an incoming event and a clock tick. Therefore, event streams such as the user interface and external control interfaces should be considered when designing the sequence factorizer.

4.3.1. Heterogeneous Clock Rates in Sequences

Consider a case where each control signal is associated with a different source. We would like to maintain the audio sequence as a section, but this is no longer possible for the control section, as each iteration responds to a different activation state.

In this case, the reactive factorization must compute a distinct activation state for each iteration of the sequence. If there is a section of the iteration with an invariant activation state section, this can be factored into a sequence of its own.

Such sequence factorization can be achieved via hylomorphism, which is the generalization of recursive sequences. The theory is beyond the scope of this article, but based on the methods in literature[6], any sequence can be split into a series of two or more sequences. In audio context, this can be leveraged so that any activation-invariant code as possible can be separated into a sequence that can be maintained throughout the compilation pipeline. The activation-variant sections must then be wholly unrolled. This allows the codegen with activation state filtering to produce highly efficient code.

5. CONCLUSIONS

This paper presented an overview of Kronos, a musical signal processing language, as well as the design of its reactive signal model. Kronos is designed to increase the flexibility and generality of signal processing primitives, limiting the vocabulary that is requisite for programming. This is accomplished chiefly via the type system and the polymorphic programming method as well as the unified signal model.

The reactive factorization algorithm presented in this paper can remove the distinction between events, messages, control signals and audio signals. Each signal type can be handled with the same set of primitives, yet the code generator is able to leverage automatically deduced signal metadata to optimize the resulting program.

The concepts described in this paper are implemented in a prototype version of the Kronos compiler which is freely available along with a visual, patching-based interface[2]. For a final version, the compiler is currently being redesigned, scheduled to be released by the summer of 2013. The compiler will be available with either a GPL3 or a commercial license.

Some new developments of a redesigned compiler were detailed, including strategies for handling massive vectors. This is required for a radical improvement in compilation times for applications that involve block processing, FFTs and massive audio banks. As Kronos aims to be an environment where compilation should respond as quickly as a play button, this is critical for the feasibility of these applications.

As the compiler technology is reaching maturity, further research will be focused on building extensive, adaptable and learnable libraries of signal processing primitives for the system. Interaction with various software platforms is planned. This takes the form of OSC communication as well as code generation – Kronos can be used to build binary format extensions, which can be used as plugins or extensions to other systems. LLVM integration opens up the possibility of code generation for DSP and embedded devices. Finally, the visual programming interface will be pursued further.

6. REFERENCES


not to mention the capabilities of most typical multichannel arrays to convincingly localize sound, the authors contend that the flexibility and degree of control of such a system has clear application in the fields of acousmatic composition, live electronics and sound design. Most importantly, with the normalization of multichannel systems in cinemas and the proliferation of massive multichannel systems in institutions worldwide, sound processing techniques that allow for the automation of sophisticated trajectories of sound beyond the possibilities of manual fader-pushing will become increasingly important in the domain of sonic art.

3. CYCLIC/CENTRIC MOTION

Denis Smalley, in his article on ‘spectromorphology’ [11], develops a typology of ‘motion and growth processes’ that may be present in sound-based composition. Of these processes, one category, ‘cyclic/centric motions’, is of particular relevance to this project. This category includes processes such as rotation, spin, spiral, vortex, and centrifugal motion. These processes are defined as giving the ‘impression of motion related to a central point. This can be achieved through spectromorphological variation alone, but is frequently aided by spatial motion.’ These spatial motions, however, are difficult to achieve in a standard DAW, requiring complex fader automation. With spin/drift, however, true multichannel rotations, spins, spirals, vortices and centrifugal motions are trivial.

4. IMPLEMENTATION

spin/drift has been implemented as a Max/MSP patch running on OS X with audio output routed to an equidistant octophonic speaker array. A buffer~ object is first populated with a soundfile or live sound. A custom-coded Max external (spin/drift~) is responsible for calculating the differential equations, locating and updating emitter location and state, generating the array of particles and their state using stochastic formulae, and keeping track of particle birth, life and death. It also calculates the velocities and accelerative forces within the parameter values and ranges set by the user. It maintains an array of grain instances and trajectories, and produces audio output by using a multichannel granular synthesis algorithm in which each grain is dynamically spatialized with Pulkki’s Vector Base Amplitude Panning (VBAP) [12]. Distance encoding of grains is applied using dynamically controlled lowpass filtration and effects bus level (with threshold, base value and rolloff able to be set by the user). The effects bus is routed via separate outlets from spin/drift~ which, in this patch, are sent to a convolution reverb whose impulse response may be chosen by the user for simulation of the required space.

5. CONTROL PARAMETERS

5.1. Coordinate system and typical trajectories

spin/drift uses a coordinate system whose basic unit of distance is the Speaker Array Radius (SAR), equal to the distance from the centre of the speaker array to the perimeter of the array. A value of (0,0) is therefore the centre of the array, (1,0) the rightmost speaker, and (0,1) the front-centre speaker. A number of control parameters may be set in either Polar or Cartesian coordinates. Both emitter and particles can move while the soundfile is being granulated, and this motion can be in circular form (called ‘spin’; angular velocity in degrees/second) or linear (called ‘drift’; a displacement vector measured in SAR/second). Additionally, the emitter and particles can have both spin and drift, and the emitter can also have a ‘spin increment’, creating an Archimedian spiral trajectory. Global accelerative forces include a gravitational pull vector (in SAR/second2) and viscous drag (which slows particle velocity over time).

5.2. Typical Granular Synthesis Parameters

A number of the expected control parameters for a standard granular synthesis algorithm are implemented in spin/drift:

1. Grain duration and variance (in ms)
2. Hop size and variance (in ms)
3. Initial buffer readhead position, readhead ‘jitter’ (ms), and time-stretch amount (% of real-time)
4. Grain envelope type, amount and skew
5. Emitter base pitch shift and drift, particle pitch shift variance and drift (semitones)
6. Grain gain and variance (dB)

5.3. Novel Granular Synthesis Parameters

Unconventional control parameters implemented in this version of spin/drift include:

1. Initial emitter location, spin (°/sec), and drift (SAR/sec)
2. Emitter ‘spin increment’ (i.e. spiral)
3. Particle ‘distance of emission’ and variance from emitter location
4. Particle ‘angle of emission’ and variance (degrees), and whether the angle is absolute or relative to current emitter angle
5. Particle spin and drift, and whether to spin around current emitter location or (0,0)
6. Gravitation pull and/or viscous drag (SAR/second2)
7. Reverberation (see ‘Implementation’) applied dynamically, related to calculated distance of grain position

Figures 1–4 show example trajectories (albeit somewhat simplified) that are easily accomplished in spin/drift. Trajectories are plotted on a Polar grid. The emitter trajectory is indicated with a dotted line, while particle trajectories are indicated with dashed lines. NB: Speakers and head are not to scale.

6. USER INTERFACE

Figure 5 shows the main control GUI, which includes a draggable emitter object and graphical updates of the particle locations. The control parameters are set as standard MaxMSP floating-point objects, and have been exposed to a patr object for preset management and preset interpolation. It is also a trivial development to expose the parameter controls to some kind of MIDi or OSC controller.

7. FUTURE DEVELOPMENTS

An intended and relatively straightforward development is the extension of spin/drift’s pantophonic (2D) soundfield into a periphonic (3D) soundfield through the use of an equidistant spherical or hemispherical speaker array. As a further extension, the spatial locations could be encoded through some kind of ‘space-agnostic’ system such as Ambisonics, assuming the presence of an Ambisonics decoder for playback.

Another intended development is the creation of a modular system of particle ‘behaviours’, as seen in many CGI particle systems. These behaviours are a set of algorithms that can be applied in series, in arbitrary
not to mention the capabilities of most typical multichannel arrays to convincingly localize sound, the authors contend that the flexibility and degree of control of such a system has clear application in the fields of acousmatic composition, live electronics and sound design. Most importantly, with the normalization of multichannel systems in cinemas and the proliferation of massive multichannel systems in institutions worldwide, sound processing techniques that allow for the automation of sophisticated trajectories of sound beyond the possibilities of manual fader-pushing will become increasingly important in the domain of sonic art.

3. CYCLIC/CENTRIC MOTION

Denis Smalley, in his article on ‘spectromorphology’ [11], develops a typology of ‘motion and growth processes’ that may be present in sound-based composition. Of these processes, one category, ‘cyclic/centric motions’, is of particular relevance to this project. This category includes processes such as rotation, spin, spiral, vortex, and centrifugal motion. These processes are defined as giving the ‘impression of motion related to a central point. This can be achieved through spectromorphological variation alone, but is frequently aided by spatial motion.1 These spatial motions, however, are difficult to achieve in a standard DAW, requiring complex fader automation. With spin/drift, however, true multichannel rotations, spins, spirals, vortices and centrifugal motions are trivial.

4. IMPLEMENTATION

spin/drift has been implemented as a Max/MSP patch running on OS X with audio output routed to an equidistant octophonic speaker array.2 A buffer~ object is first populated with a sounfile or live sound. A custom-coded Max external (spin/drift) is responsible for calculating the differential equations, locating and updating emitter location and state, generating the array of particles and their state using stochastic formulae, and keeping track of particle birth, life and death. It also calculates the velocities and accelerative forces within the parameter values and ranges set by the user. It maintains an array of grain instances and trajectories, and produces audio output by using a multichannel granular synthesis algorithm in which each grain is dynamically spatialized with Punkt’s Vector Base Amplitude Panning (VBAP) [12]. Distance encoding of grains is applied using dynamically controlled lowpass filter and effects bus level (with threshold, base value and rolloff able to be set by the user). The effects bus is routed via separate outputs from spin/drift which, in this patch, are sent to a convolution reverb whose impulse response may be chosen by the user for simulation of the required space.

5. CONTROL PARAMETERS

5.1. Coordinate system and typical trajectories

spin/drift uses a coordinate system whose basic unit of distance is the Speaker Array Radius (SAR), equal to the distance from the centre of the speaker array to the perimeter of the array. A value of {0,0} is therefore the centre of the array, (1,0) the rightmost speaker, and (0,1) the front-centre speaker. A number of control parameters may be set in either Polar or Cartesian coordinates. Both emitter and particles can move while the soundfile is being granulated, and this motion can be in circular form (called ‘spin’: angular velocity in degrees/second) or linear (called ‘drift’: a displacement vector measured in SAR/second). Additionally, the emitter and particles can have both spin and drift, and the emitter can also have a ‘spin increment’, creating an Archimedean spiral trajectory. Global accelerative forces include a gravitational pull vector (in SAR/second2) and viscous drag (which slows particle velocity over time).

5.2. Typical Granular Synthesis Parameters

A number of the expected control parameters for a standard granular synthesis algorithm are implemented in spin/drift:

1. Grain duration and variance (in ms)
2. Hop size and variance (in ms)
3. Initial buffer readhead position, readhead ‘jitter’ (ms), and time-stretch amount (% of real-time)
4. Grain envelope type, amount and skew
5. Emitter base pitch shift and drift, particle pitch shift variance and drift (semitones)
6. Grain gain and variance (dB)

5.3. Novel Granular Synthesis Parameters

Unconventional control parameters implemented in this version of spin/drift include:

1. Initial emitter location, spin (*/sec), and drift (SAR/sec)
2. Emitter ‘spin increment’ (i.e. spiral)
3. Particle ‘distance of emission’ and variance from emitter location
4. Particle ‘angle of emission’ and variance (degrees), and whether the angle is absolute or relative to current emitter angle
5. Particle spin and drift, and whether to spin around current emitter location or {0,0}
6. Gravitation pull and/or viscous drag (SAR/sec²)
7. Reverberation (see ‘Implementation’) applied dynamically, related to calculated distance of grain position

Figures 1–4 show example trajectories (albeit somewhat simplified) that are easily accomplished in spin/drift. Trajectories are plotted on a Polar grid. The emitter trajectory is indicated with a dotted line, while particle trajectories are indicated with dashed lines. NB: Speakers and head are not to scale!

Figure 1. ‘Centrifugal’ motion (over 1 sec). Emitter initial location={2,0}, spin=360°/sec. Particle velocity=3 SAR/sec, angle of emission=0° (relative)

Figure 2. Emitter initial position (40) and spin 360°/second. Particle ‘distance of emission’ = 1 SAR, spin 360°/second around emitter location.

Figure 3. Emitter initial position (40) and spin 360°/second. Particle spin -22.5°/sec around (0,0).

Figure 4. (6 secs). Emitter initial location={0,0}, spin=360°/sec, spin increment=1 SAR/sec. Particle trajectories=0 (i.e. at points along emitter trajectory).

6. USER INTERFACE

Figure 5 shows the main control GUI, which includes a draggable emitter object and graphical updates of the particle locations. The control parameters are set as standard Max/MSP floating point objects, and have been exposed to a patch object for preset management and preset interpolation. It is also a trivial development to expose the parameter controls to some kind of MIDI or OSC controller.

Figure 5. GUI controls for spin/drift

7. FUTURE DEVELOPMENTS

An intended and relatively straightforward development is the extension of spin/drift’s pantophonic (2D) soundfield into a periphonic (3D) soundfield through the use of an equidistant spherical or hemispherical speaker array. As a further extension, the spatial locations could be encoded through some kind of ‘space-agnostic’ system such as Ambisonics, assuming the presence of an Ambisonics decoder for playback.

Another intended development is the creation of a modular system of particle ‘behaviours’, as seen in many CGI particle systems. These behaviours are a set of algorithms that can be applied in series, in arbitrary

---

1 [11], pp. 115–117, p. 116
2 Future versions will allow for arbitrary speaker placements. See ‘Future Developments’.

---
order, to affect the trajectories of the emitted particles. As an example, particle behaviours might include:
1. Bounded spaces and edge effects (e.g. ‘bouncing’ trajectories)
2. Interparticle attraction or repulsion
3. Randomized motion (e.g. drunken walks)
4. Dynamical non-linear systems (e.g. Lorenz attractors, cellular automata)
5. Flocking systems (boids, swarms)
6. Vortex and other ‘wind’ motions (may not be so effective in audio)

Two other developments that would reap artistic benefits:
1. A non-linear method for creating grains instead of the current fixed ‘hop-size with variance’. We imagine some kind of ‘swarm’ or ‘burst’ algorithm could be applied to the creation times of grains to further simulate environmental effects
2. Quantization of the readhead to the nearest attack transient to avoid the traditional ‘smoothness’ of granular envelope attack portions in most granular synthesis algorithms.

Finally, we intend to implement a multi-voice version in late 2013.

The spindrift object and sample Max/MSP patch will be available for download from http://michaelnorris.info in late 2013.

8. REFERENCES

FORMAL SEMANTICS FOR MUSIC NOTATION CONTROL FLOW

Zeya Jin
Carnegie Mellon University
School of Music, Pittsburgh, PA
zyej@andrew.cmu.edu

Roger Dannenberg
Carnegie Mellon University
Computer Science Department, Pittsburgh, PA
rbd@cs.cmu.edu

ABSTRACT

Music notation includes a specification of control flow, which governs the order in which the score is read using constructs such as repeats and endings. Music theory provides only an informal description of control flow notation and its interpretation, but interactive music systems need unambiguous models of the relationships between the static score and its performance. A framework is introduced to describe music control flow semantics using theories of formal languages and compilers. A formalization of control flow answers several critical questions: What do the control flow indications mean? What is the mapping from performance location to static score location? Conventional notation is extended to handle practical problems, and an implementation, Live Score Display, is offered as a component for interactive music display.

1. INTRODUCTION

Music notation has been evolving for centuries, creating a symbolic system to convey music information. Early music notation contained only lines and notes, which are sufficient for communicating pitches and durations. It was later that bar lines and time signatures emerged, grouping music into measures and introducing the idea of beats.

The notation for music control flow, like repeats and codas, came even later. Control flow helps to identify repeating structures of music and eliminates duplication in the printed score. In the Classical period, control flow notation is closely tied to music forms such as binary, ternary and sonata and is more of a musical architecture than a means of saving space. Conventional practice for control flow notation is well established. The literature [6, 15] has formalized the notation in all kinds of ways and there is little conflict among definitions. However, traditional music theory has not explored the possibilities of expanded or enriched representations for control flow, and there is a gap between often simplified theoretical ideals and actual practice, especially in modern works. In practice, we find nested repeats, exceptions and special cases indicated by textual annotations, multiple endings, and symbols for rearrangement.

We encountered this gap between theory and practice in the implementation of music notation display software. We needed a formal (computable) way to relate notation to its performance, and we found conventional notions too limiting to express what we found in actual printed scores. To this problem, we developed new theoretical foundations based upon models of formal language and compilation, and we applied these developments to the implementation of a flexible music display system.

Music control flow is the reading order of measures affected by control symbols including the time signature, measures, repeats, endings, etc. It can also be viewed formally as a function f that maps the performed beat k to a location of a score, cm,b,k, a measure and beat pair. (f(k) describes the reading order of the score. In principle, we can rewrite the score in the order f(1), f(2), . . . to create an equivalent score with no control flow (other than reading sequentially). We call this the “flattened score” or “performance score.” Audio recordings and MIDI sequences are both in the order of the flattened representation of the corresponding score.

Existing music theory devotes little attention to control flow, and in fact, there does not seem to be even a standard term for the concept of control flow. To define the meaning of control flow symbols, the conventional practice is to use words and visuals to illustrate the reading order. For example, Read uses arrows to mark the true reading order (see Figure 1) [15]. This approach defines both the syntax and meaning.

Figure 1: Control flow definition in Read’s book