User-configurable machine vision for mobiles. In *Proceedings of the Psychology of Programming Interest Group*, September 2011.

[12] STEIM. Bigeye. http://www.steim.org/steim/bigeye.html/.

[13] G. Wang. Designing smule's iphone ocarina. In *Proceedings of New Interfaces for Musical Expression.* NIME, June 2009.