Choosers: designing a highly expressive algorithmic music composition system for non-programmers

Matt Bellingham[1], Simon Holland[2] and Paul Mulholland[2]

[1] University of Wolverhampton
[2] The Open University
matt.bellingham@wlv.ac.uk

Abstract. We present an algorithmic composition system designed to be accessible to those with minimal programming skills and little musical training, while at the same time allowing the manipulation of detailed musical structures more rapidly and more fluidly than would normally be possible for such a user group. These requirements led us to devise non-standard programming abstractions as the basis for a novel graphical music programming language in which a single basic element permits indeterminism, parallelism, choice, multi-choice, recursion, weighting and looping. The system has general musical expressivity, but for simplicity here we focus on manipulating samples. The musical abstractions behind the system have been implemented as a set of SuperCollider classes to enable end-user testing of the graphical programming language via a Wizard of Oz prototyping methodology. The system is currently being tested with undergraduate Music Technology students who are typically neither programmers, nor traditional musicians.

Keywords: music, composition, algorithmic composition, graphical programming, music programming languages, interaction design, user interface

1 Introduction

Algorithmic composition systems are able to support creativity in a number of ways [1] but these tools are generally only available to programmers. We present an algorithmic composition system designed to be accessible to users without programming skills and with minimal musical training. The intention is to enable the manipulation of detailed musical structures more rapidly and more fluidly than would normally be possible for such a user group.

These requirements led us to iteratively develop, via the programming walk-through method [2], non-standard programming abstractions such as the chooser (see Fig. 3) which in a single basic element enables indeterminism, parallelism, choice, multi-choice, recursion, weighting, and looping. The system has general musical expressivity, but for simplicity here we focus on manipulating samples.

The system is designed with the following principles in mind:

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. http://hdl.handle.net/2436/621151.

- Parsimony – a small number of consistent powerful ideas do the work combinatorially;
- Musically meaningful structuring actions are simple and quick to do;
- Both bottom-up and top-down construction are allowed in any combination;
- Affordances are designed for users of varying music computing abilities via progressive disclosure.

The musical abstractions behind the system have been implemented as a set of SuperCollider [3] classes to enable end-user testing of the graphical programming language during development via a Wizard of Oz prototyping methodology. The system is being tested with undergraduate Music Technology students who are typically neither programmers nor traditional musicians. While they may be conversant with some elements of music theory, their background is often as self-taught music producers with experience of making music electronically using music sequencers/DAWs.

In order to illustrate the principles behind the system, we will present its principal elements in turn, starting from the most elementary.

2 The play area and playing samples

The largest central area of the user interface is known as the play area. It is here that the user is able to assemble, audition, and package musical structures. For someone who wishes to work bottom-up (i.e. starting from existing samples), the simplest way to start is to drag samples into the play area. Samples are shown in boxes (see Fig. 1) and can be auditioned by clicking on the box.

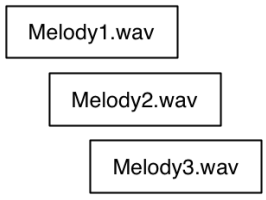

Fig. 1. Samples dragged into the play area.

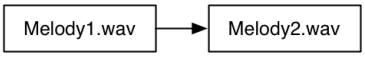

Fig. 2. An example of a simple sequence.

Samples, shown in boxes in the play area, can be assembled into a sequence. A sequence is shown by arrows connecting boxes (see Fig. 2). The direction of the arrow indicates the order of the sequence. Once the first sample has finished playing the second sample will begin. Only a single arrow can enter or exit each element in a sequence. Boxes and sequences can be put inside other boxes, thereby packaging them into a single unit. Anything in a box can be played by clicking on it.

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.

So far, nothing is remotely novel. However, in Section 3 we turn to powerful abstractions called choosers. These have a less familiar set of affordances and are the principal source of power of the new formalism.

3 Soundable choosers

Boxes referring to samples or sequences can be snapped together vertically to create what are known as soundable choosers. Soundable choosers combine indeterminism, choice, parallelism, multi-select weights, looping, and abstraction in a single language element, as explained in detail below.

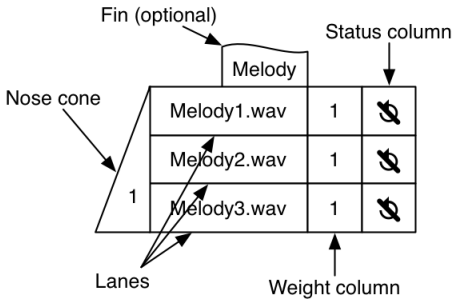

Fig. 3. A soundable chooser showing each element.

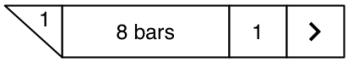

Fig. 4. A time chooser with a duration of 8 bars.

Each box in a soundable chooser has its own lane. If there are two or more lanes in a chooser, the lane will display three additional elements: the nose cone on the left, and the weight column and status column on the right. The nose cone allows the user to specify how many of the lanes will be played simultaneously. For example, if there are fives lanes, and the nose cone contains the number two, two lanes will be chosen to play simultaneously. If the nose cone contains the number one, only a single lane will be chosen. Note that every time the chooser is played, the choice of lanes is made afresh non-deterministically (i.e. at random).

The weight column, shown immediately to the right of the sample name, determines the relative likelihood of a lane to be selected for playback. Fig. 3 shows three lanes each with a weight of one, meaning they are equally likely to be selected. If one lane's weight was changed to two it would now be more likely to be selected for playback. If a lane is given a weight of zero it will never be selected for playback. This can be a useful way of trying arrangement ideas without removing the lane altogether. The status column shown on the far right of each lane allows for control over how the chooser deals with looping and stopping playback, as we will describe in detail below.

Just as samples can be assembled into sequences, so can soundable choosers. The nose cone can be set to zero, which will result in the soundable chooser

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.

being skipped in the context of a sequence. Optionally, the fin can be used to give the chooser a name, such as the chooser named 'Melody' in Fig. 3.

4 Time lanes and time choosers

A time lane is a lane whose purpose is to specify a musical duration (usually in beats or bars or seconds) rather than to specify something to be sounded.

A time lane can be included in what is known as a time chooser (see Fig. 4). Note that the time chooser has a nose cone which contrasts visually with the nosecone of a soundable chooser. Time choosers and soundable choosers have similarities and differences. They are similar in that both make nondeterministic selections from the lanes they contain. However the business of a soundable chooser is to choose one or more lanes to be sounded, whereas the business of a time chooser is to choose a single duration. A time chooser can select only a single time lane or none – depending on the number in its nose cone. It is impossible for a time chooser to select multiple durations at once.

A time chooser can be used alone as part of a sequence – however, when used in this way it will simply result in a rest of the specified duration. A time chooser has more interesting behavior when combined with a soundable chooser, as detailed below.

A soundable chooser can be given a limited musical duration by adding a time chooser to its bottom (see Fig. 5) - note how the nose cones fit together. The duration of the soundable chooser will now be controlled by the time chooser. Together the two types of chooser are referred to as a full chooser, or just a chooser for short.

The purpose of a time chooser within a full chooser is to moderate in a non-deterministic manner how long the soundable chooser and its individual lanes play. Possible interactions between the settings of soundable and time choosers can make the results more varied than might be imagined, as demonstrated in the following example.

Fig. 5 shows a full chooser consisting of a soundable chooser containing three samples, and a time chooser with a duration of eight bars. The status column of the time chooser shows either a soft stop (shown as $>$) or a hard stop (shown as $\times$). In this example, one of three samples will be selected for playback. All samples are set to loop, as shown by the circular arrow. When the duration shown in the time chooser (8 bars) has elapsed the currently-playing sample will be stopped. As the sole lane of the time chooser is set to a hard stop, the sample selected by the soundable chooser will be stopped as soon as time is up. If, by contrast, it were set to a soft stop, the sample would not be stopped at this point – instead the loop controlling the sample would be turned off and the sample would play to its end. Consequently, if the sample was shorter than 8 bars duration, it would loop one or more times before playing to its end after the loop was switched off. If the sample were longer than 8 bars, the entire sample should play just once.

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.

Fig. 5. A full chooser, consisting of a soundable chooser and a time chooser.

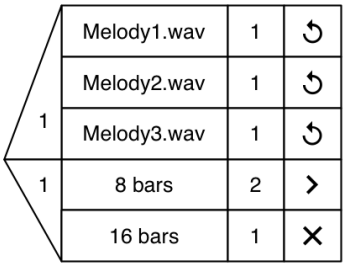

Fig. 6. A chooser with two lanes in the time chooser. Only one will be selected. Note the different weights which make an 8 bar duration more likely, the soft stop for the 8 bar option, and the hard stop for the 16 bar option.

In general, a time chooser's nose cone can be set to either one or zero. If it is set to zero the soundable chooser will run as though there is no time chooser. This allows for quick arrangement changes, with the possibility of infinite playback if the soundable chooser lanes are set to loop. If the soundable chooser is not set to loop the sample(s) will play and the chooser will be released when they have finished playing, regardless of length. There can be more than one time lane in a chooser (see Fig. 6), but only one will be selected when the chooser runs. The selection is made using the same lane weight system as for soundable lanes.

Sometimes it is desirable to ensure that a chooser lane is always selected for playback. Any soundable lane with 'A' in the weight column will always be selected and played. The number in the soundable chooser nose cone is constrained by these mandatory lanes; for example, in the example shown in Fig. 7 the soundable chooser nose cone number can only be three (play all) or zero (do not play any lanes: this can result in a rest if the time chooser selects a duration, or skipped if both soundable and time choosers have nose cone values of zero).

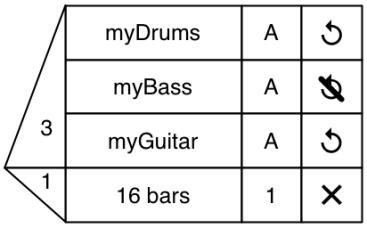

Fig. 7. A chooser with three lanes set to always play.

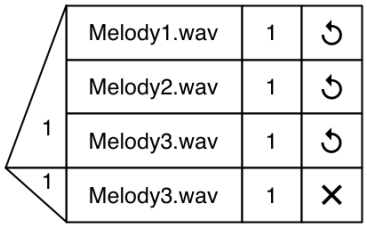

Fig. 8. The duration of the `melody3.wav` sample is used in the time chooser. Note that `melody3.wav` is also one of the samples in the soundable chooser and so it is used for both duration and playback.

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.

5 Setting the duration using the length of a sample

Although the duration in a clock lane is usually specified by a number of beats, bars, or seconds, it can also be specified by a sample. It is important to note that any sample added to a time chooser will not be audible; instead its duration is used to control the time. In order for a sample to be both audible and used to set the duration of the chooser, it is necessary to add it to both the soundable and time choosers. An example is shown in Fig. 8.

6 Boxes and repetition

Boxes can contain any playable (samples, choosers or sequences) or refer to them by name. Note that the example in Fig 9 uses boxes which refer to playables named 'Joe Verse' and 'Joe Chorus'. The system allows the use of such references even when a playable of that name does not yet exist; this allows the user to create placeholders without having to immediately populate them. Importantly, this allows users to work using either a top-down or bottom-up approach. A top-down approach is when the user begins with a high-level outline of the piece (an ABA structure, for example) and then populates those sections, working down to the note or sample level. A bottom-up approach is the opposite: the user begins at the note or sample level and builds musical sections before organising the macro structure of the piece.

Explicit repeats can be added via multiplication notation below playables or references to playables, as shown in Fig. 9 and Fig. 10.

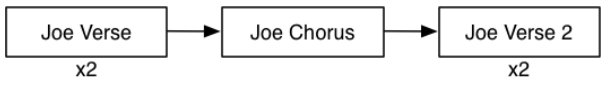

Fig. 9. A simple sequence showing the use of repetition notation.

Fig. 10. The same sequence as shown in Fig 9 but with the duration of each repetition constrained using a time chooser.

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.

7 Nesting and recursion

So far, musical structures have been assembled largely bottom-up via sequencing, looping, choice and multi-choice (possibly weighted), non-determinism and packaging. In this section we see how high level structures can be elaborated downwards using nesting and recursion.

The first column of any lane is actually a box, and consequently can make reference to any other chooser or element. Thus, one chooser can be nested inside another chooser, either visually or by referencing a named chooser. In the example shown in Fig. 11, due to the number three in the nose cone, the parent soundable chooser will always select all three lanes. The top two lanes contain child soundable choosers which will select one of two samples. The `Drums.wav` sample will always play. The duration of the full chooser is controlled by the time chooser which is set to hard stop after a duration of 32 bars. Importantly, in terms of understanding the effect of time constraints on nested choosers, note that in this example the child choosers are not set to loop, while the parent lanes are set to loop. Thus, if the chosen child sample has a duration under 32 bars, the parent lane will loop and re-select a sample until the time chooser's duration expires and playback is stopped.

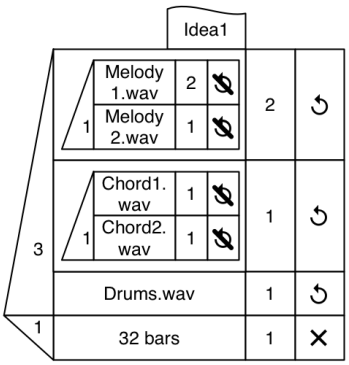

Fig. 11. An example showing two soundable choosers nested inside a parent chooser.

An alternative to visually nesting a chooser is to simply reference a chooser's name (as given in the fin of the nested chooser) inside a lane of the parent chooser. This notation can be preferable to visual nesting in some situations as it allows for the reuse of musical material, as well as minimizing visual complexity. The user is free to use either notation in any combination.

This paper outlines core functionality but, due to space constraints, omits the mechanisms used to create, reuse, and coordinate variables and variable choices. Variables allow the user to easily reuse material (which can be integers, status types, or duration information) by defining a variable once and creating multiple references from that variable to a single element.

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.

8 Related work

In a previous paper [4] we have used the Cognitive Dimensions of Notations framework [5] to review the usability of a representative selection of software capable of algorithmic music composition. We found that most existing software requires the user to have a considerable understanding of constructs in either graphical (e.g Max, Pure Data) or text-oriented (e.g. SuperCollider, ChucK, Csound) programming languages: such knowledge requires a significant learning overhead. Users are often required to have an understanding of musical notation and/or music production equipment such as mixing desks and patchbays. Several pieces of software imposed working practices which were not conducive to compositional processes. In some instances the user was unable to define, and subsequently change, the musical structure. Finally, complex visual design in graphical programming languages led to patches with multiple connections, making them difficult to read and to navigate.

9 Conclusions

This paper outlines a new abstraction, the chooser. The chooser system is designed to meet several needs: to enable our target users to create algorithmic music of arbitrary complexity; to facilitate graphical programming with minimal syntactical concerns; and to make common musical tasks simple.

The work presented in this paper is based on a desire to create simplicity in a user interface for algorithmic music, and to offer consistency and flexibility with an emphasis on enabling musically useful structural changes to be made quickly and easily. While there are a number of programs capable of algorithmic music there is a lack of software designed for non-programmers. It is hoped that the design outlined here will enrich the tools available to composers and musicians.

References

1. Jacob, B. L.: Algorithmic Composition as a model of creativity, Organised Sound, Cambridge University Press, 1(03), pp. 157 – 165. (1996)
2. Bell, B., Citrin, W. V., Lewis, C., and Rieman, J.: The Programming Walkthrough: A Structured Method for Assessing the Writability of Programming Languages; CU-CS-577-92. Computer Science Technical Reports, Paper 554. (1992)
3. McCartney, J.: Rethinking the Computer Music Language: SuperCollider. Computer Music Journal, 26(4) pp. 61 – 68. (2002)
4. Bellingham, M., Holland, S., and Mulholland, P.: A Cognitive Dimensions analysis of interaction design for algorithmic composition software. In du Boulay, B. and Good, J., editors, Proceedings of Psychology of Programming Interest Group Annual Conference 2014, pp. 135 – 140. (2014)
5. Green, T. R. and Petre, M.: Usability Analysis of Visual Programming Environments: a 'cognitive dimensions' framework. Journal of Visual Languages and Computing, 7: pp. 131 – 174. (1996)

Cite as Bellingham, M., Holland, S. and Mulholland, P. (2017) Choosers: designing a highly expressive algorithmic music composition system for non-programmers. *2nd Conference on Computer Simulation of Musical Creativity* – 11th to 13th September 2017, The Open University, Milton Keynes, UK. `http://hdl.handle.net/2436/621151`.