Andrew R. Brown,* Toby Gifford,* and Robert Davidson†
*Griffith University
140 Grey Street, South Brisbane, 4054, Australia
{andrew.r.brown, t.gifford}@griffith.edu.au
†The University of Queensland
St Lucia, Brisbane, 4072, Australia
r.davidson2@uq.edu.au

Techniques for Generative Melodies Inspired by Music Cognition

Abstract: This article presents a series of algorithmic techniques for melody generation, inspired by models of music cognition. The techniques are designed for interactive composition, and so privilege brevity, simplicity, and flexibility over fidelity to the underlying models. The cognitive models canvassed span gestalt, preference rule, and statistical learning perspectives; this is a diverse collection with a common thread—the centrality of "expectations" to music cognition. We operationalize some recurrent themes across this collection as probabilistic descriptions of melodic tendency, codifying them as stochastic melody-generation techniques. The techniques are combined into a concise melody generator, with salient parameters exposed for ready manipulation in real time. These techniques may be especially relevant to algorithmic composers, the live-coding community, and to music psychologists and theorists interested in how computational interpretations of cognitive models "sound" in practice.

Gravity does not explain architecture, but architecture is subject to its law; likewise, perceptual laws do not explain music, but music cannot escape their influence.
— Eugene Narmour (1990, p. 4)

This article presents techniques for interactive composition inspired by models of music cognition. Many of the techniques presented here stem from the practice of live coding—writing software that generates music as a performance practice—and, in particular, the challenge of real-time algorithmic generation of melody.

Algorithmic composition has a long tradition mostly focused on offline music generation, and live coding is a decade-old and burgeoning digital performance practice (McLean, Rohrhuber, and Collins 2014). Despite this, the generation of melodies in real time is still relatively underdeveloped, with many practitioners working in electroacoustic or electronic dance music genres for which melody is of peripheral interest.

Melodic construction has, however, been a central touchstone for studies in music cognition, and there has been particular focus on the principles of Gestalt psychology—relating to the perception of holistic structure and stable organization in the world. A systematic application of Gestalt principles in music theory was initiated by Leonard Meyer (1956) and continued by Eugene Narmour (1992).

More recently, these ideas have been compared to empirical data from computational music analysis and modeled as probabilistic tendencies to which people are sensitized through enculturation (Huron 2006; Temperley 2007). As such, these concepts are amenable to computational implementation as outlined in this article. Our motivation for this work is to develop computer-assisted compositional techniques for creative purposes and we are not suggesting that the algorithms are either cognitive or compositional models. In particular, we are interested in achieving a parsimonious implementation that supports creative exploration, rather than providing a complete and accurate model of the psychological process that inspired the algorithms.

The attention of many music cognition studies is on symbolic music representations, in particular, diatonic and metric music represented as a notated score. The techniques presented here inherit this focus, and the examples use symbolic representation of music as "note" events. This article does not discuss the direct application of these ideas to audio signal processes—instead, we concentrate on probabilistic modeling of melody, exemplified through algorithmic music practices, for the application of the techniques.

The choice of a common-practice musical context for our research, which includes diatonic pitch sets and metric rhythms, was made for several reasons. First, much of the music cognition research

Computer Music Journal, 39:1, pp. 11–26, Spring 2015
doi:10.1162/COMJ_a_00282
© 2015 Massachusetts Institute of Technology.

that we are inspired by was conducted in these contexts and its application beyond these is not entirely certain. Second, the ability to judge the success of the algorithmic techniques is made more straightforward when comparing them to a well-understood aesthetic framework. Finally, there are many creative applications for diatonic and metrical music, to which the techniques may be applied, including most Western popular music.

Melody—a monophonic series of note events—is an important musical element for many styles of music, and consequently it is often an integral part of the compositional process. Generated melodies assist this process by providing material that may be applied to a work in the form of musical riffs, bass lines, or themes, for example.

Algorithmic music practices, especially live coding, benefit when techniques can be concisely implemented and they are open to improvised modification (Brown and Sorensen 2009). In developing these techniques inspired by music cognition, we have searched for succinct methods of expression and sought to expose musically salient parameters to facilitate control during live performance or interactive composition. These practical considerations have naturally acted as a filter on the types and complexity of the psychological theories that can be applied. Indeed, the individual techniques we present are simplistic. In keeping with the Gestalt perspective, however, we contend that their use in combination amounts to more than their sum. In implementing these techniques, we have built on our previous work in algorithmic music methods (Sorensen and Brown 2007) and utilized libraries and design patterns from the Impromptu programming environment (Sorensen 2005). The examples presented are coded in the language Scheme with Impromptu extensions, although it should be straightforward to translate them into other computer music environments.

Background

The generative techniques presented in this article take the form of probabilistic models from which notes are generated stochastically. The use of probabilistic models has a long history in generative music, both for sound synthesis (Xenakis 1992) and for symbolic score generation (see Nierhaus 2009 for an overview). In score generation, probabilistic models of music generation have included stochastic grammars (Cope 2000), probabilistic music theory (Cambouropoulos 1997), and purely statistical models (Papadopoulos and Wiggins 1999).

Probabilistic models have also found their way into perceptual theories of music. Adopting Meyer's (1956) theory of musical expectations, music cognition researchers have articulated a "statistical learning" account of musical expectations, and formulated corresponding statistical descriptions of music (Huron 2006, p. 360). Such descriptions have not, however, been widely utilized for generative purposes. Some notable exceptions are Pearce and Wiggins's (2006) experiments in "analysis by synthesis," and the more recent work of Maxwell, Pasquier, and Eigenfeldt (2011).

In this article, we implement functions for melody generation loosely based on such probabilistic models of expectation. Following Huron (2006), Temperley (2007), and others, we discuss these techniques in relation to corresponding concepts from Gestalt psychology, from which Meyer (1956) and Narmour (1990) originally drew inspiration.

Proximity

Some of the better-known Gestalt principles are those related to grouping. One of these is the *law of proximity*, which states, "objects or stimuli that are viewed as being close together will tend to be perceived as a unit" (Corsini 1999). Although typically discussed in terms of visual perception, the law of proximity in its original formulation was proposed to apply to both visual and auditory perception (Wertheimer 1938). Numerous psychological studies have established proximity as a general grouping principle in music, via proximity in time (Deutsch 1999a), proximity in pitch (Bregman 1990), or proximity or similarity along more abstract musical dimensions (Deliège 1987).

Figure 1. Gaussian distribution used for selecting the pitch, dynamic, and duration of a note.

Figure 2. A random walk function for iterative melodic pitch selection.

Figure 3. A random walk exceeding reasonable range bounds.

```
(play piano (pc:quantize (random-gaussian 60 4)    ;; random pitch
                         (pc:scale 0 'ionian)) ;; diatonic constraint
                         (random-gaussian 100 10)  ;; random velocity
                         (random-gaussian 0.4 0.1));; random duration
```

Figure 1

```
(define (g-walk pitch)
    (play-note (now) piano pitch 100 *second*)
    (callback (+ (now) *second*) 'g-walk
              (pc:quantize
              (random-gaussian pitch 3)
              (pc:scale 0 'ionian))))
```

Figure 2

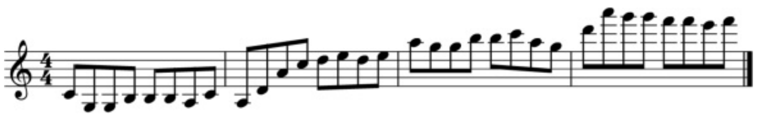

Figure 3

These observations from music psychology are consistent with rules expounded in traditional composition texts advising that pitches in melodies should often move by step, good voice leading should minimize interval step size, and so on (see for example, Goetschius 1902).

For generative purposes, one of the areas where proximity has an impact is in melodic sequences. The code in Figure 1 uses a Gaussian distribution for selecting the pitch, dynamic, and duration of a note. The use of a Gaussian distribution rather than, for example, a uniform distribution over a narrow range, aligns with models from the literature (von Hippel and Huron 2000; Temperley 2007), and seems to us to produce pitch variation that sounds more natural.

In Impromptu, the `random-gaussian` function takes two arguments: mean and standard deviation. The pitch is a MIDI pitch value in the range 0–127, which we constrain within a diatonic pitch-class set; the dynamic is a MIDI velocity, also in the range 0–127; and the duration is in beats (1 beat = 1.0). This code will choose a random diatonic pitch close to middle C (MIDI pitch 60) quantizing to a C major (Ionian) pitch class set. Notes will have a MIDI velocity near 100 and a duration of about 0.4 beats.

Random Walk

Pitch proximity, as a melodic tendency, can be modeled by a *random walk*, a commonly used process in generative music (Xenakis 1976; 1992, p. 289). A random walk is a sequence that moves by randomly sized "steps." That is, each element is generated by adding a small step to the previous element. We can use the Gaussian distribution to construct a simple random walk melody as shown in the code in Figure 2. The next pitch is randomly selected in the last lines of the function.

Range Constraint

Although a random walk is an easy way to generate material, the melody may meander beyond the playable pitch range, as shown in Figure 3. The beaming patterns in the early notated examples are somewhat arbitrary. Later in the article metric organization becomes explicit and time signatures and barlines are introduced. Audio versions of the musical examples accompanying this article are available online at explodingart.com/cmj-melodies.

Equally problematic from the perspective of music perception is that this undirected meandering

Figure 4. The `elastic-g` *function and a method to constrain its output to a diatonic context.*

Figure 5. Some output from use of the `elastic-g` *function for pitch.*

```
(define (elastic-g prev sd fixed amnt)
    (random-gaussian (+ fixed (* (- prev fixed) amnt)) sd))

(pc:quantize (elastic-g pitch 3 60 8/10) (pc:scale 0 'ionian))
```

Figure 4

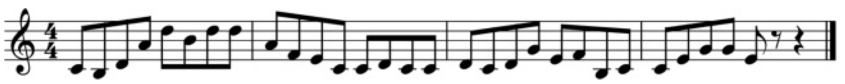

Figure 5

is at odds with computational musicology literature claiming that "most melodies seem to favor the center of their range" (von Hippel 2000, p. 316).

A simple approach to range constraint is the imposition of boundary limits by clipping. Hard boundaries tend to trap values near the boundaries, however, making for passages that sound unnatural. A variation is to use "mirror" boundaries that "bounce" the values away from the boundaries by the extent of their overshoot (Xenakis 1992), which somewhat (but not entirely) mitigates boundary hugging.

Alternatively, in the spirit of research by von Hippel and others we propose a range constraint technique that operates by adding a tendency toward the mean of the range. Rather than use of the previous pitch as the point of departure for the next step in the random walk, a mean value is selected between the previous step and the middle of the range.

This technique creates a kind of "elastic band" force (Larson 2012) that increases with the distance away from the mean. Varying the strength of this force—shown as the argument 8/10 in Figure 4—allows the composer to control the pitch range according to their stylistic preference. Code implementing our `elastic-g` function is shown in Figure 4, with some example, iterated output notated in Figure 5.

Goal-Oriented Behavior

The need for further structure in the melodies shown so far is clear. The random walk process, even when range-constrained, lacks larger-scale organization and the "meandering" is, and sounds, directionless.

In music theory, the importance of "structural tones" in a melody is widely discussed. Structural tones, being important points in the harmonic and melodic contexts, can provide points of climax, variety, and resolution. In the field of musicology, structural points in music have been characterized as nodes in "time-span reductions" (Lerdahl and Jackendoff 1983) or in "event hierarchies" (Bharucha 1984). Empirical research has found that structural tones are important elements in musical improvisation (Large, Palmer, and Pollack 1995) and listening (Bigand and Parncutt 1999).

Directed Random Walk

To provide the random walk melody with more structure, we constructed a technique that allows target pitches to guide the melodic contour. We call this a directed random walk.

The `elastic-g` function already has the desired property of drawing the pitch toward a target; previously understood to be the middle of the range. To obtain a directed random walk we combine the range constraint tendency and the goal direction tendency by calculating a moving target for the `elastic-g` function that balances the two tendencies. So that this is easy to use in dynamic creative contexts, a new function that encapsulates this process is created, called `directed-g`. The code in Figure 6 shows the function and an example of using of it for calculating a next pitch.

Figure 6. The `directed-g` *function and its use in selecting a pitch constrained to a diatonic context.*

```
(define (directed-g mean sd target t-weight middle m-weight)
    (elastic-g mean sd (+ mean (+ (* (- middle mean) m-weight)
                                  (* (- target mean) t-weight)))
                       8/10))


(pc:quantize (directed-g pitch 3 84 .8 60 .2) (pc:scale 0 'ionian))
```

So far, we have provided a process for moving toward a single target. In order to provide a useful guide to melodic contour, a series of pitch targets is required, along with a way of progressing from one to the next. A utility function, `make-stepper`, was created to facilitate iteration through a list of pitch values that will serve as structural tones. When the target pitch is reached, the next target pitch is selected. The code in Figure 7 implements a directed random walk function that uses the `directed-g` function and the `make-stepper` utility.

The music in Figure 8 shows some typical output from this directed Gaussian walk program. Notice that each target (C5, G3, C6, and C4) is hit, that sometimes there is an overshoot of the target (either before or after it is hit), and the time taken to hit successive targets bears only a moderate correlation to the pitch distance traveled.

Good Continuation

The Gestalt law of *good continuation* states that there is an innate tendency to expect that a perceived pattern or implied direction will continue. In music, this might suggest that an upward series of pitches can have a tendency to continue upwards, or that an established 4/4 meter can be relied upon to persist. Good continuation, then, is strongly aligned with repetition, reuse, and development of musical fragments.

Wundt Curve

A factor that moderates tendencies of good continuation is the interest created by confounding expectations: pleasant surprises. The way the balance between interest and tedium changes with repeated exposure was studied by the German psychologist Wilhelm Wundt in the 1800s and resulted in the *Wundt curve*, "a well-known arousal response curve from studies of animals and humans to various forms of arousal" (Saunders and Gero 2001). The curve, shown in Figure 9, depicts change in interest as the number of exposures (repetitions) increase.

Meyer discusses this effect in the context of music, by reference to the *principle of saturation*, whereby "a figure which is repeated over and over again arouses a strong expectation of change" (Meyer 1956, p. 134).

Wundt's theory suggests that recapitulation of previous structure should be used in moderation to maximize the benefits of familiarity and expectation, while avoiding being overly predictable and uninteresting. In some musical circumstances, such as in electronic dance music, repetition can become stable and expectation of change is modified. In a personal communication to the authors in 2013, Narmour suggested that "once a certain point of repetition is reached, the ongoing change recedes to ground, like wallpaper, which is the nature of a vamp (rather than ending in increasing frustration)." In this section, we will look at techniques that help provide more local structure and patterning, while maintaining a degree of unpredictability.

Process

According to Narmour, repeated notes (specifically, pitch repetition ignoring duration, which may be different from note to note) and sequences of small steps in the same direction are both "subject to the bottom–up Gestalt laws of similarity, proximity, and common direction" (Narmour 1990,

Figure 7. A melody function iterates through a list of pitch values that serve as structural tones.

Figure 8. Output from the `dir-g-walk` *program with target pitches highlighted.*

Figure 9. The Wundt curve.

```
(define (dir-g-walk beat pitch target)
    (play piano pitch 100 1/2)
    (callback (*metro* (+ beat (* 1/2 1/2))) 'dir-g-walk
              (+ beat 1/2)
              (pc:quantize (directed-g pitch 4 target .8 60 .2)
                           (pc:scale 0 'ionian))
              (if (= pitch target) (stepper (list 72 55 84 60))
                                   target))))
```

Figure 7

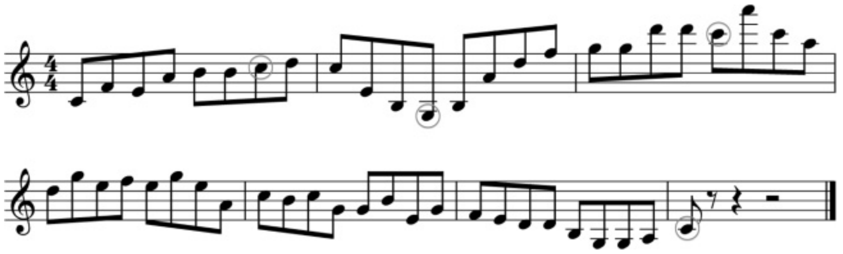

Figure 8

p. 97). He suggests that when we hear repeated intervals or pitches we infer a pattern, and anticipate continuation of that pattern. Following Meyer, he refers to a continuing pattern generally, including repetition of "signed," or directed, intervals, as a *registral process*. For example, a pitch step from C to D would imply a step from D to E. In Narmour's theory, the process tendency only happens for intervals smaller than five semitones. Therefore, in our implementation intervals smaller than five semitones are candidates for repetition. Narmour's theory is more nuanced and proposes *intervallic process*, where interval size is the expectation regardless of direction.

Various theorists suggest, but for different reasons, that there is a tendency for a reversal of direction after larger intervals. As indicated previously, we include procedures for ensuring directional change that bear some similarity to von Hipple's (2000) notions of a regression to the mean. We use a simplified notion of "process" here, where directional momentum is maintained. This can, in part, be interpreted as a nod toward the musical "forces" analogy proposed by Larson (2012).

The code in Figure 10 shows our implementation of an interval process tendency for use in our

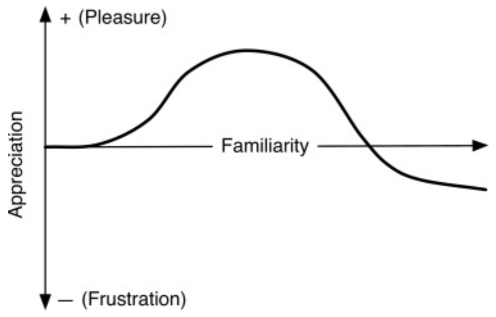

Figure 9

melody generator, which will generate a repeat of the previous interval when three conditions are met. First, the interval should be less than a tritone. Second, in keeping with the Wundt curve, the number of consecutive processes (i.e., repeats) is limited to four. Third, for variety, there is a small (30 percent) chance that the intervallic process will end on any given step. The percentage value and maximum repeat value were chosen based on experimentation and can be varied for aesthetic effect.

The process function created by `make-process`, when called, returns either the next pitch in the intervallic sequence or, if a nonsequence pitch needs

Figure 10. This code generates a function that returns either the next pitch in a sequence being repeated or false when repeat-halting conditions are met.

Figure 11. Use of the repeat making function `make-process`*.*

```
(define make-process
    (lambda ()
        (let ((prev-p 60) (repeat-cnt 0))
            (lambda (pitch)
                (let* ((interval (- pitch prev-p))
                       (next-pitch (+ pitch interval)))
                    (set! prev-p pitch)
                    (cond ((or (> (abs interval) 5)
                               (> repeat-cnt 3)
                               (< (random) .3))
                           (set! repeat-cnt 0)
                           #f)
                         (else (set! repeat-cnt (+ repeat-cnt 1))
                             next-pitch)))))))
```

Figure 10

```
(define process (make-process))

(define (proc-g-walk beat pitch target)
    (play piano pitch 100 4/10)
    (let ((next-pitch (process pitch)))
        (if (boolean? next-pitch)
            (set! next-pitch (directed-g pitch 3 target .8 60 .4))
            (callback (*metro* (+ beat 1/4)) 'proc-g-walk
                      (+ beat 1/2)
                      (pc:quantize next-pitch (pc:scale 0 'ionian))
                      (if (< (abs (- next-pitch target)) 0)
                          (stepper (list 72 55 84 60))
                          target)))))
```

Figure 11

to be selected, the value "false." It can be embedded into a melody generator as shown in Figure 11 with the `proc-g-walk` function.

The effect of imposing some degree of interval step process on the melody algorithm can be seen in Figure 12. There are more scalar and broken-chord passages within the pitch contour. This reduces the number of changes in direction and provides some greater sense of intentionality at the note-by-note level.

Segment Process

Thus far, we have focused on the ways in which pitch sequences are organized, based on features that aid perceptual continuity. Many studies from both music cognition and music theory have also emphasized the way humans structure music into larger chunks, or segments. Studies from both music theory (Goetschius 1902) and music perception (Narmour 1973; Huron 2006) have highlighted the musical importance of structured reuse of musical segments such as motifs, themes, and variations. Reuse of materials makes the larger-scale structuring quite evident, and studies have shown that sequences structured in this way are easier to understand and recall, and perhaps more readily appreciated (Deutsch 1999b).

For simplicity, we focus on segment repetition (possibly transposed), while continuing our strategy of including stochastic variation to add novelty and interest.

A utility function `make-seg-repeat` does much of the hard work for repeating segments of the melody. Like the `make-process` function, it takes care of "remembering" recent note pitches, but in this case it stores the last *n* values, where *n* is specified when the function is called. The variability allows the composer to determine, at

Figure 12. A notated sample of output from the `process-g-walk` program.

Figure 13. Output from the `seg-g-walk` program.

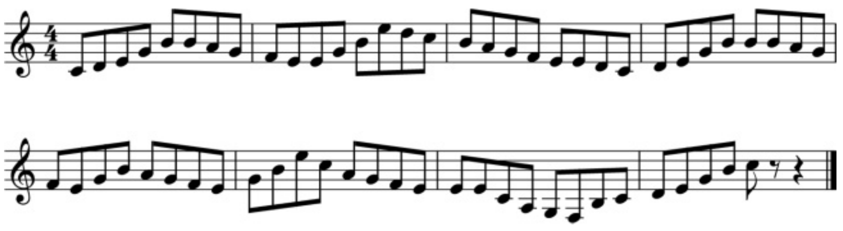

Figure 12

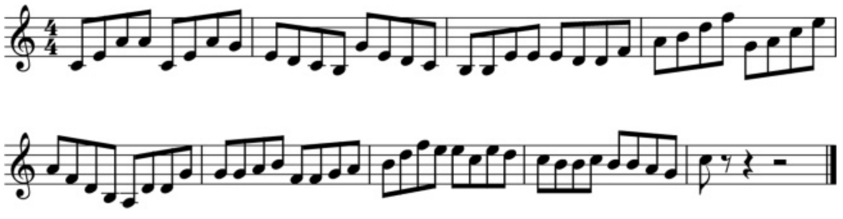

Figure 13

run time, how large a segment should be to match local considerations, such as metric context. The second argument is `bypass-amnt`, which specifies the degree to which the stored values should be replaced with newly generated values.

There are options in the `make-seg-repeat` function to allow for transposition of the segment. The two options available are: (1) `transpose-range`, the distance either side of the original pitch the repetition may be transposed, measured in semitones; and (2) `pitch-class-set`, the scale that will be used for the transposition, which is most likely to be the same as that used in other parts of the generative process and will allow for approximately diatonic transpositions. At the start of each repeated sequence, a random transposition amount within the range is chosen. This function returns another function, named `seg-repeat`, shown here, which is the function most directly used while composing or performing:

```
(define seg-repeat (make-seg-repeat 4
0))
```

The use of the `seg-repeat` function is quite straightforward. All pitches are passed through it just before playback and the function either passes these pitches on, or it replaces them with those from a repeating segment it has stored. This approach is achieved using the following code:

```
(set! pitch (seg-repeat pitch))
```

Output from the melody generation program using `seg-repeat` is displayed in Figure 13. The segment length is set to four, and the transpose range value to two, which allows for sequences as well as direct repetitions, and the bypass value is set to 0.3, which introduces some variety into the repetitions.

Rhythmic Pulse and Ratios

Up to this point we have neglected rhythm in constructing melodies. We now apply the concept of process to the sequential ordering of rhythmic values. Music is often pulse based, and durations are generally simple multiples or divisors of this pulse. In Western music theory, this is evident in the relationships between commonly used durations. In music psychology, the phenomenon of pulse and of intervals of simple pulse ratios has been demonstrated outside of any cultural music context

Figure 14. Code to generate and play back a melody that now includes rhythmic variation selected by the `next-g-beat` *function.*

```
(define rhythmic-g-walk
    (lambda (beat pitch target dur)
        (set! pitch (seg-repeat pitch 0.5 3))
        (play (/ (random) 30) piano pitch 100 (* dur (random 6 10)
              1/10))
        (let ((next-pitch (process pitch)))
            (if (boolean? next-pitch)
                (set! next-pitch (directed-g pitch 2 target .8 60
                                    .4)))
            (callback (*metro* (+ beat (* 1/2 dur))) 'rhythmic-g-walk
                      (+ beat dur)
                      (pc:quantize next-pitch (pc:scale 0 'ionian))
                      (if (< (abs (- next-pitch target)) 3)
                          (stepper (list 60 72 55 84 60))
                          target)
                      (next-g-beat dur 1/2 (list 1/4 1/2 1) 0 1)))))
```

by various "tapping" tests (e.g., Fraise 1984). We are interested here in the selection, order, and distribution of durations for our melody case study.

Taking as a starting point the isochronous eighth-note pattern we have used in examples until now, we can elaborate the rhythm by implementing simple divisions and groupings of the eighth note. This has a correspondence with dynamic attending theory (Jones and Boltz 1989). Following the terminology of this model, the eighth-note pulse serves as the *referent time period* that "anchors" our temporal attention. Subdivision and larger groupings of the referent period are perceived as such.

Restricting attention to the ratio two, for example, rhythmic elaborations of the simple eighth-note pulse are either subdivided into two sixteenth notes, sticking with the eighth note, or extended to a quarter note. To choose from this set of durations we select randomly, with a Gaussian distribution centered on the referent period (eighth note), and quantize to this discrete set of choices.

To implement Gaussian selection of durations from a set, we have developed some utility functions using design patterns similar to those shown above for pitch. One of these, `next-g-beat`, will be used directly. It accepts the previous duration value and returns the next one. The function exposes parameters that we feel are valuable to vary in real time. These include the referent time period (which might change infrequently), the duration list that includes all the duration values from which to select (typically an ordered set with the referent period in the middle of the list), and the mean and standard deviation values for the Gaussian distribution. Varying the standard deviation is particularly useful because as this value gets higher it makes it more likely that outlying values (toward either ends of the list) are selected. As with our early descriptions of pitch organization, one perceptual effect that is not taken into account in this implementation is the ordering of rhythmic values, or groupings, nor the musical significance of increasing or decreasing information rates (note density).

We use the `next-g-beat` function as part of our melody program, `rhythmic-g-walk`, which selects note durations and ensures rhythmic groupings based on the referent duration (one half beat, in this case). In this code fragment, we have also added to the play function some variation in onset timing using the optional second argument. We also added some variation in the performed duration of each note—varying from 60 percent to 100 percent of the interonset duration. The latest version of the program, which includes the rhythm generator, is displayed in Figure 14. An example of output is shown in Figure 15.

As an experiment in the utility of the `next-g-beat` function for variation at run time, the music in Figure 16 was generated by the same `rhythmic-g-walk` program but with some changes: the same referent period, an addition to duration values to include triplets, and a wider standard deviation for more even use of the durations. Figure 16 displays output generated using the following settings:

```
(next-g-beat dur 1 (list 1/3 1/4 1/2
1) 0 1.5)
```

Figure 15. Sample output from the `rhythmic-g-walk` *program.*

Figure 16. Adding variety that is more rhythmic by varying beat selection parameters.

Figure 17. Pulse strengths in a 4/4 meter. (Adapted from Lerdahl and Jackendoff 1983.)

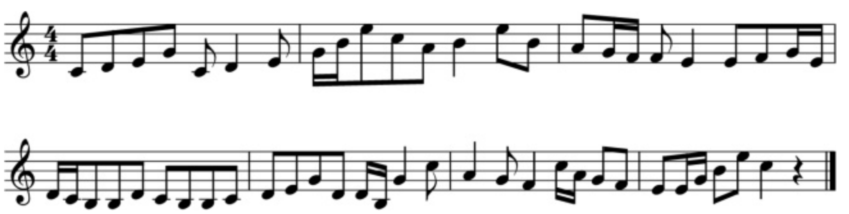

Figure 15

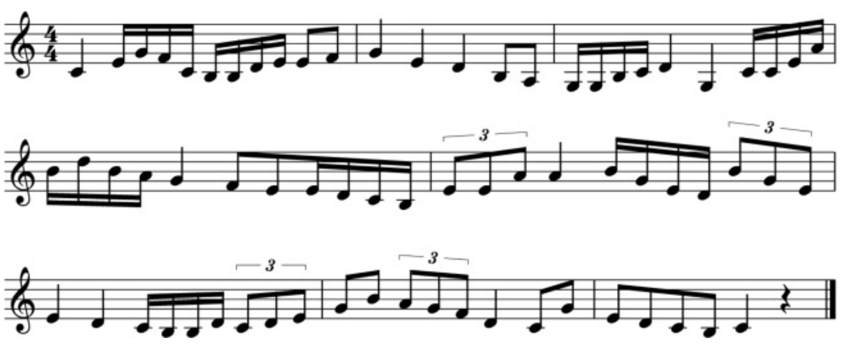

Figure 16

Metric Contextual Sensitivity

Musical meter is frequently described as a regular pattern of strong and weak pulses (Cooper and Meyer 1960; Lerdahl and Jackendoff 1983; Large 1994). London (2004, p. 4) maintains that "meter is a perceptually emergent property of musical sound, that is, of our engagement with the production and perception of tones in time." A number of factors contribute to emphases of pulses (and hence perception of meter) including accent, duration, harmony, and timbre (Cooper and Meyer 1960, p. 7).

The manner in which musical features depend on their position within the meter is an important aspect of melodic structure (Narmour 1990; London 2004). We refer to position of a beat within the metrical pattern (such as downbeat, second beat, etc.) as the *metric context*. In this section, we discuss the importance of metric context, and how we can use it to condition statistical distributions of dynamics, pitch, duration, and harmonic progression.

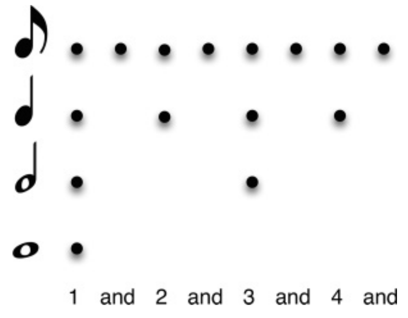

Figure 17

Metric Emphasis

The strength of various beat locations within the meter is often shown as equivalent to the number of metrical levels within which the beat belongs, as shown in Figure 17.

Metric emphasis differs from accent, which is a surface level property of the music. Accent may be created by stress, by dynamics, by articulation (Cooper and Meyer 1960, p. 7), or more generally

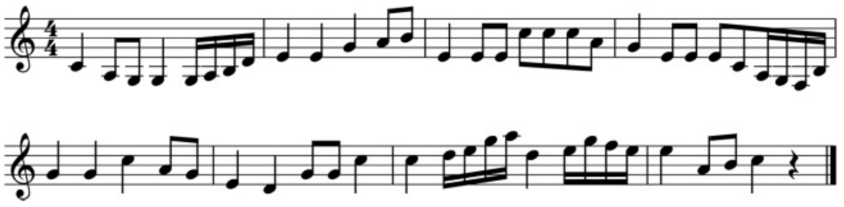

Figure 18. `Metric-g-walk` *with metrically conditioned pitch and duration.*

by a contrasted value of any salient musical parameter (Deliège 1987). Metric emphasis, in contrast, is a purely perceptual phenomenon formed in response to the patterning of accents. In order for meter to emerge from patterning of accents, it seems reasonable that the distribution of accents should be conditioned in some way by the metric context.

Some studies (Huron 2006; Temperley 2007) have found that note onset probabilities correlate with metric emphasis. Dynamics are also assumed to correlate with metric emphasis (Cooper and Meyer 1960; Clynes 1983), and harmonic changes are most likely to occur on downbeats (Temperley 2007). Additionally, our own research indicates that probability distributions for both duration and tonal stability depend on metric emphasis.

In the following sections we implement a form of *metric contextual conditioning* for these musical parameters in our melody generator. In the interest of brevity, we adopt a single simple mechanism for approximating the contextual behavior of these parameters. The value for each parameter (dynamic, pitch, duration, and harmony) is randomly selected from a discrete list of possible values. These lists are ordered by level of emphasis. For example, the possible values for pitch are drawn from the list of diatonic scale degrees, ordered by tonal stability based on the Krumhansl-Kessler key profiles: 1 5 3 4 6 2 7 (Krumhansl and Kessler 1982). In a given metric context this list will be truncated, however, further constraining the possible values. On the downbeat, for example, the list is truncated to triadic pitches (i.e., 1 5 3) whereas on a metrically weak offbeat the full list is available for random selection. Similarly, a list of available dynamics ordered from loud to soft, and durations from long to short, are truncated according to metric context. This technique is implemented by the function `shrink-list`.

Metric Constraints

One application of the `shrink-list` function is for pitch selection. The function is used as a modifier for the pitch-class list, and the result is passed to the `pc:quantize` function. The argument for minimum length is set to three, so that the most restrictive list (used on the downbeat) consists of the first three pitches in the list—the triadic pitches.

```
  (pc:quantize next-pitch (shrink-list
beat '(0 7 4 5 9 2 11) 3))
```

To modify duration choices the shrink-list function is applied to the list of available durations before selection. Recall that the second argument is the reference time period (set here to 1 [i.e., a quarter note]) and that the duration list values have been selected to be ratios of the reference time period (e.g., the argument 1/2 represents an eighth note).

```
  (next-g-beat dur 1 (shrink-list beat
'(1 1/2 1/4 1/3) 2) 0 1)
```

The output shown in Figure 18 demonstrates that with the metric constraints in place notes on the downbeat are pitch limited to C, E, and G and that rhythmic groupings are based on the referent time period; a quarter note.

Harmonic Progression

To implement a change in harmonic context, substitutions to the active pitch class set can be made. These changes are controlled by a simple Markov process—a common approach to modeling chord progression expectations that we based on work by Huron (2006, p. 251). A change of harmonic context is triggered at the end of every four beats.

Figure 19. A melody with shifts between C, D, and B♭ major.

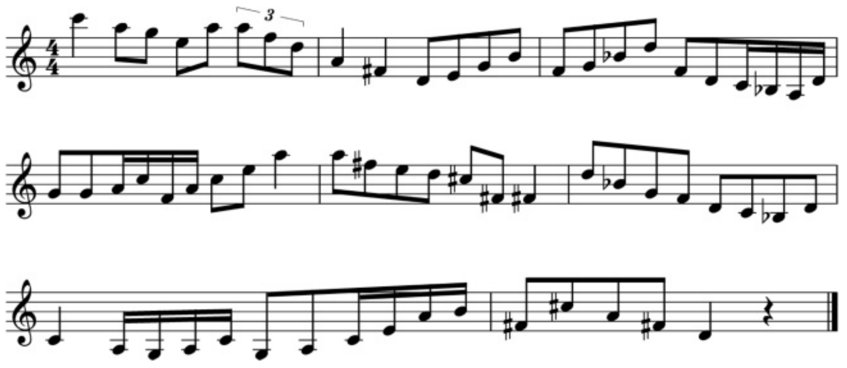

Figure 19 shows a sample of the output after adding this change. For the purpose of visual clarity, harmonic changes between C, D, and B♭ major were specified (even thought this is an unlikely harmonic progression) so that additional accidentals would make the transitions more obvious in the notation.

Closure

Closure, or stability, is a fundamental notion in Gestalt psychology, playing a complementary role to good continuation, or process. The techniques described earlier seek to engender continuation, motion, momentum, instability, or otherwise imply that the music is ongoing. Closure, conversely, is a perception of completion, repose, or stability (Meyer 1956, p. 139). Compositionally, we will use closure to provide a sense of phrasing.

As a first step towards utilizing closure in dynamic algorithmic music systems, we sought to create a real-time measure of the "level of closure" in a generated musical stream. Inspired by Meyer's theory, we consider numerous factors as contributing to the overall sense of closure. Completion of pattern—melodic, harmonic, and rhythmic—are key components. Additionally, the local dynamics of various musical parameters, without reference to prior patterning, is attributed with closural power (Meyer 1956, p. 81).

Narmour (1990) has articulated an explicit theory of parametric closure. His conditions of closure include a number of simple properties of note-to-note transitions: movement from a short to long duration, weak to strong metric emphasis, dissonance to consonance, large to small interval, and change in registral direction.

We have previously implemented computational analyses for tracking these conditions of closure (Brown, Gifford, and Davidson 2012), and applied them to a corpus of folk songs, to find patterns of congruence among the parameters and to assess their relative contributions to melodic completion. There we concluded that rhythmic, metric, and tonal conditions of closure contributed most strongly.

Phrase Endings

Meyer suggested that points of strong closure articulate melodic structure, acting as the endpoints of perceptual objects in the musical surface. By calculating a real-time measure of the total parametric closure in our generated melodies, we hope to identify "opportune" moments for ending phrases, or ending the melody. We have implemented closure functions independently for metric, tonal, dynamic, and durational closure. The `total-closure` function calculates the sum of these.

We have found that designing an algorithm to find opportune points to stop playing is one of the more problematic generative music tasks, and this function seems to be reasonably effective, though not entirely reliable. The notated examples presented previously have used manually selected end locations, but Figure 20 shows an entire melody generated and halted by the melody program using the closure measure.

Figure 20. A melody generated and concluded algorithmically.

Figure 21. A function that integrates all the techniques presented in this article.

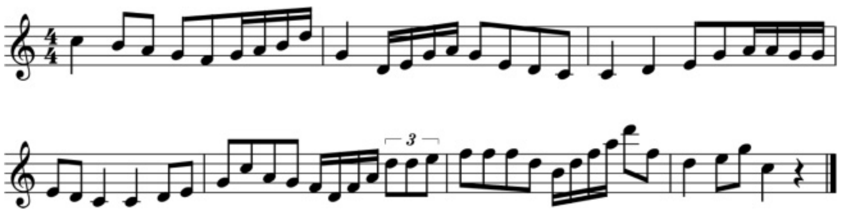

Figure 20

```
(define closure-g-walk
    (lambda (beat pitch target dur pc-set cnt)
        (set! pitch (seg-repeat pitch 0.7 2 2
                        (shrink-list beat pc-set 3)))
    (let* ((dyn (cosr (random-gaussian 90 5) 15 .5))
          (next-pitch (process pitch))
          (tc (total-closure (list beat pitch dyn dur))))
          (play (/ (random) 30) piano pitch dyn
                (* dur (random 6 9) 1/10))
          (if (boolean? next-pitch)
              (set! next-pitch (directed-g pitch 2 target .8 60 .5)))
          (if (= (modulo (+ beat dur) 4) 0)
                (set! pc-set (list-ref pc-sets
                   (random (cdr (assoc (car pc-set) prog)))) pc-set))
          (if (or (> tc 7) (< cnt 32))
              (callback (*metro* (+ beat (* 1/2 dur)))
                        'closure-g-walk (+ beat dur)
                        (pc:quantize next-pitch (shrink-list
                             (+ beat dur) pc-set 3))
                        (if (< (abs (- next-pitch target)) 3)
                            (stepper (list 60 72 55 84 60))
                            target)
                        (next-g-beat dur 1 (reverse (shrink-list
                               (+ beat dur) '(1 1/2 1/4 1/3) 2)) 0 1)
                        pc-set
                        (+ cnt 1))))))
```

Figure 21

The code in Figure 21, `closure-g-walk`, integrates all the techniques presented in this article, and produced the melody shown in Figure 20. The size and complexity of the code are manageable for creative purposes, although a complete work would likely include more than melodic phrases. The code is densely packed with variables that enable run-time editing and manipulation. As we have seen throughout this article the techniques can be accumulated, and some could be omitted as required. This ability for code structure to have a developmental quality is particularly important for performative practices—allowing a kernel of an idea to be quickly articulated and then elaborated over time. This kind of functionality can also assist compositional processes and the fluid development of melodic ideas.

As these techniques have been developed, they have been tested in our musical practice. In particular, they were used by the authors for the performance of the original work *Multiple Beginnings* at the First International Live Code Festival (Germany) in April 2013; for the audio-visual installation *Connections*, selected for inclusion in the *[d]Generate* exhibition of generative art at the Gympie Art Gallery (Australia) in June 2013 (a video excerpt of *Connections* is available online at vimeo.com/67930857); and in the composition of a work titled *Entanglement*, presented at the 2013 International Computer Music Conference

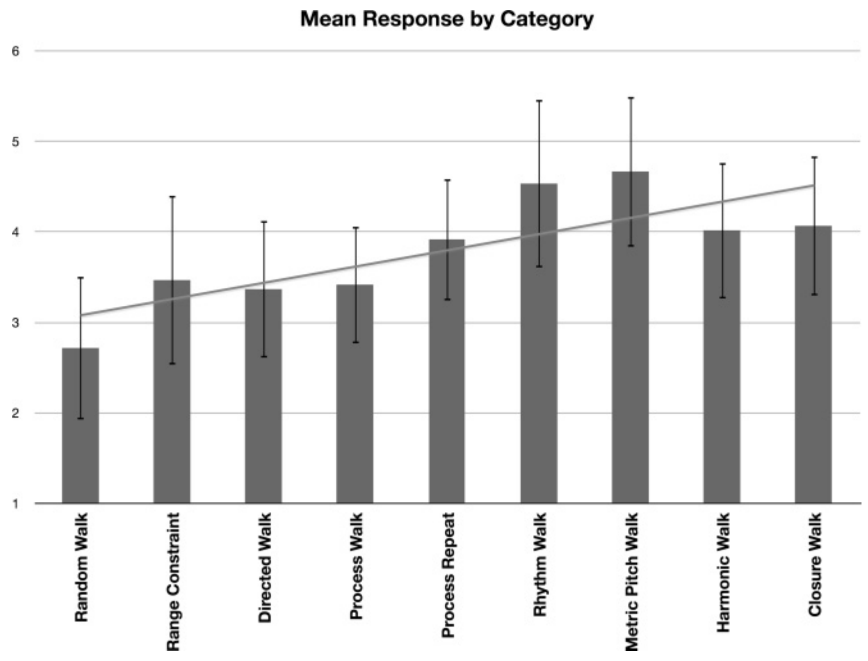

Figure 22. Results from the listening trials including the trend slope.

(a video of *Entanglement* is available online at vimeo.com/82061102). As a further check on their effectiveness, they were subjected to some external aesthetic evaluation.

Evaluation

In order to assess the effect of the incremental developments in the melody algorithm, examples of the generated output at each stage were recorded and evaluated by experienced musicians.

Three melodies were sequentially generated for each of ten stages of the developing melody algorithm. Audio examples of these 30 melodies played by computer on a digital piano were presented to participants in random order. These melodies can be accessed online at explodingart.com/cmj-melodies, where they have been categorized by stage. There were 32 participants, all of whom were experienced, adult musicians. Participants were asked to rank the melodies from poor to good on a seven-point scale.

The mean scores for melodies at each stage are shown diagrammatically in Figure 22. The stages follow the developmental phases of the algorithm outlined in this article. Although the data indicate that there is, perhaps unsurprisingly, some variety in the responses, there is a reasonably clear increase in score as the melody algorithm becomes more elaborate. There is a noticeable increase when rhythmic variety is included, again not surprisingly.

We interpret these results as confirming our own judgment that each stage of the algorithmic development makes a perceivable contribution to the melodic character. These results are only indicative, however, given the modest sample of melodies and number of participants involved.

Conclusion

This article has outlined a series of techniques for generative music that have been inspired by theories of music cognition, in particular on theories of proximity, goal seeking, good continuation, context sensitivity, and closure. They have been applied to the generation of melodic contour, metrical rhythms, and phrase endings. There has been some attention to harmonic movement but this could be elaborated further in future research.

The techniques are focused on symbolic music-making in a western tonal and metric setting. They have been demonstrated in the context of an algorithmic melody generator showing how they integrate with one another, and have been evaluated through listening trials that confirm a tendency for improvement as the techniques accumulate.

The presented implementation is succinct, and a balance has been sought between hiding complexity and exposing parameters necessary for expressive control. A number of the techniques have required the writing of utility functions to support their implementation. Where this has been necessary we have tried to maintain the balance of brevity and expressive control. In addition, these techniques, and their parsimonious implementation, have benefitted from our own research in modeling music intelligence and seeking empirical evidence for musicological theories in computational analysis of musical works.

We hope that this article may provide some insight into the richness of opportunity in applying music cognition studies to generative techniques and show how such techniques can be applied to succinct compositional processes and live-coding performance.

References

Bharucha, J. J. 1984. "Event Hierarchies, Tonal Hierarchies and Assimilation: A Reply to Deutsch and Dowling." *Journal of Experimental Psychology* 113(3):421–425.

Bigand, E., and R. Parncutt. 1999. "Perceiving Musical Tension in Chord Sequences." *Psychological Research* 62(4):237–254.

Bregman, A. S. 1990. *Auditory Scene Analysis: The Perceptual Organization of Sound*. Cambridge, Massachusetts: MIT Press.

Brown, A. R., T. Gifford, and R. Davidson. 2012. "Tracking Levels of Closure in Melodies." In *Proceedings of the International Conference on Music Perception and Cognition*, pp. 149–152.

Brown, A. R., and A. Sorensen. 2009. "Interacting with Generative Music Through Live Coding." *Contemporary Music Review* 28(1):17–29.

Cambouropoulos, E. 1997. "Musical Rhythm: A Formal Model for Determining Local Boundaries, Accents, and Meter in a Melodic Surface." In M. Leman, ed. *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology*. Berlin: Springer, pp. 277–293.

Clynes, M. 1983. "Expressive Microstructure in Music, Linked to Living Qualities." In J. Sundberg, ed. *Studies of Music Performance.* Stockholm: Royal Swedish Academy of Music No. 39, pp. 76–181. Available online at www.speech.kth.se/music/publications/kma/papers/kma39-ocr.pdf. Accessed November 2014.

Cooper, G., and L. B. Meyer. 1960. *The Rhythmic Structure of Music.* Chicago, Illinois: Chicago University Press.

Cope, D. 2000. *The Algorithmic Composer*. Madison, Wisconsin: A-R Editions.

Corsini, R. J. 1999. *The Dictionary of Psychology.* New York: Psychology Press.

Deliège, I. A. 1987. "Grouping Conditions in Listening to Music: An Approach to Lerdahl and Jackendoff's Grouping Preference Rules." *Music Perception* 4(4):325–360.

Deutsch, D. 1999a. "Grouping Mechanisms in Music." In D. Deutsch, ed. *The Psychology of Music*. 2nd ed. San Diego: Academic Press, pp. 299–348.

Deutsch, D. 1999b. "The Processing of Pitch Combinations." In D. Deutsch, ed. *The Psychology of Music*. 2nd ed. San Diego: Academic Press, pp. 349–411.

Fraise, P. 1984. "Perception and Estimation of Time." *Annual Review of Psychology* 35:1–7.

Goetschius, P. 1902. *Counterpoint Applied.* New York: G. Schirmer.

von Hippel, P. 2000. "Redefining Pitch Proximity: Tessitura and Mobility as Constraints on Melodic Intervals." *Music Perception* 17(3):313–327.

von Hippel, P. T., and D. Huron. 2000. "Why do Skips Precede Reversals? The Effect of Tessitura on Melodic Structure." *Music Perception* 18(1):59–85.

Huron, D. 2006. *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, Massachusetts: MIT Press.

Jones, M. R., and M. Boltz. 1989. "Dynamic Attending and Responses to Time." *Psychological Review 96*(3):459–491.

Krumhansl, C. L., and E. J. Kessler. 1982. "Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys." *Psychological Review* 89(4):334–268.

Large, E. W. 1994. "Dynamic Representation of Musical Structure." PhD dissertation, Ohio State University, Department of Computer Science and Engineering.

Large, E. W., C. Palmer, and J. B. Pollack. 1995. "Reduced Memory Representations for Music." *Cognitive Science* 19(1):53–96.

Larson, S. 2012. *Musical Forces: Motion, Metaphor, and Meaning in Music*. Bloomington: Indiana University Press.

Lerdahl, F., and R. Jackendoff. 1983. *A Generative Theory of Tonal Music*. Cambridge, Massachusetts: MIT Press.

London, J. 2004. *Hearing in Time: Psychological Aspects of Musical Meter*. Oxford: Oxford University Press.

Maxwell, J., P. Pasquier, and A. Eigenfeldt. 2011. "The Closure-Based Cueing Model: Cognitively-Inspired Learning and Generation of Musical Sequences." In *Proceedings of the Sound and Music Computing Conference*. Available online at www.smcnetwork.org/system/files/smc2011_submission_189.pdf. Accessed October 2014.

McLean, A., J. Rohrhuber, and N. Collins. 2014. "Editors' Notes." *Computer Music Journal* 38(1):4–5.

Meyer, L. B. 1956. *Emotion and Meaning in Music*. Chicago, Illinois: University of Chicago Press.

Narmour, E. 1973. *Beyond Schenkerism.* Chicago, Illinois: University of Chicago Press.

Narmour, E. 1990. *The Analysis and Cognition of Basic Melodic Structures*. Chicago, Illinois: University of Chicago Press.

Narmour, E. 1992. *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. Chicago, Illinois: University of Chicago Press.

Nierhaus, G. 2009. *Algorithmic Composition: Paradigms of Automated Music Generation.* Vienna: Springer.

Papadopoulos, G., and G. Wiggins. 1999. "AI Methods for Algorithmic Composition: A Survey, a Critical View, and Future Prospects." In *Proceedings of the AISB'99 Symposium on Musical Creativity,* pp. 110–117. Available online at citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.8064&rep=rep1&type=pdf. Accessed November 2014.

Pearce, M., and G. Wiggins. 2006. "Expectation in Melody: The Influence of Context and Learning." *Music Perception 23*(5):377–405.

Saunders, R., and J. S. Gero. 2001. "The Digital Clockwork Muse: A Computational Model of Aesthetic Evolution." In *Proceedings of Artificial Intelligence and the Simulation of Behavior,* vol. 1, pp. 12–21.

Sorensen, A. 2005. "Impromptu: An Interactive Programming Environment for Composition and Performance." In *Proceedings of the Australasian Computer Music Conference,* pp. 149–153.

Sorensen, A., and A. R. Brown. 2007. "aa-cell in Practice: An Approach to Musical Live Coding." In *Proceedings of the International Computer Music Conference*, pp. 292–299.

Temperley, D. 2007. *Music and Probability*. Cambridge, Massachusetts: MIT Press.

Wertheimer, M. 1938. "Laws of Organisation in Perceptual Forms" [W. Ellis, trans.]. In W. Ellis, ed. *A Source Book of Gestalt Psychology*. London: Dover, pp. 71–88.

Xenakis, I. 1976. "Foreword." N'Shima [score]. Paris: Salabert.

Xenakis, I. 1992. *Formalized Music: Thought and Mathematics in Music*. Stuyvesant, New York: Pendragon.