The Trinity Test: Workshop on Unified Notations for Practices and Pedagogies in Music and Programming

Chris Nash
Department of Computer Science and Creative Technology, University of the West of England,
Frenchay Campus, Coldharbour Lane, Bristol, BS16 1QY, UK
chris.nash@uwe.ac.uk

ABSTRACT

This paper outlines a workshop to explore intersections of programming and music in digital notation. With the aid of the Manhattan music programming and sequencing environment (Nash, 2014), methods for representing both high-level processes and low-level data constructs in both domains will be explored and debated. The goal of this research is to establish ways of using music concepts to teach programming (and vice versa), working towards digital pedagogies and platforms supporting intrinsic motivation, virtuosity, and auto-didactic learning.

The proposed schedule begins with a presentation of findings from studies of both programming and music students, followed by an introduction to the Manhattan software, a sequencer supporting end-user programming (combining declarative and imperative programming idioms) for real-time manipulation of live music notation. The second half of the workshop invites participants to explore concepts in, and overlaps between, programming and music using the software (provided). Beginning with simple structured exercises and examples, the activities will proceed to freer exploratory design and experimentation, drawing on the participants' backgrounds in music and programming. The workshop concludes with a discussion of conclusions and future directions for research.

1. INTRODUCTION

Programming and music are rewarding creative domains mediated by notation [1], yet each presents steep learning curves and high entry thresholds. In formal practice, initial progress often depends on a source of extrinsic motivation (e.g. a parent, teacher, or other external reward), before the individual acquires the knowledge, confidence, and self-sufficiency to enable deeper learning and enjoyment; the pre-requisites of creativity. [2]

This workshop is designed to explore and develop concepts and notations for teaching both music and programming, considering aspects of programming language design and established practices in digital and traditional music. This proposal first sets out the background to the challenges faced by learners, educators and technologists, highlighting existing tools and pedagogies for end-user programming in music, before outlining a schedule based on discussion and hands-on experimentation with tools, such as the Manhattan environment.

2. BACKGROUND

Digital technologies to facilitate exploration and education have become a popular topic of research in both music [3,4] and programming [5,6], where learning environments seek to provide accessible, informative and enjoyable ways to explore and interact with concepts in a given domain. Notably, interest has recently been spurred by the recognition that existing school curricula do not provide a grounding in computer science principles. [7]

Programming languages provide a powerful tool for modeling and manipulating processes and ontologies in applied domains, but which students often find abstract and esoteric when taught without a practical context. [8]

Music, by contrast, is a notation-oriented domain that is appreciated by lay individuals, yet is becoming increasingly accessible to amateurs through the advent of digital tools (notably, sequencers and DAWs of varying complexity, from *GarageBand* to *Ableton Live*). This is not only attributable to improvements in usability and affordability, but the provision of a rapid edit-audition cycle in these programs, which allows a user to easily experiment with manipulating the notation then quickly audition the result. This increases the liveness (see [9], c.f. [10]) of the user experience, while reducing the onus on literacy and complex thinking. [3]

Music can thus be seen as an accessible and enjoyable context for teaching programming, as successfully explored by Sam Aaron's *Sonic Pi* project [6], which offers a cross-platform environment (supporting desktops and the *Raspberry Pi*) for live coding musical performance using a simple imperative-style interpreted language, connected to the *SuperCollider* engine. The tool exposes users – notably including secondary schools and other early- or non-programmers – to key coding concepts (such as expressions, variables, conditional statements, iteration, and functions), while also scaling to more elaborate generative applications of music.

Oriented around text-based generative algorithms, the challenge facing *Sonic Pi* lies in the integration with other established digital music workflows and mainstream musical practice, as encapsulated in the notations and interfaces of music software such as sequencers and typesetters. In contrast to the imperative style of most programming languages, such editors are based around the arranging of notes across timelines and tracks (or parts), suggestive of a more *declarative* style of programming, such as that seen in spreadsheets. [12]

3. THE MANHATTAN PROJECT

Manhattan [11] (see Figure 1) is an end-user programming environment based around the *soundtracker* style of sequencer software, characterized by grids of music called *patterns* containing text cells specifying notes or other musical events. Playback steps row-by-row down the grid, analogous to a digital player piano or linear sequencer timeline viewed vertically. In Manhattan, the standard MIDI sequencing functionality is extended with the option to define cell contents using formulas, in a mode similar to popular spreadsheet programs, but which are then evaluated at playback moves through a song.

This imposes a sequential order of execution, where each formula affects the 'state' of the music (the contents or interpretation of the pattern), engendering an imperative style of programming, but one where interaction is focused on editable musical data supporting aspects of declarative programming, and inheriting many of benefits that have made spreadsheets one of the more successful models of end-user programming. [12]

In the UI, the visibility of the data (music), rather than code (formulas), is prioritized, such that original sequencer-oriented user experience is unchanged, supporting traditional workflows, and the effect of code on data is apparent. [15] Like spreadsheets, users also have a scalable exposure to programming abstractions, where they can: avoid formulas entirely, in favor of more traditional sequencing interaction; insert occasional or simple expressions to script isolated dynamic behavior at points in a piece (e.g. conditional repeats, random elements); or use patterns of formulas to generate entire pieces (e.g. algorithmic music, minimalism, aleatoric music, etc.). The environment can run standalone or inside a VST/AU plugin (in a sequencer/DAW) to further facilitate integration with a user's existing digital music practices.

Manhattan is being developed as part of a research project involving artists, universities, and schools that is looking at tools to support creative and pedagogical practices in both music and programming. At UWE Bristol, music students are taught to code in C/C++, but take time to appreciate the power and utility of programming, even in the context of music (e.g. handling MIDI) and audio (e.g. effects, synthesis, and DSP). Among other initiatives, Manhattan is being used in lessons to introduce students to abstract thinking and modeling processes, key concepts in both software development and composition.

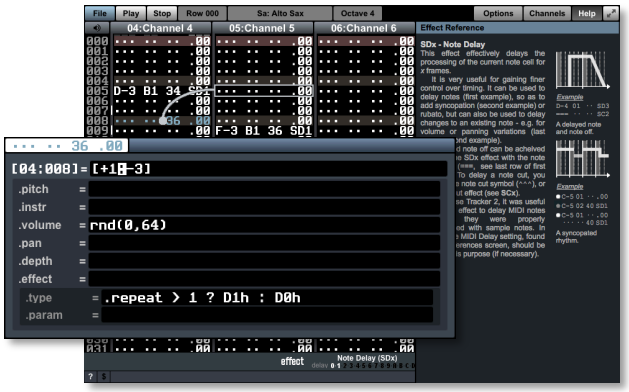

Figure 1 – Manhattan: Trinity software

Manhattan is also the basis for a project looking at virtuosity, peer competition, and *serious games* [13] in digital music, and their use to foster engagement with music and programming concepts. This aspect of the research builds on the established culture of notation-focused virtuosity in tracker use (e.g. the defiant embrace of now obsolete technological limitations on expression, such as 4 note polyphony; see [14]), restricting users to a single looped pattern (e.g. 4 bars) of music and challenging them to craft Manhattan formulae to generate and manipulate the music to create longer, more dynamic, evolving and musically interesting performances.

4. WORKSHOP OBJECTIVES

The workshop is designed with the following objectives:

- Discuss and debate practical applications of the intersection between music and programming practice and related concepts in their notations.
- Explore analogous concepts and processes in the syntaxes and semantics of music and programming.
- Experiment with specific musical examples of programming concepts using the Manhattan software.
- Establish future directions and collaborations for continued research in end-user programming and digital pedagogies for music.

5. SCHEDULE

A draft progression for the workshop is as follows:

12:30 – Welcome and Introductions (10m)
Host and delegates introduce themselves, briefly describing their background and respective areas of interest or expertise.

12:40 – Opening Remarks (20m, Chris Nash & Sam Aaron)
Audio/visual presentation with live software examples (e.g. *Manhattan* and *Sonic Pi*), highlighting concepts, notations parallels, and usability issues in digital music practices or programming languages, with reference to end-user computing.

13:00 – Initial Questions / Discussion (15m)
Open discussion amongst delegates of the issues raised in the opening presentation, specifically in the context of the delegates' own experiences and research interests. Used to help frame and guide subsequent interactive sessions.

13:15 – Interactive Session 1 (Example Exercises, 45m)
Structured exercises from provided materials, designed to introduce delegates to the fundamentals of the Manhattan tool, while also providing specific examples of programming concepts (e.g. variables, arrays, iteration, functions, conditional statements) presented in a practical musical context.

14:00 – Interactive Session 2 (Experimentation, 45m)
Drawing on and combining their own musical and programming experiences, delegates are invited to join one of two activity groups, both using Manhattan (or other tools, such as *Sonic Pi*) to experiment with new ideas: *Group CM (Code to Music)* discusses, explores, and develops musical expressions of ideas from programming or algorithmic / generative music; *Group MC (Music to Code)* explores the use of formulae to encapsulate traditional music practices, forms, or works (common practice music, MIDI arranging, electronic, folk and popular styles).

14:45 - Closing Discussion (15mins)
Brief review of issues and findings (or research questions) that emerged, and call for interest in further research / collaboration.

6. INTENDED AUDIENCE

As an exploration of end-user programming, no specific expertise is required. However, the workshop would particularly suit those with backgrounds, research interests or experience in: notation, composition (modern or common-practice), sequencing, programming (usage and semantics), pedagogy, virtuosity, live coding or usability.

7. REQUIRED RESOURCES

The workshop has no special requirements. Depending on attendance, it will require a single room with a capacity of 20-30, with tables arranged in a square. There should be a projector/screen with VGA or HDMI connection and a high-quality stereo system for computer audio (no PA is required). The ability to record the session is desirable, though this shall be managed by the workshop organizers, if facilities are not available in the room itself.

Participants are likely (and encouraged) to bring and use their own laptops, for which software will be supplied through USB sticks. Participants can retain the software. Workshop registration should indicate their preferred OS (OS X or Windows), or whether they need a computer to be provided. In the latter case, hosting in a computer lab may be helpful, though sharing machines is also possible. Access to WiFi (e.g. eduroam) is desirable.

If available, tea and coffee would be desirable in the corner of the room, for access throughout the workshop. No fee is required for participation in the workshop, unless deemed appropriate by the conference organizers.

About the Organizers / Speakers

Dr Chris Nash (principal organizer) is a professional programmer and composer – currently Senior Lecturer in Music Technology at UWE Bristol, teaching software development for audio, sound, and music (DSP, C/C++, Max/MSP). His research focuses on digital notations, HCI in music, virtuosity, end-user computing, systematic musicology, and pedagogies for music and programming.

Dr Sam Aaron (guest speaker) is a musician, researcher, and developer at both Cambridge University and the *Raspberry Pi* Foundation. He is an expert in programming language design and semantics, and creator of the *Sonic Pi* project, which uses music to engage children and other non-coders in programming. He is an active performing live coder, recently at Moogfest 2016.

Sam Hunt (technical support, recorder) is a post-graduate researcher at UWE Bristol, currently completing a PhD in music content analysis and generative applications in digital music composition, supervised by Chris Nash.

Dom Brown (technical support) is a post-graduate researcher at UWE Bristol, currently completing a PhD in gestural interaction for music, supervised by Chris Nash.

8. REFERENCES

[1] L. Church, C. Nash, and A. F. Blackwell, “Liveness in notation use: From music to programming,” 22, 2010, *Proceedings of PPIG 2010*, 2010, pp. 2-11.

[2] M.A. Collins and T.M. Amabile. “Motivation and creativity,” in *Handbook of Creativity* (ed. R.J. Sternberg). Cambridge, UK: Cambridge University Press, 1999, pp. 297–312.

[3] L. Scripp, J. Meyaard, and L. Davidson. “Discerning Musical Development: Using Computers to Discover What We Know,” in *J. of Aesthetic Education*, vol. 22, no. 1, Uni. of Illinois Press, 1988, pp. 75-88.

[4] H. Taube and A. Burnson, “Software to Teach Music Theory”, in *Proceedings of the ICMC 2009*, Montreal, Canada Aug. 16-21, 2009.

[5] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. “The Scratch Programming Language and Environment,” in *ACM Trans. Comput. Educ.,* vol. 10, no. 4, November 2010, pp. 16:1-16:15.

[6] S. Aaron, A.F. Blackwell, and P. Burnard. “The development of Sonic Pi and its use in educational partnerships: co-creating pedagogies for learning computer programming”, in *Live Coding in Music Education: Special Issue of The Journal of Music Technology and Education*, 2016 (in press).

[7] N. Brown, S. Sentance, T. Crick, and S. Humphreys. “Restart: The Resurgence of Computer Science in UK Schools,” in *ACM Trans. Comput. Educ.*, vol. 1, no. 1 (January 2013), 2013, pp. 1:1-1:22.

[8] A. Forte "Programming for communication: Overcoming motivational barriers to computation for all", in *Human Centric Computing Languages and Environments,* 2003, pp. 285-286.

[9] C. Nash, and A. Blackwell, “Liveness and flow in notation use,” *Proc. of NIME 2012*, 2012, pp. 28-33.

[10] S.L. Tanimoto, “VIVA: A visual language for image processing,” in *Journal of Visual Languages & Computing*, vol.1, no.2, Elsevier, 1990, pp. 127-139.

[11] C. Nash, “Manhattan: End-User Programming for Music,” Proc. of NIME 2014, 2014, pp. 28-33.

[12] J.F. Pane and B.A. Myers. *Usability Issues in the Design of Novice Programming Systems*. Carnegie Mellon University, Technical Report CMU-CS-96-132, 1995.

[13] M. Zyda. "From visual simulation to virtual reality to games," *Computer,* vol.38, no.9, 2005, pp. 25-32.

[14] C. Nash, and A. Blackwell, “Tracking virtuosity and flow in computer music,” *Proc. of ICMC 2011*, 2011, pp. 572-582.

[15] C. Nash, “The Cognitive Dimensions of Muisc Notations,” in *Proc. of TENOR 2015*, Paris, 2015.