How can expressive algorithmic composition software be made more productive and engaging for novice users?

Matt Bellingham
Matthew.Bellingham@open.ac.uk
Music Computing Lab / MCT

Supervisors name/s: Simon Holland and Paul Mulholland
Status: Part-Time
Probation viva: Before
Starting date: 01/07/2013

I. Introduction

Algorithmic composition systems are systems that generate music algorithmically in such a way that the same output is never generated twice. Such systems typically offer only limited levels of control to users unless they have detailed technical knowledge of both programming and music. Despite the many potential uses of algorithmic composition for non-programmers, this restriction appears to limit the extent to which novice users find such systems productive and engaging, thus restricting their take-up. The research investigates the extent to which principled and innovative approaches to interaction design might be able to alleviate this problem. Such approaches might enable musicians without programming knowledge to exert more musically meaningful and engaging control over complex algorithmic composition. It is hoped that the development of a musician-facing interface for algorithmic composition and playback will illuminate issues concerning the authenticity of recorded performance and inform the development of linear digital audio workstations (DAWs). It is also hoped to contribute to the development of music listening tools in the context of the continuing decoupling of physical media from domestic music playback.

II. Background

There are various distinctions between aleatory, chance and indeterminacy within music composition, and there are several techniques used which affect the composition or performance of a piece. An algorithm can be considered a set of operations which are to be performed. In this sense a composer's instructions to the performer can be considered an algorithm ('take the next step based on the following parameters'). It is common, however, for the term 'algorithmic music' to refer to choices which are made by non-human means (such as dice or computers). Algorithms have been used as part of the compositional process for hundreds of years. Musical dice games were popular in the eighteenth century, with composers such as Mozart creating pieces involving algorithms. Aleatoric techniques were later used by Marcel Duchamp, John Cage, Charles Ives, Olivier Messiaen, Karlheinz Stockhausen and many others. Computers have been used by composers since the 1950s and there are a variety of approaches, such as Stochastic systems and AI.

III. Proposed solution

A. Slappability and Direct Combination

The development of software which is both usable to novice computer programmers and yet sufficiently expressive to allow for complex musical arrangements requires a suitable interaction framework. One interesting candidate is Direct Combination, a framework which derives from a user interface developed for a piece of music composition software.

Daniel Oppenheim initially developed *Slappability* as an interaction framework for his DMIX music composition software (1994). The design allows for any object to act as the source; the user drags the source onto the target object. Once the mechanism is initiated any of the source object's characteristics can be applied to any of the target object's parameters. *Direct Combination* (DC) is a development and generalisation of Slappability by Oppenheim and Simon Holland (1999). DC allows users to select any number of interaction objects and uses this selection to constrain the search space of possible commands. In this way users can indicate which objects they wish to use without having to learn or remember the operations that the system allows. The selection of objects narrows options to those that are contextually relevant. DC is a suitable candidate for the proposed research activity due to its ability to accept input commands in many different orders (S Holland, Gedenryd and Morse, 2002), offering a good match for the spontaneous methodology often used in music composition (Collins, 2005). Users will not have to learn a potentially large number of operations in order to use the software; such operations can be abstract and therefore difficult to recall. The user will be able to select the objects they wish to manipulate and the system will constrain options to show only contextually relevant content.

The research also considers the concept of flow (Csikszentmihalyi, 2014) as it considers the effect a carefully optimised experience can have on the user. It provides a useful framework in which to consider creativity, challenge and absorption. The tradeoff between expressivity and usability (Repenning and Ioannidou, 1997) will be an important aspect.

IV. Methodology

The chosen research method is a mixed methodology using an iterative research by design approach. This method requires

the iterative creation and evaluation of algorithmic composition systems and their user interfaces to uncover issues, sharpen theory, and produce candidate solutions. An iterative series of interaction designs have been developed to implement this approach using expert inspection methods. A text-based DC implementation built in *SuperCollider* has been taken through several iterations. The research will develop a range of prototypes which will be tested using a variety of software usability inspection methods. The results of the tests will provide hypotheses which will, in turn, inform subsequent iterations. This development cycle will repeat and lead to final stage prototypes which will be evaluated with end-users.

V. Initial findings

We began with an assumption, informed by an analytical review, that the software already in the space presents the inexperienced end user with a sub-optimal experience. To clarify this assumption a representative selection of algorithmic music software was analysed. The main findings were that existing software exhibited the following characteristics:

Requires significant user knowledge Much of the reviewed software requires the user to be competent in the design and implementation of algorithms within either text-oriented or graphical programming environments. A selection of software assumes musical knowledge, and some software requires the user to understand both areas.

Use of metaphor The use of metaphor is broadly applied in the area. Graphical programming languages make use of a patch-cable metaphor, while other software reference mixing desks, musical staff notation or sequencer-like piano rolls. Metaphor can be used to infer purpose and to leverage existing knowledge but it can also present a barrier to users with limited experience in the referenced fields.

Imposes a defined working practice Some reviewed software requires users to commit to a particular design which is subsequently difficult to change once the patch has become more complex.

Lack of structural awareness The user is not easily able to define, and subsequently change, the musical structure to be used in the composition. Users are able to create patches which express structure when using one of the reviewed programming languages but this requires significant programming knowledge.

Complex visual design The reviewed software exhibits a wide range of graphical design. Graphical programming languages, using a patching metaphor, allow complex operations but visibility is significantly reduced once the patch's size increases, leading to low juxtaposability. There are also issues with surfacing pertinent information and allowing the user to easily compare several elements.

VI. Future work

The goal of the project is to develop music composition software which is capable and engaging. The nature of music composition means that there are other constraints (such as the harmonic, melodic and rhythmic functionality). Given the problem under consideration a potentially useful match seems to be Direct Combination but, although this gives a principle, there are no guidelines for an application to music. Therefore, in order to find out the extent to which this makes algorithmic music composition engaging and to find out how DC should be applied it is necessary to carry out iterative design-based research.

Pilot software testing A review of candidate software platforms. Initial findings suggest that the software will be developed in *SuperCollider* and will use Open Sound Control to connect the primitives, allowing for future expansion and interoperability.

Research leading to personas The intended user for the software is someone already familiar with music performance or software, and who is not skilled in programming. Matching users will be surveyed to create personas.

Early development of alternative software prototypes Software prototypes will be developed to test the findings from the paper prototyping stage. Initially, multiple potential designs will be developed in parallel and tested using a variety of software usability inspection methods in order to generate rich usability data in a short period of time. Each prototype will provide a hypothesis which will inform the next iteration. It is expected that some loose conclusions, assumptions and claims will be drawn from the variant tests.

Development of the final stage prototype *Software 1* will be the complete candidate software, including all DC functionality. *Software 2* will be the control software with select DC functionality removed. This will allow for the effectiveness of the DC framework to be fully tested and considered.

Empirical summative end-user studies The final software design will be the source of two objective end-user tests. In the first test users will be observed using the software, with data collected on the number of musical constructs created in a given time, and on the complexity of models created. The second test will require a larger pool of users and will be used to validate findings from the first test and to facilitate the collection of an increased amount of objective data.

References

Collins, David (2005) 'A synthesis process model of creative thinking in music composition', *Psychology of Music* 33 (2), pp. 193–216.

Csikszentmihalyi, Mihaly (2014) 'Flow', *Flow and the Foundations of Positive Psychology*, Springer Netherlands, pp. 227–238.

Holland, S, Gedenryd, H and Morse, D (2002) 'Applying Direct Combination to afford spontaneity in pervasive computing', *International Journal of Ad Hoc and Ubiquitous Computing*.

Holland, Simon and Oppenheim, Daniel (1999) 'Direct Combination', *CHI '99*, Pittsburgh, pp. 262–269.

Oppenheim, Daniel V (1994) 'Slappability: A New Metaphor for Human Computer Interaction', *Music Education: An Artificial Intelligence Approach*, ed. by Matt Smith, Alan Smaill and Geraint A Wiggins, Workshops in Computing, Springer London.

Repenning, A and Ioannidou, A (1997) 'Behaviour processors: layers between end-users and Java virtual machines', *Proceedings of the 1997 IEEE Symposium on Visual Languages*, pp. 402–409.