An Interface for Sequencing with Concatenative Sound Synthesis

Lukasz Kuzma

Master Thesis submitted in partial fulfillment of the requirements for the degree:
Master in Information, Communication and Audiovisual Media Technologies

Supervisor: Xavier Serra

Department of Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Spain
September 2008

ABSTRACT

Although there has been a good deal of research in concatenative synthesis, most existing approaches tend to focus on offline composition. The realtime systems that have been developed place emphasis on source pruning and on exploration rather than composition. This paper presents an interface for sequencing with concatenative synthesis with particular emphasis on realtime performance. The interface is designed to address some weaknesses of previous realtime sequencing systems, by allowing the user to seamlessly work at the local level of individual beats as well as the global level of musical passages. The interface deals mostly with the target side of concatenative synthesis by using sound classes, numerical descriptions of sound that group source sound units. The interface exploits the fact that sound classes are purely numerical and provides mechanisms for interpolation, modulation, and stochastic variation to aid in creating and modifying compositions in a performance context.

ACKNOWLEDGEMENTS

This thesis would have been impossible without the generous help of Jordi Janer and Maarten de Boer. The project builds on the BeatMash software and would not make a peep without it. The efforts of Martin Haro and Gerard Roma were invaluable in grouping sounds and thus defining the sound classes on which the project relies. Xavier Serra kindly provided the environment, the most basic premise of all our efforts, where this work could happen.

CONTENTS

1 INTRODUCTION

Although it has many progenitors and has appeared in various guises, what has come to be called concatenative synthesis has experienced a surge in recent years, as the number of projects using similar techniques has grown tremendously [8]. Concatenative synthesis, sometimes referred to as audio mosaicing, generally consists of building a sound, called the *target*, from a database of source sounds, called *units* [5]. Typically, the source database is created by segmenting a stream of recorded audio into pieces of equal length or along demarcations from an accompanying score or using an algorithm that attempts to extract useful pieces. The collection of source clips, or *corpus* [6], is then analyzed and classified according to a set of descriptors, which may either be measured or user-attributed. These descriptors are used to select matching units from the source collection with which to replace segments of the target. Occasionally, further processing is used to match the target or to smooth transitions between chosen units.

This general method is extremely flexible and has been approached from many angles [5] enabling a wide range of musical applications [9]. Audio can be interactively synthesized, rearranged, or filtered. Timbre can be altered gradually or completely, instruments can be morphed, and musicians can interact with their own recordings.

For real-time interaction and sequencing, most of the implementations currently in use focus on pruning the space of available source units. In part this is because live or lengthy audio streams are used as the target. Due to the looping nature of some contemporary music, such as techno or electro, target specification can be totally predetermined. Given this fact, it would be interesting to attempt a target-oriented approach to concatenative synthesis.

In addition, segmentation and analysis can utilize a large number of descriptors. Attempting to prune the source corpus requires a user to navigate through a high-dimensional descriptor space or through a limited lower-dimension projection of the

space. It may be useful to attempt to combine descriptors into higher order groupings to allow for more intuitive and musical control.

Finally, concatenative synthesis offers a number of opportunities that may yield paradigmatic changes in real-time sequenced performance. If target sounds are specified symbolically, the fact that this specification is purely numerical can be exploited, as it readily lends itself to manipulation through interpolation, modulation, and stochastic variation. These manipulation methods are of course available with more traditional synthesis approaches, but they become arguably more powerful here. Rather than, for example, changing a parameter like filter cutoff, many aspects of the synthesized sound can be controlled at once. This is similar to sequencing the parameters of a synthesizer, but with concatenative synthesis more features of a sound can be immediately controlled. The user can affect the drums, bassline, and melody all at once, with a single parameter. Thus, we have a situation reminiscent of that of the DJ, where some degree of control is given up in order to gain a higher-order immediacy.

This thesis presents an interface that utilizes a lexicon of sound class symbols to sequence music. This application has been designed with usability in a live performance context as a major goal. Moreover, the application attempts to exploit the features offered by concatenative synthesis to extend its functionality beyond that of more traditional approaches. The paper begins with an overview of the current state of concatenative synthesis. This is followed by a history of sequencing approaches that is intended to contextualize the approaches being employed here. A detailed description of the design and implementation of the interface that has been developed comes next, and this is followed by a discussion of the novelty and usefulness of the approaches developed. Finally, some possible future extensions to the work are described.

2 CONCATENATIVE SYNTHESIS

2.1 History

Taken in its broadest sense, concatenative synthesis has been developing since the days of *Musique Concrète* when tapes of musical recordings were first cut apart and spliced back together. Initially, sounds of instruments could be interspersed with those of speech or machinery. Eventually, with the looped tapes employed by the Mellotron, instruments and vocals could even be simulated with loops made from live recordings. These analog techniques continued to be employed until digital devices were finally capable of recording and playing back sounds. We have since witnessed a huge proliferation of musical applications of this technology, from sampling to granular synthesis to music information retrieval.

During the last decade, there has been a concentration of interest in what has come to be called concatenative synthesis or audio mosaicing. Like sampling, concatenative synthesis lies between the extremes of granular synthesis on one end and song identification and retrieval on the other. However, unlike traditional sampling, which also works at this time scale, concatenative synthesis techniques analyze and classify the sound units used, and generally the process of putting the sound units together is algorithmically rather than manually controlled.

2.2 Current Implementations

Quite a large number of implementations of concatenative synthesis are described in [1] and extensively classified in [4] and particularly [5]. Here we will constrain our focus to the more relevant real-time systems and compare their methodology and consider some of their drawbacks.

2.2.1 MoSievius

MoSievius, "a framework intended for the rapid implementation of different interactive mosaicing techniques" [2], allows a user to interactively specify both the

source of sound samples as well as the target sound to be resynthesized. Target specification occurs in real-time by analyzing an incoming audio stream. As an example, the authors describe the sounds from a live trumpet being used as input and being replaced by samples taken from a trombone.

Analyzing the signal in real-time leads to latency that is at least as long as the target segment being considered [2]. The authors propose two methods to surmount this. One method is to keep using a source segment until the incoming sound differs sufficiently from the segment, whereupon a new segment is chosen. The second solution proposes the use of segments so short that the latency is no longer perceptually significant. The authors suggest that this method requires more processing of the source slices in order to smooth out the transitions between them.

These problems hinge on the inability to know the target specification wholly, since it is being analyzed as the input occurs. Therefore, the authors propose focusing real-time interaction on the source instead by filtering the set of source segments being utilized at any point. Their implementation, the Sound Sieve, is a means of limiting the source sounds by setting thresholds in their descriptor space. The user can interactively set these thresholds, effectively creating a path through the space over time. In this way, the target can be resynthesized using only sounds with chosen qualities.

The authors further suggest using the Sound Sieve to filter a specified target and then concatenating the results. This way, the target sounds are themselves used as the source, thus portions of the target are removed. Time stretching can be used to fill the resulting gaps, thereby preserving the original temporal structure. This method allows the Sound Sieve to act as a filter that acts on higher order descriptors, like RMS energy or pitchiness.

2.2.2 CataRT

The CataRT system [8] is a collection of patches for Max/MSP that perform concatenative synthesis and, like the Sound Sieve, allow the user to interactively

specify a subset of the source corpus. A high-dimensional descriptor space is projected onto two dimensions, and source clips appear as points in this space. The user can interactively position a point in the space and specify a threshold radius around this point, thus marking a subset of the available sounds.

The actual triggering of source sounds is accomplished in several ways. First, there is a series of mouse-driven modes: bow plays units whenever the mouse is moved, fence plays a unit whenever a new unit becomes closest, beat uses a metronome to trigger units, while chain simply plays a new unit as soon as the previous one is finished. Furthermore, CataRT includes a looping sequencer that lets the user specify one envelope to control descriptor value and another to control descriptor weight. There is also a drumbox mode in which descriptors can be sequenced along a fixed number of beats, with a sound class assigned to each beat. Within each sound class, the descriptors can be edited or modified in real-time.

2.2.3 Granuloop

Similarly to MoSevius and CataRT, Granuloop [10] lets the user interactively navigate the source corpus. In this case, the source is provided by four beat loops that are chopped into 32 units each. The energy and spectral similarity of the units is calculated. From this data transition weights between units are calculated in turn.

The four loops are placed at the corners of a square, and the user movers a cursor around the square, thus controlling playback through its proximity to each of the loops. At each time slice, a unit from one of the four loops is chosen based on its match with the current slice of the closest loop and weighted by the position of the cursor. Thus the distinction between source and target is far less significant than in the other systems examined. The elements of each loop are simultaneously being used as potential target and source units, with the probability of a unit being played determined by the cursor.

2.2.4 BeatMash

The BeatMash project uses a library of sound loops that are cut up into beat-length segments. Any loop can be selected and used as the target. A number of other loops can be loaded, and the segments of these loops become available as the source units. As above, the most similar units of the source loops are used to replace target units. Each loop has a similarity fader that allows the user to interactively control the weight assigned to its similarity. By using these faders, the user can influence the source choice interactively and control the occurrence of a given loop's segments in the mix. As in Granuloop, the distinction between source and target is blurred, because any of the loaded loops can be chosen as the target. However, there is still always a single selected target.

2.3 Discussion

The focus of the systems described has largely been on manipulating the database of available sounds. There has been some work on specifying the target using descriptors, as in the CataRT drumbox mode, or by using live input, but target manipulation has not really been the central concern. Arguably, more work could be put into manipulating the synthesis target. For instance one could utilize a target space, the bounds or axes of which are constituted by intended targets. This way, one could move about the target space, interpolating between targets in real-time. For loop-based music, this would be a quite interesting way to perform. Furthermore, by using loops many targets can be known in their entirety, avoiding some of the latency problems discussed above and in [2]. In addition, target descriptors for the entire target or for target segments could be adjusted in real-time, adulterating the actual loops. At the extreme, targets could be entirely synthetic, consisting purely of points in descriptor space sequenced in time as in CataRT.

By using live input, or navigating the descriptor space in real-time, the existing systems focus more on exploration than on deterministic sequencing. It is possible to create loop envelopes with CataRT, but there is no way to sequence longer pieces,

modulating various descriptors together. This leaves a lot of open possibilities with regard to sequencing systems geared specifically toward concatenative synthesis.

More work can also be done to create meaningful groupings and scalings of descriptors. One of the problems encountered by CataRT is that the corpora are "unevenly distributed over the descriptor space" [8]. Also, only two dimensions of descriptors can be navigated at once. This may not be optimal from a musical standpoint. It may be useful to combine descriptors into higher-order groupings that can be more musically meaningful.

Finally, as suggested in [2], symbolically specifying the target could be quite interesting. Rather than having users move around in a single plane of descriptor space, one could create a symbolic, musically significant lexicon with which to specify target parameters. Symbols could correspond to single descriptor values or points (or ranges) in the descriptor space. They could even be combined in various ways over time. Descriptor values could be made to change at each musical frame, or they could be made to interpolate.

3 SEQUENCING PARADIGMS

3.1 Analog

Some of the earliest sequencers were analog devices that produced a specified voltage at each of several steps. There were typically 8 or 16 steps represented by a bank of knobs or sliders. A clock looped through the steps and the devices would produce a voltage dependent on the position of each knob or slider. With such a device we have only very local control over a single line of notes in a bar or two. This has in fact been used to great effect in minimal music ranging from contemporary work like that of Philip Glass to electronic dance music, particularly Trance. The slight modulations to a repeating pattern that can be performed using this sequencing method can be considered dull and repetitive to some, but to others they are hypnotic and trance-inducing.

One limit here is that it is difficult to simultaneously control more than one dimension. Pitch, for example, can be easily controlled, but simultaneously manipulating pitch, duration, and some timbral parameter would be extremely difficult, simply because of the number of knobs or sliders requiring simultaneous attention. Furthermore, performing sudden changes that affect all steps at once, like changing from a verse to a chorus, again might require the very rapid setting of sixteen sliders, which, depending on tempo, could be quite a feat. Finally, as with all aspects of the old analog synthesizers, storing and recalling a pattern was a matter of much manual labor, and again, was not something that could be rapidly performed.

3.2 Digital Pattern

The first digital sequencers did not stray very far from their analog predecessors. They initially also simply had note and possibly velocity values assigned to a number of steps. Here, however, it was much easier to store and recall patterns, and thus to make rapid performance changes that were difficult with the analog machines. This gain was unfortunately offset by the loss of tactile control and visual feedback. The famous Roland 808 and 909 drum machines had dedicated buttons for every step, but subsequent generations of sequencing machines did away with these. Thus, the immediacy of a bank of sliders was replaced by numbers or dots on a liquid crystal screen, buried behind menus, rendered half-inaccessible by membrane buttons.

One could however now perform by changing from pattern to pattern, or even by using pre-arranged groups of patterns. One could in fact simply play back entire MIDI songs, giving performance an odd, charlatan air. The machines had relegated performance to the role of sideshow, if only by limiting the scope of the music to rigid patterns etched on grids. Because the programming of these devices had become obscured by their interfaces, programming in real-time while performing was no longer really feasible; one could only switch from loop to loop. Thus, local control had been lost, and beyond the ability to change the sequence of loops, there was little global control as well.

3.3 Digital Timeline

With the increasing affordability of computers, sequencers became software, and these were typically aimed at writing compositions of longer duration, like entire songs, rather than just little two bar patterns. With this paradigm, performability was no longer considered and had become near impossible. People could still "perform" by either playing back pre-sequenced music, or by selecting portions of pre-made sequences for playback. Even this selecting-as-performance was cumbersome and the effectiveness of it is quite questionable.

It is in all likelihood no accident that this time saw the rapid rise of the DJ as performer. The DJ could play entire compositions with effective arrangements of passages already sequenced, i.e. recorded. Rather than really playing this music, the DJ would add a live component, by rapidly crossfading between songs, by scratching records, by tweaking the mixer's equalizer. Thus, the DJ lacked a large degree of control, but could play pre-recorded, complex compositions with builds and breakdowns calculated to keep a crowd moving, and could provide some visual spectacle with a handful of tricks and changes to the whole mix.

3.4 Digital DIY

This period also saw the rapid rise of DIY performance tools. With no off-the-shelf performance software available, many musicians turned to software that allowed them build their own. The MAX/MSP platform was quite popular, and myriads of interfaces ranging from the simple to the positively bizarre began to crop up. Generator was used to great success, with many hybrid designs based on the analog machines and XoX boxes described above.

3.5 Digital Performance

Finally the industry took notice, and Ableton released its Live software, geared, as the name suggests, toward live performance. Live in many ways harkens back to the pattern juggling proffered by the second generation digital devices described above. While offering little more in terms of sequencing paradigm, its success arguably lies in its interface. The performer can very rapidly manipulate scores of patterns one after another on a large display.

The one perhaps paradigmatic feature new to Live was the introduction of queueing; rather than being turned on immediately, patterns could be queued up to play on the next down beat, or on the next measure evenly divisible by some chosen time step. Thus, quite drastic changes could be preset, allowing the near-real-time creation of phrases, and the ability to shift between compositional passages.

3.6 Discussion

From the foregoing, it should be clear that despite numerous technical advances, performance aided with sequencers can still be greatly improved. New applications can address temporal constraints, manual control limits, and the ability to control audio on a small as well as large time scale. Because of the live nature of performance, low-level editing is difficult, however the greater the degrees of low-level control available, the more potential there is for an interesting performance. At the same time, the ability to sculpt the high-level evolution of musical passages is desirable if the performance is to move beyond extremely repetitive compositions.

4 DESIGN

This project begins with a symbolic lexicon for target specification, and aims to provide an interface that can be played in a performance and can be used to create song-length compositions. It attempts to allow seamless and often simultaneous interaction across several levels. To this end, a circular motif was chosen and used

throughout the application. The loop is, of course, circular, and, as in many approaches above, the loop is the basic unit of composition. The application adds the ability to quickly and easily zoom between levels, and to be able to see lower-level details even while working at a higher level. Finally, while the loop pattern has been retained as the most basic sequence unit, the methods of manipulating the loop are novel.

4.1 Loop Level

Interaction with the application begins at the Loop Level {fig. LL1}. Here the user can construct and modify an individual loop. As in a traditional step-sequencer, a loop consists of 16 steps. Unlike with a traditional step-sequencer, the steps are arranged in a circle, in order to best portray the circularity of the loop. When interactively modifying loop phase, as described below, the circular arrangement becomes extremely helpful in illustrating the loop's behavior.

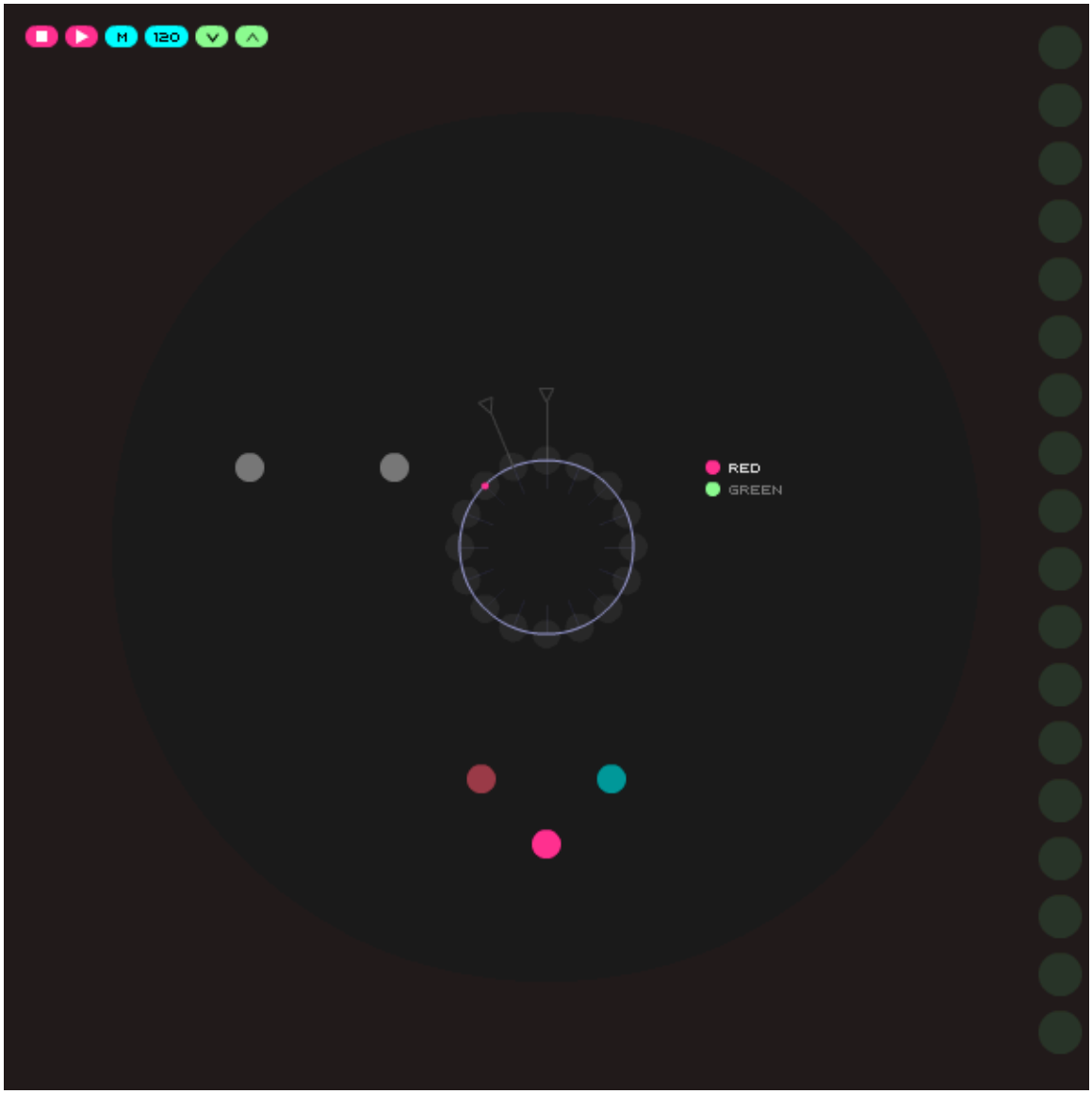

figure LL1

To the right of the loop is a list of available sound classes. The user begins by selecting one of the sound classes by clicking on it. Once a sound class is selected, clicking a position in the loop places the sound class at the corresponding loop step. Hovering over a loop step that has been assigned to a sound class reveals a delete button that allows the user to remove the sound class from the step {fig. LL2}.

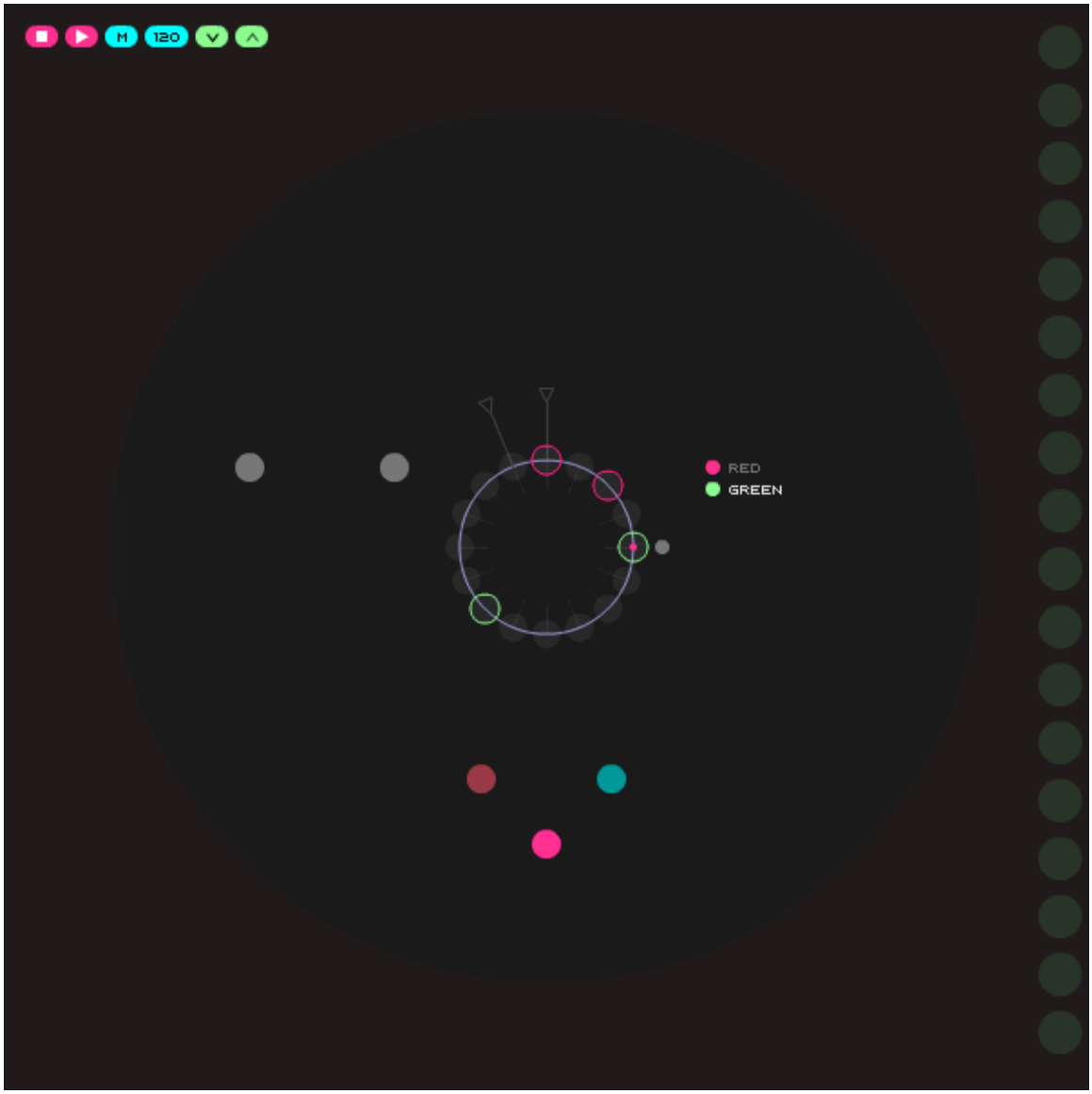

figure LL2

When the sequence is running, a small pink circle indicates the current clock position. Thus, the circle steps through the sequence going clockwise through all of the steps acting as a "play head". If a step has been assigned a sound class, a sound from that class is triggered when the clock indicator reaches that step.

Mousing just outside of the sequence area reveals the length adjustment bezel {fig. LL3}. Clicking on the bezel sets the final loop step to the step closest to the click position. Thus, clicking at 9 o'clock makes the 13th step the final step of the

sequence. After the "play head" reaches this position, it will skip the following steps and begin at the first step again.

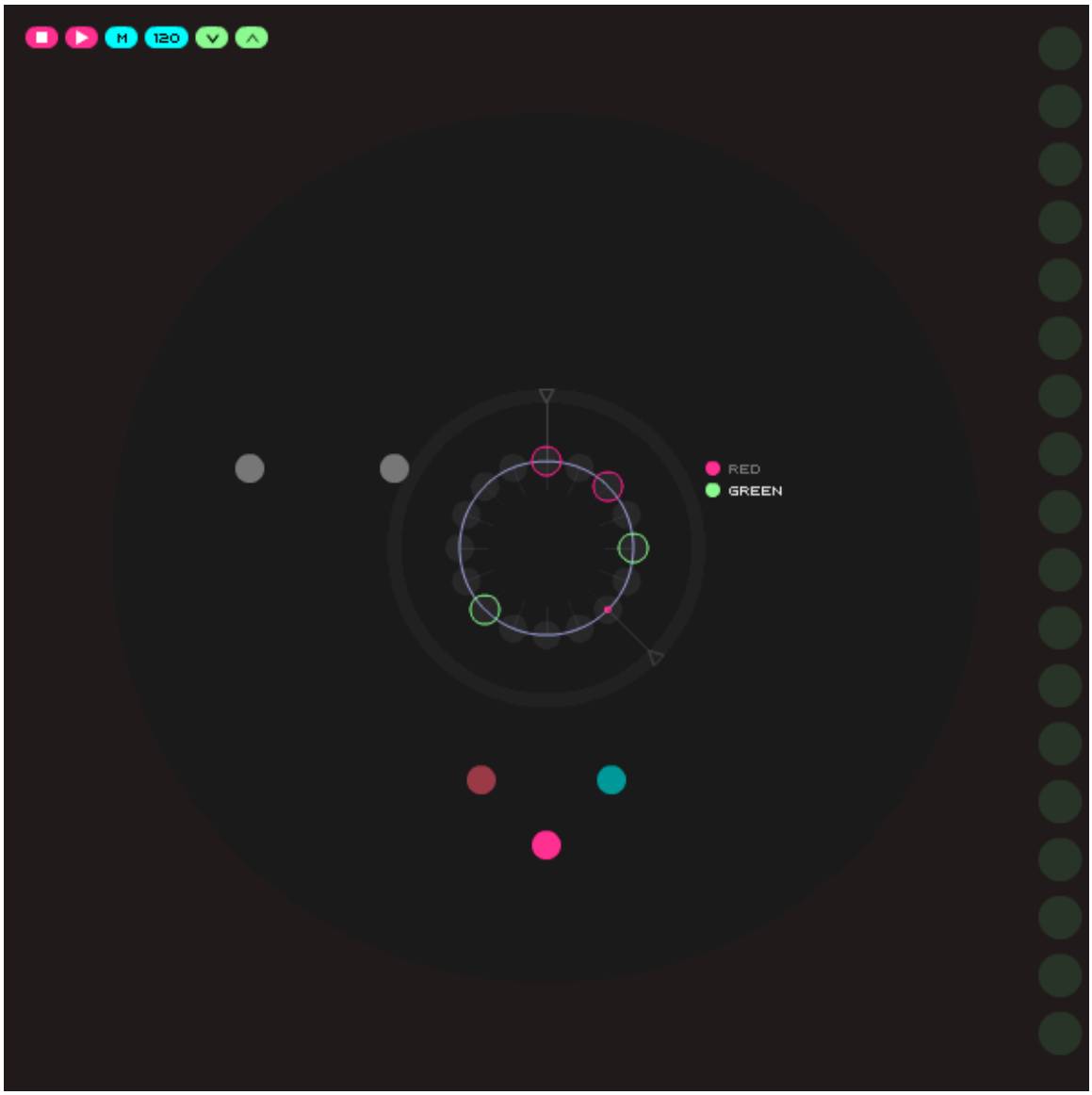

figure LL3

Holding the mouse button down over the center of the loop reveals the phase adjustment knob {fig. LL4}. Dragging this knob up or down rotates the position of all of the sequence steps, essentially altering the phase of the loop with respect to the master clock and any other loops that might be running. The addition of other loops will be described in the Composition Level section.

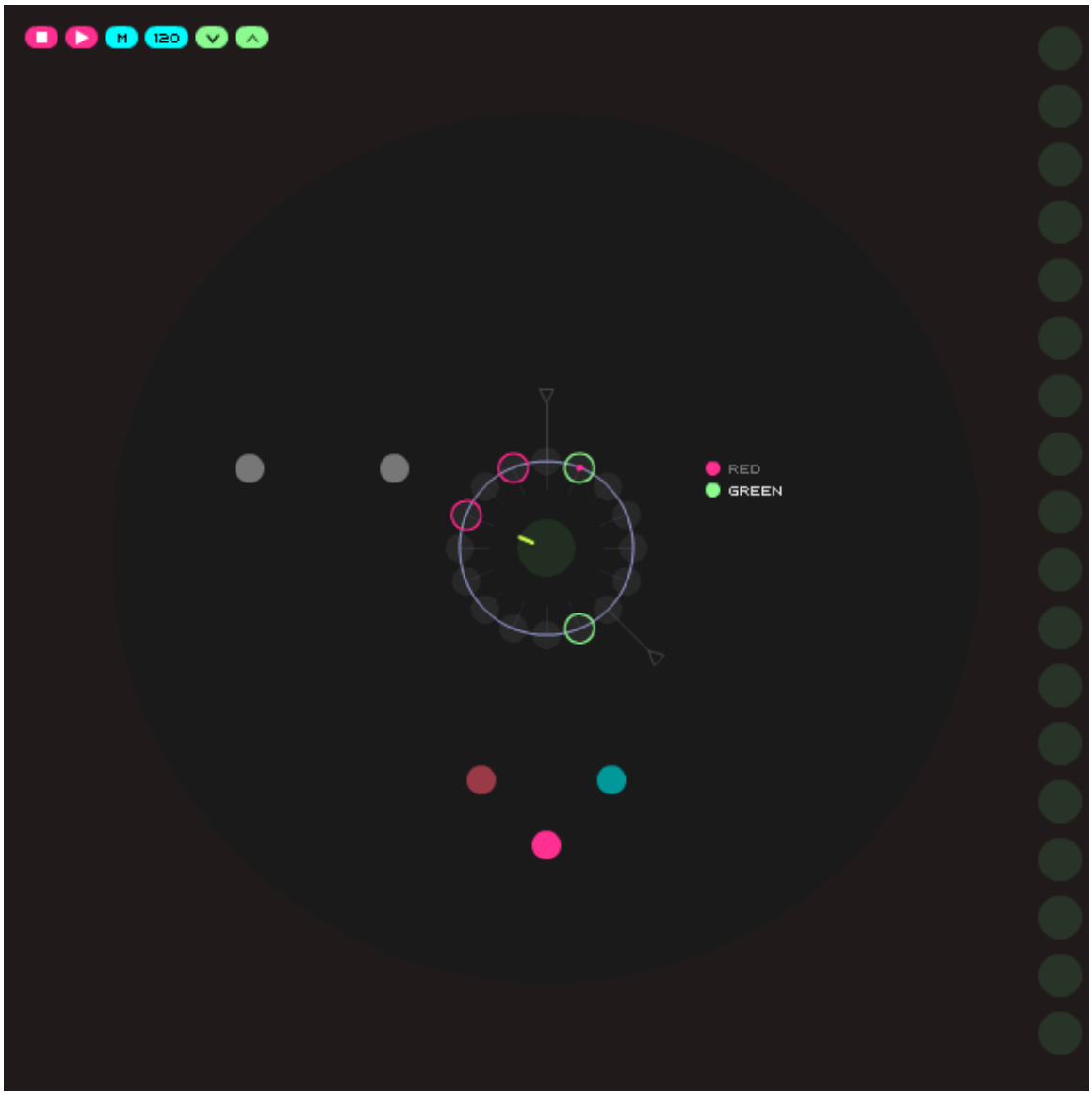

figure LL4

Below the loop sequencer are Mute, Solo, and Delete buttons. The red, leftmost button is the Solo button, which, predictably, puts the loop into solo mode, turning off playback for any other loops that are not also in solo mode. Several loops can be soloed at the same time. Alternately, the Mute button mutes the given sequence. Turning off solo mode is done by either clicking the Solo button again or by clicking the Mute button. Similarly, turning off muting is accomplished by either clicking the Mute button again or by clicking the Solo button. Finally, the bottom-

most button is the Delete button. Clicking this button deletes the current loop, and, if done at Loop Level, zooms the application out to Composition Level so that new loops can be made or other existing loops can be accessed.

To the left of the loop area are Save and Close buttons. Clicking the Save button saves a copy of the current loop in the Sequencer Bank, described in its own section below. Alternately, the user can drag the Save button to a specific position of the Sequencer Bank. Clicking the close button zooms the application out to the Composition Level. Conversely, clicking any loop step that has been assigned a sound class zooms the application in to the Browser Level.

4.2 Browser Level

Zooming in to the Browser Level reveals the individual sound files contained in a sound class along with controls that allow class attributes to be modified {fig. BL1}. Here, the user can constrain the sounds being played when a sound class is triggered. By default, the sound closest to the center of the class is played back. This center is obtained by averaging all descriptor values for all sounds in the class. Euclidean distance based on descriptor values is used to determine distance between sounds and the distance of sounds from center.

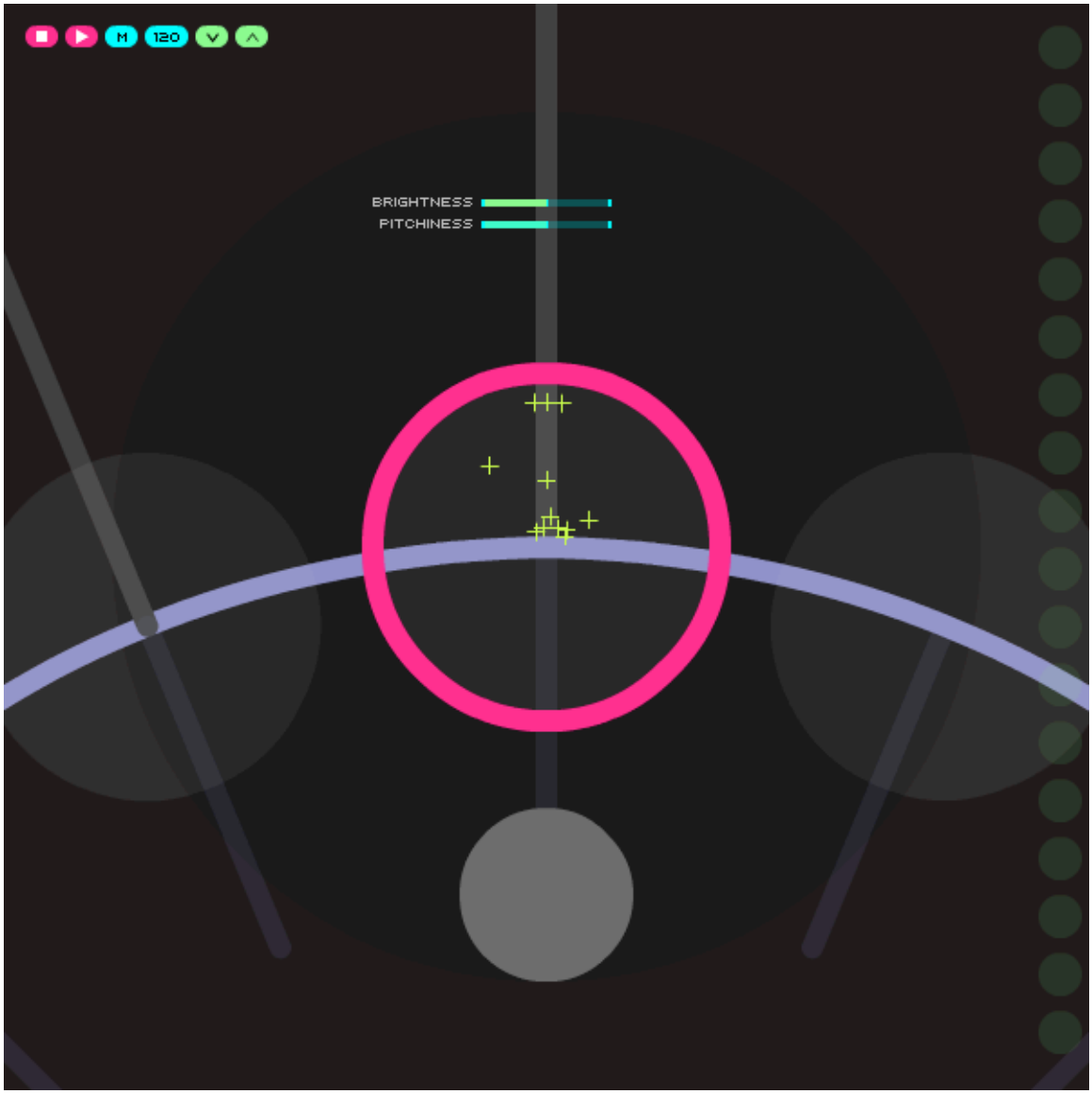

figure BL1

Clicking on individual sound files allows the user to audition them, while a CTRL-click selects an individual file for playback {fig. BL2}. When an individual file is selected, that file plays when the class instance is triggered. In this mode, the application behaves like a traditional sequencer triggering samples.

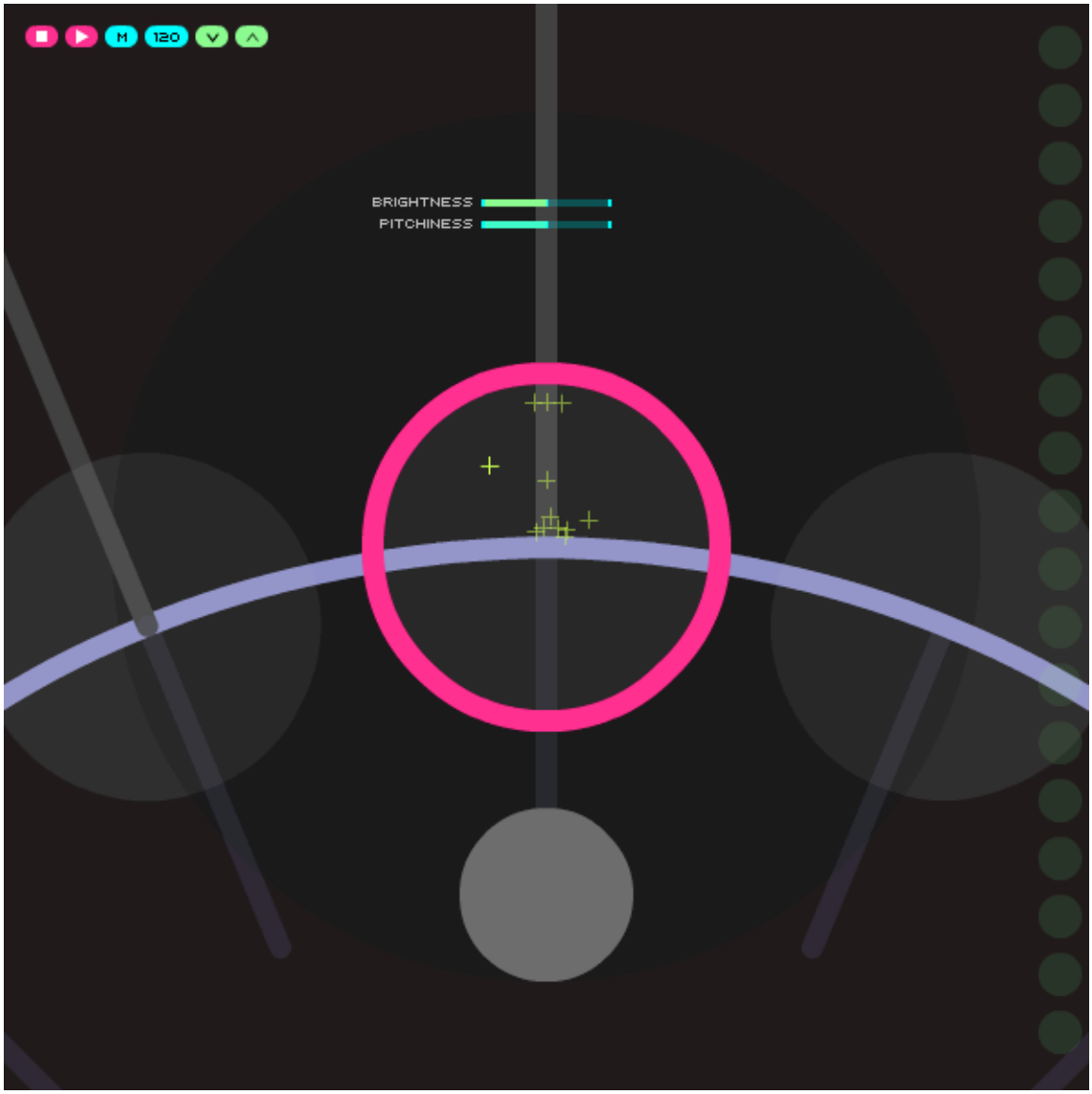

figure BL2

A number of attributes can be modified in a sound class instance. Attributes are higher-level parameters that map to single or, more typically, multiple descriptors. An attribute, such as "brightness", may map with different weights to several descriptors, thereby providing the user an intuitive description of lower level information and a way to manipulate multiple descriptors at once. An attribute is modified by clicking in the area next to the attribute name. This modification changes the descriptor values sent to the audio engine, and so different sounds in the class are triggered.

4.3 Composition Level

At the Composition Level, all active loops can be seen and manipulated {fig. CL1}. The Composition Level consists of a Stage upon which we see active loops. As with the other levels, we can see the Sequence Bank, but at this level we can interact with it more fully. At the Composition Level, loops stored in the Sequence Bank can be dragged onto the Stage and thereby added to the current composition.

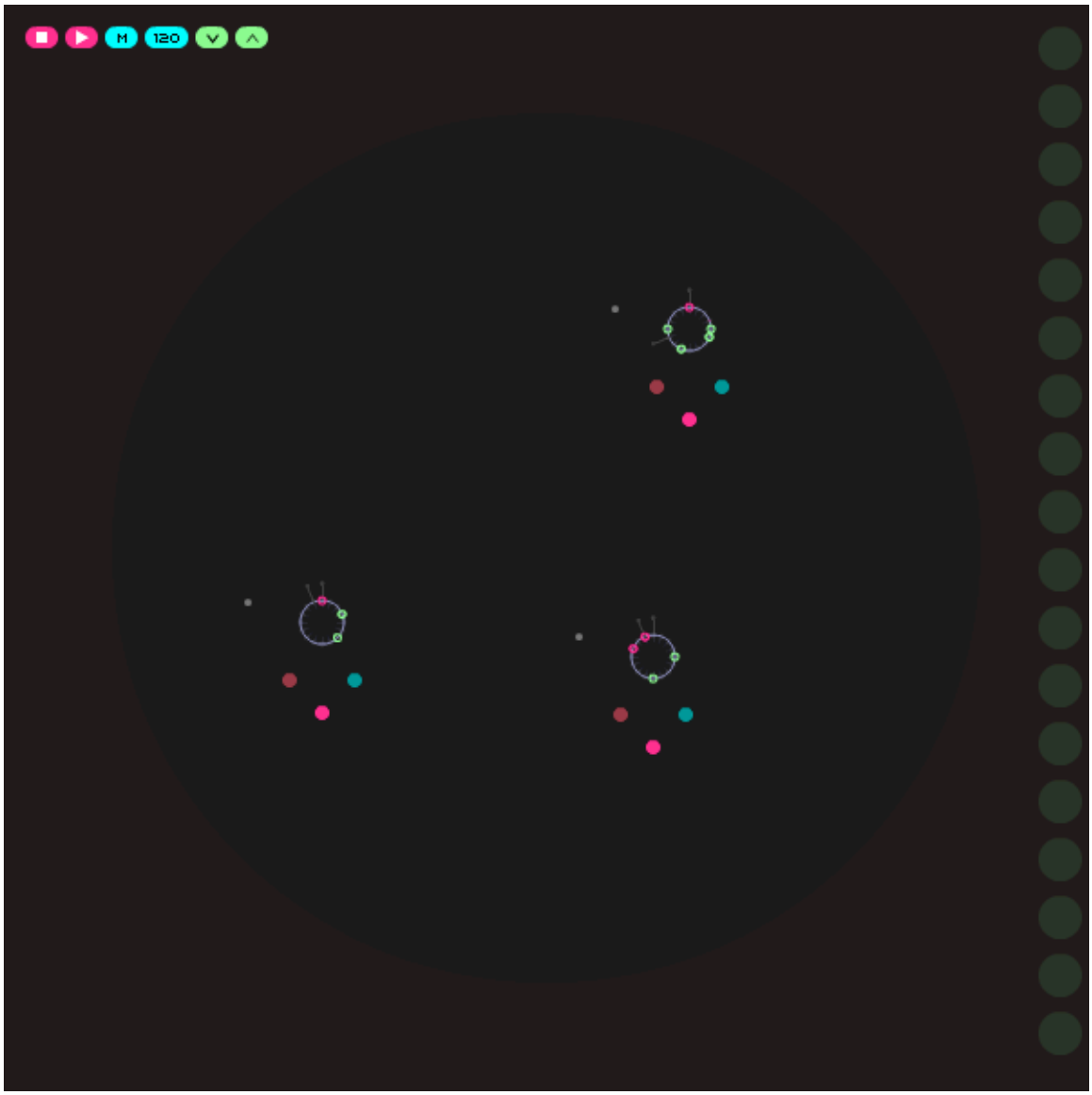

figure CL1

A number of the controls described in the Loop Level section are available here. Furthermore, each loop has a hit area revealed by rolling the mouse over the loop {fig. CL2}. Loops can be moved around the Stage by dragging them by their respective hit areas. Clicking the center of the loop zooms into Loop Level, allowing the user to manipulate the loop more closely. In the current implementation of the application, the distance of a loop from the center determines its influence on the sound being output. That is, if multiple loops are placed on the stage, and more than one loop has a sound class assigned to the same sequence step, a weighted average of the descriptors of both both classes is sent to the audio engine. This allows the user to interactively blend sequences and adjust the interpolation by moving the loops around the stage.

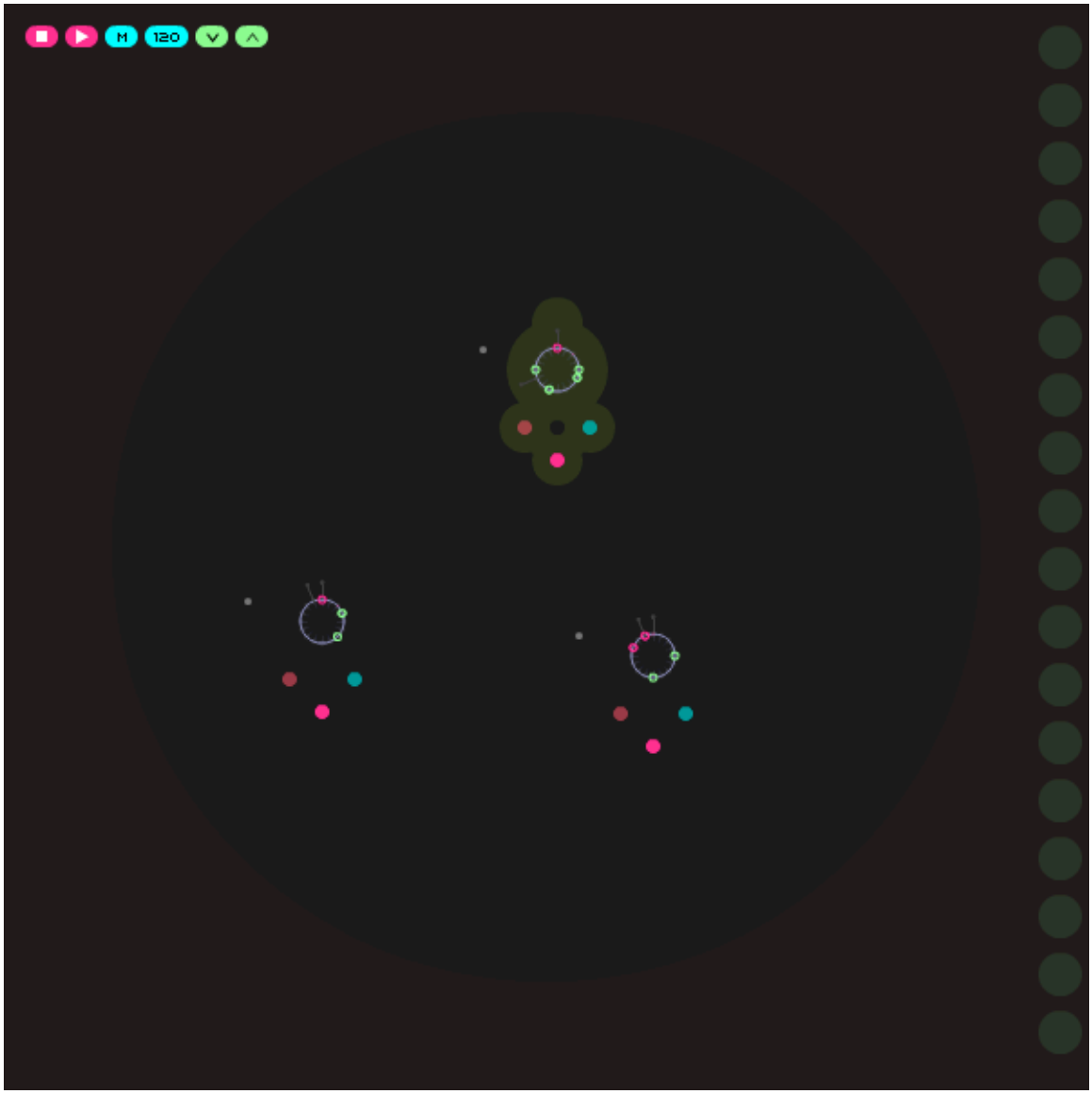

figure CL2

Finally, the design features a gesture drawing feature described in depth in the Future Work section below.

4.4 Global Controls

The application has a control area that is accessible globally. It includes standard transport controls: Stop and Play, which control sequencer playback, a Master/Slave toggle, a tempo control, and Save and Restore buttons.

By default, the application runs using its own clock. The application tempo can be changed by clicking in the tempo area and typing in a new tempo and subsequently clicking elsewhere. The tempo must be between 60 and 240 due to limitations of the audio engine. By clicking on the Master/Slave button, the application can be put into slave mode, disabling its internal clock. In this mode, sequences advance a step when the application receives an appropriate OSC command.

The Save button stores the application state, including all loops and loop parameters, into a Local Shared Object (LSO). This is a proprietary format used by Flash to store local data. Pressing the recall button clears the stage and loads the content of the last saved LSO. This mechanism is quite limited, but can easily be extended to allow for multiple files, and it can be adapted to a client/server model by sending the same data object using, for example, AMFPHP to be serialized by a server.

4.5 Sequencer Bank

The Sequencer Bank can be used to store and recall loops within the context of a composition {fig. SB1}. Each loop has a Save button that can be used to place it in the Bank. To recall a loop from the bank, the user can simply drag it from the Bank area to any position on the Stage. The entire Sequencer Bank is saved and retrieved along with the composition.

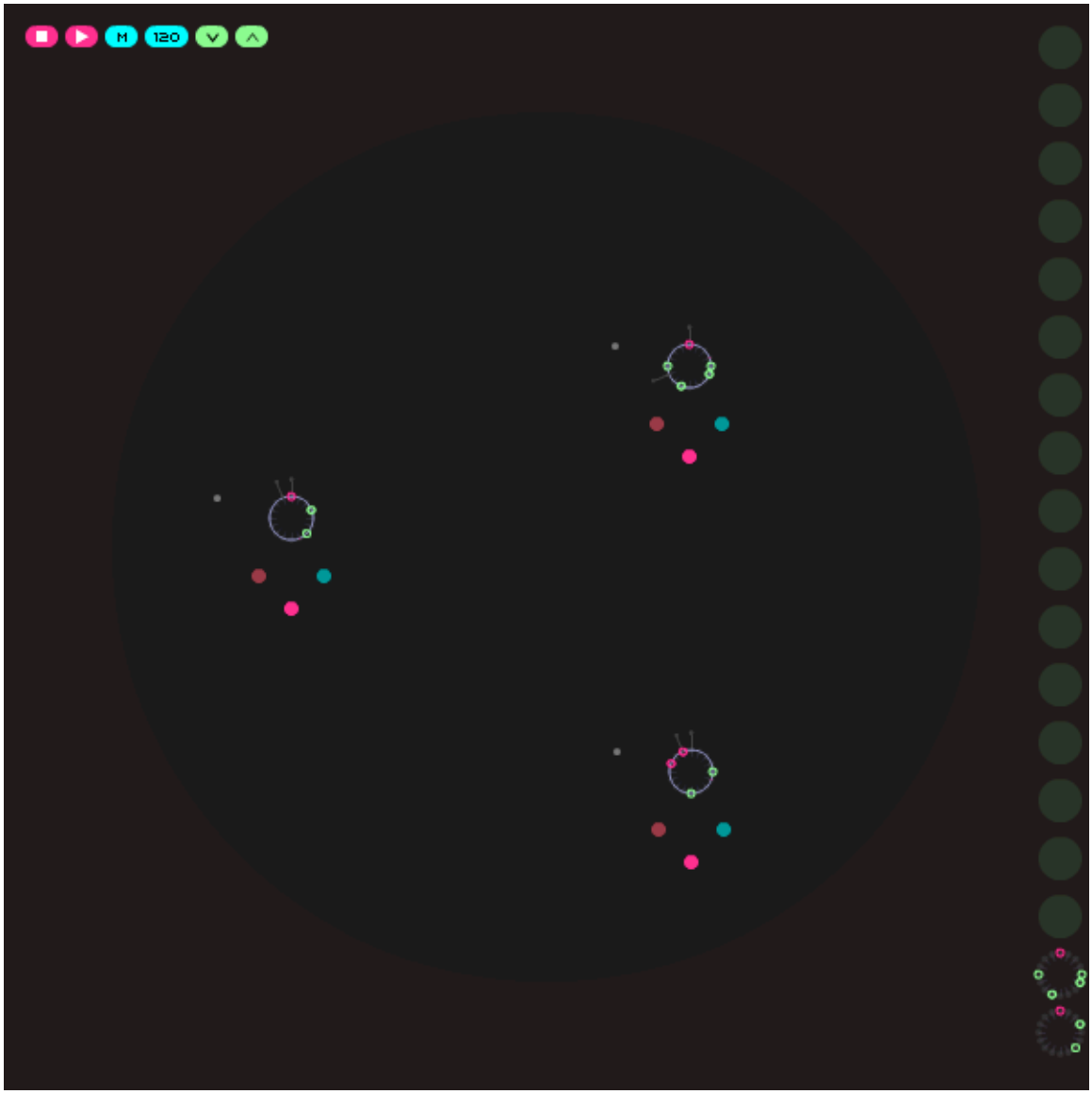

figure SB1

5 RESULTS

At the Composition Level, the gesture drawing feature will give the user the ability to very quickly and easily map out the global characteristics of a composition or performance. The music's evolution can be specified over a long time frame, typically dealing with durations on the order of 32 bars. This allows the performer to focus on local aspects of the composition, like adjusting a single step of a particular loop while global changes are taking place. Furthermore, loops can be manually moved around

as the composition is playing, thus allowing for instant or gradual departures from the score implied by an existing gesture.

At Loop Level, the application presents an interactive step sequencer, with all steps immediately accessible as in the XoX machines of old. This lets the user quickly and easily create and modify patterns like one would on an 808. The application, however, adds the ability to quickly move between different loops, to have the visual, immediately readable representation of many loops that can additionally be faded in and out or stored and recalled. The very structure of the loops is visually represented, so instead of deciphering loop content from a name or a number, the user can clearly see the loop's step pattern.

Two of the more novel features of the application are the length and phase controls. The length control lets the user alter a loops length as the loop is playing. This can make for interesting realtime rhythmic variation, both when a loop is playing alone as well as in polyrhythmic combination with other loops. The phase control allows the user to effectively alter the stepwise phase of the loop with respect to the global time, altering the loop's temporal position. Thus, a loop can be shifted against the rhythm, again allowing for a lot of realtime variation. Naturally, both controls can be used in conjunction to great effect offering endless beat-juggling possibilities. Of course these controls have been implemented in other programs, but the circular design employed in the application makes the phase and length settings easier to set and to read at a glance and does not require the existence of extraneous bars. For example, in Live one can set a loop using start and end locators, but to adjust the phase, one has to drag the locators across the loop. Without the circular design, one needs an extra measure before as well as after the loop being manipulated so that these locators can slide past the loop's end.

The Phonon Level gives the user perhaps the most fine-grained control of the application. Here the user is able to specify with some precision what sounds are actually played when a sound class is triggered. The user can force the application to only play a specific sound file at each given step, thus acting like a standard

sequencer triggering a sampler. More typically, however, a range of sound files is selected. Sound files within the range are then triggered randomly or based on attribute settings, as the application further adds attribute sliders that allow realtime or automated control of sequence parameters. These could be compared to the knobs on an XoX beatbox used to modulate the timbre and duration of a given drum sound. However, they work at a higher level, affecting not just one instrument, but potentially many instruments at once. Here we see the aforementioned situation of giving up some control in order to work at a coarser grain as a DJ might. Because the attributes can be sequenced using LFOs or envelopes, the user can, again, set up musical parameters that evolve over a long time scale.

Because at all levels the user is usually dealing with classes rather than specific samples, compositions may have a further degree of freedom. By allowing some stochastic variation in the selection of sound files within a class, the application can be used to sequence the overall contour of a composition. The composition, however, might use different, though similar, sounds every time it is played. In a live context this can spice up the experience as even a single loop can play with endless variation. The method is also well suited for video game music and film scores, where a certain mood might be created and made to last during a scene or level, without the need to actually compose the entire duration. The music could retain its feel, while at the same time exhibiting automated variation.

6. FUTURE WORK

A number of improvements on and extensions to the application are currently planned. Perhaps the most significant is the addition of the gesture control system already being the developed. As mentioned above, the distance of a loop from the center of the Stage area determines its influence on the sounds being played. Rather than remaining fixed, however, the position of this point can be made to move around the Stage area over time. A user could draw a path on the Stage, describing the temporal evolution of the point {fig. FW1}. These gestures will lend the application a quite high-level dimension, since users will be able to draw gestures

that determine the global behavior of the music being generated. The gestures can be drawn in a freeform manner, or they can be constructed from color-coded lines {fig. FW2}. Using color codes to represent different durations would separate the time dimension from the descriptor dimensions determining degrees of interpolation between loops. Finally, these gestures can be saved and recalled in a stack just as the loops presently are, giving the performer a really broad palette.

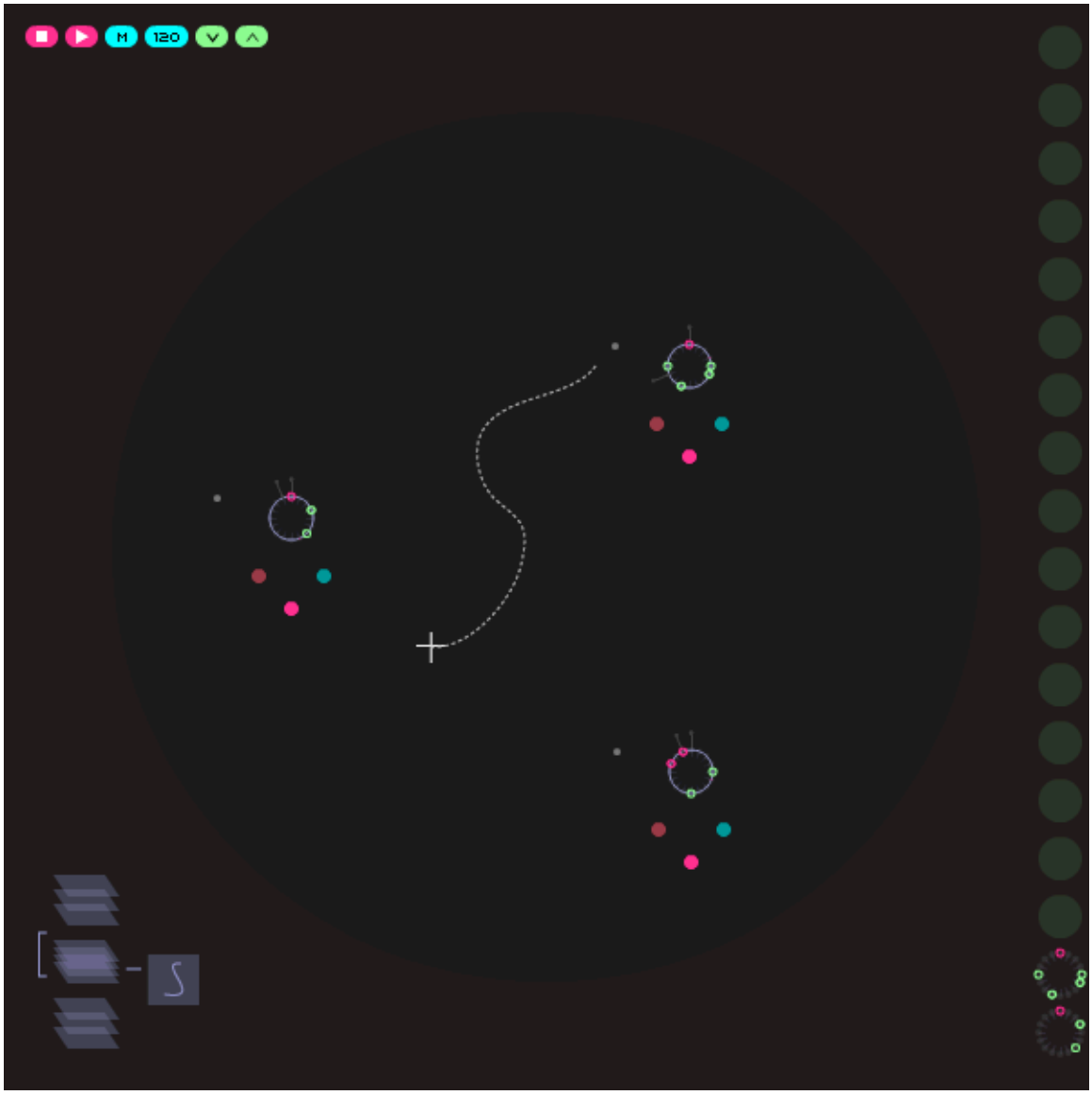

figure FW1

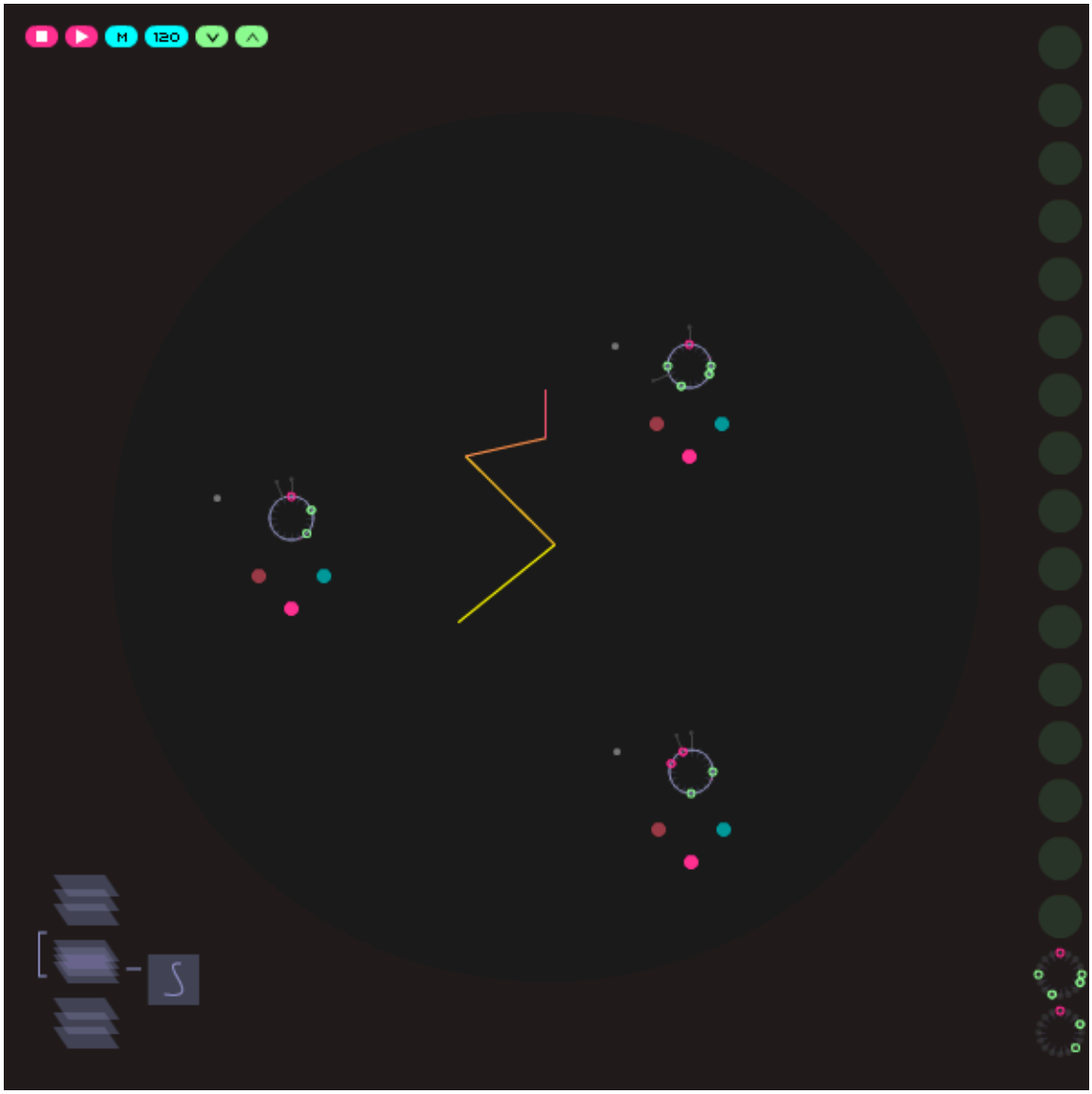

figure FW2

The focus of the first phase of the application has been on specifying and manipulating the synthesis target. Further refinements can be done on the source side. The sound classes can be extended and better defined. As with some other applications, the addition of a database to the system could make for improved source selection as well as allow for more manipulation of the selection process. These improvements can then be coupled with more selection tools at the Browser Level of the application, furthering the interaction possibilities at the low-level of composition.

Although the application design calls for the modulation of attributes with envelope or LFO signals, these have not yet been implemented. Popup controls at the Browser Level will give the user the ability to set LFO parameters or draw envelopes. As described above, the intention here is to provide mechanisms that enable sound evolution on the time scale of musical passages. The attribute controls would modulate several descriptors at once, gradually changing loops as they play.

The mapping of attributes to descriptors also needs to be addressed. Currently, this mapping is specified in a configuration file. Any number of attributes can, in principle, be mapped to any number of descriptors. Each attribute-to-descriptor mapping consists of a numerical range for the attribute, a descriptor name, and a weight. All mappings are presently linear, although other mapping curves are planned. Creating musically appropriate attributes and mapping them to descriptors in a musically useful and mathematically sound manner is a significant challenge. An attribute like "brightness" might map to the energy content, zero-crossing rate, and spectral centroid. However, it will take a lot of fine-tuning to create a mapping that will give the user an expected response. Furthermore, mapping a single attribute value to several descriptors could easily produce impossible combinations or combinations that do not come near any sounds in the corpus.

Going beyond the source selection typical to concatenative synthesis, the application may benefit from the addition of sound transformation based on descriptors, as described in [13]. This could address some of the difficulty of effectively mapping attributes to descriptors, particularly when the descriptor space of the corpus is sparse. Here, transformation would serve to fill in the gaps between source sounds. It may also be desirable to simply transform the selected units for musical effect, by, for example, altering their pitch or key.

Finally, the creation of an evaluation methodology would help to validate the application's design and could also be used to guide future development. Although objective evaluation of the interface may be difficult, some methods as discussed in [14] and [15] might be developed and effectively employed. For example, following

methods from HCI, musicians could be assigned simple tasks appropriate to the interface. Features like learning time, maximum speed, or synchronization could then be measured. Some of these measurements of task performance could then be compared with performance using other interfaces or using other versions of the application's interface. Similarly, discourse analysis might shed some light on the effectiveness of the interface and could perhaps be used to steer design decisions.

7 REFERENCES

[1] Hindle, A., "Ostitch: MIR applied to musical instruments," University of Victoria, 2005. 10 Nov. 2007 <http://churchturing.org/w/ostitch/report/index.html>

[2] Lazier, A. and Cook, P., "Mosievius: feature driven interactive audio mosaicing," Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), London, UK, pp. 312–317, 2003.

[3] Pachet, F., Roy, P., and Cazaly, D., "A combinatorial approach to content-based music selection," Proceedings of IEEE International Conference on Multimedia Computing and Systems, Firenze, Italy, pp. 457-462, 1999.

[4] Schwarz, D., "Current research in concatenative sound synthesis," Proceedings of ICMC, Barcelona, Spain, 2005.

[5] Schwarz, D., "Concatenative sound synthesis: the early years," Journal of New Music Research, vol. 35, no. 1, pp. 3–22, 2006.

[6] Schwarz, D., "Corpus-based concatenative synthesis," IEEE Signal Processing Magazine, vol. 24, no. 2, pp. 92–104, 2007.

[7] Schwarz, D., "New developments in data-driven concatenative sound synthesis," Proceedings of the International Computer Music Conference (ICMC), Singapore, pp. 443–446, 2003.

[8] Schwarz, D., Beller, G., Verbrugghe, B., and Britton, S., "Real-time corpus-based concatenative synthesis with cataRT," Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), Montreal, Canada, pp. 279–282, 2006.

[9] Schwarz, D., Britton S., Cahen, R., and Goepfer T., "Musical applications of real-time corpus-based concatenative synthesis," 9 Nov. 2007 <mediatheque.ircam.fr/articles/textes/Schwarz07b/>.

[10] Xiang, P., "A new scheme for real-time loop music production based on granular similarity and probability control," Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), Hamburg, Germany, pp. 89–92, 2002.

[11] Zils, A. and Pachet. F., "Extracting automatically the perceived intensity of music titles," Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), London, UK, 2003.

[12] Zils, A. and Pachet. F., "Musical mosaicing," Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), Limerick, Ireland, 2001.

[13] Coleman, G.; Bonada, J., "Sound Transformation by Descriptor Using an Analytic Domain," International Conference on Digital Audio Effects. Espoo, Finland, 2008.

[14] MM Wanderley, N Orio , "Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI", Computer Music Journal, 2002.

[15] Stowell, D., Plumbley, M. D., & Bryan-Kinns, N., "Discourse analysis evaluation method for expressive musical interfaces," Proceedings of New Interfaces for Musical Expression, Genova, Italy, 2008.