REPRESENTING AND AUTOMATING RHYTHMIC PATTERN TRANSFORMATIONS

Lonce Wyse
National University of Singapore
lonce.wyse@nus.edu.sg

Keith Tan
National University of Singapore
Keith@SonicDivergence.com

Peter Kellock
Independent researcher/composer
petekellock@gmail.com

ABSTRACT

In this paper, we describe a novel system for the composition and transformation of polyphonic rhythmic patterns - the Rhythmorpher. The Rhythmorpher makes automated and seamless transitions between precomposed segments despite the typically multiple and sometimes conflicting constraints the precomposed segments impose. The system is designed to address musical issues that arise in real time interactive applications such as computer games.

Rhythmorpher patterns are represented by high-level descriptive parameters. Events are generated in real-time for a percussion synthesizer so that patterns can be sensitive to external contexts and quickly respond to changes. Transitions between precomposed segments are interpolations of the high-level pattern description parameters which produce results very different from simple crossfades. Smooth transitions can be made between precomposed segments of different speeds, swing styles, instrumentations, and even time signatures, and can be executed over any duration of time.

1. INTRODUCTION

One of the most challenging environments for adaptive music is computer games. Game music is typically composed as a collection of "cells", each designed to accompany different game states which can number in the thousands. Because state durations are generally not known, cells can be either lengthy or designed to loop indefinitely. In either case, the challenge is that state changes can happen suddenly and without warning, necessitating musical transitions at arbitrary points in the composed segments.

Standard interactive game music composition systems support several ways of making transitions. Crossfades simply take a specified duration of time to fade one cell out as another fades in. This is not a very sophisticated way of fitting the two segments together. It is unable to handle tempo matching, for example, and so generally sounds jarring [5]. Another technique uses a collection of precomposed transition cells designed specifically for segues between particular precompsed segments. Since they are handcrafted, they can be sensitive to the musical characteristics of the segments they are connecting. However, they too can only be sensibly spliced in at specific points of time. Furthermore, the number of precomposed segments necessary goes up quickly with the number of possible adjacent game states and transition time flexibility requirements [3, 4].At costs of up to 1500 USD per minute of music, and potentially thousands of hours of listening time from players, a system that allows for the cheap and easy generation of music that responds well to changing contexts is essential.

The Rhythmorpher was developed to aid in the production and transition of polyphonic percussive music specifically in the area of game music. The high level parametric representations used in the Rhythmorpher are grounded in performance principals to facilitate the process of composing large amounts of music in a wide variety styles. Morphing between segments composed using the Rhythmorpher is simple and there are no restrictions on when and how long transitions can be made. A very large number of transitions are possible – one for every pair of precomposed segments, and transitions always reflect the individual characteristics of the segments they connect.

2. RELATED WORK

Agate [3] is a system that takes the game's environmental and musical data as input. A high level script containing music generation grammar interprets and translates the inputs into music. While Agate is reactive and allows for the composition of an infinite amount of dynamic music, its compositional method does not lend itself well to thematically driven musical styles nor does it provide an alternative to musical transitions other than cross fades.

FMod [1] and Jet [2] are both tools which allow for the mapping of musical events via programmed parameters. For example, both choose segments of music from a database depending on context, and both support precomposed transition segments. It is also possible to have a segment fade through a precomposed transition before reaching the destination track.

Although there are numerous systems which specialize in pitch based composition, recent years have seen an emergence of systems which focus on the generation of percussive music. Among the techniques used are genetic algorithms, intelligent multi-agents as well as Cellular Automata [9]. Some are based on genetic algorithms [6, 7] that allow for interactive user evaluations to find musically desired solutions. However, the use of a human in the loop creates a resource bottleneck [12] making them unsuitable for use in the games industry. While multi-agent systems such as

the Kinetic Engine [8] can be used to simulate drummers improvising within a percussion ensemble and hence provide a means by which composers can generate percussive music embedded in musical transitions between segments, the system has limited parametric control which makes it inappropriate for application in games.

3. THE RHYTHMORPHER

In its current implementation, the Rhythmorpher consists of 55 possible sample-based percussive instruments corresponding to the general MIDI set. It also provides an interface of high level parameters used to manipulate the instruments. These parameters can be broadly grouped into per-instrument parameters and ensemble pattern parameters. Per-instrument parameters can further be subdivided into those which generate the core rhythmic patterns and those which allow for humanization and variations of those patterns. Ensemble rhythmic pattern parameters include swing, time signature and the morphing control for transitions between preset rhythmic patterns.

A key goal in the design of the Rhythmorpher was the representation of rhythmic patterns in terms of high level descriptive parameters. This is the space in which interpolation between different preset patterns generates rhythms that lie perceptually between patterns at the interpolation endpoints. The same parametric representation is used by composers to author the preset rhythms and hence has to be intuitive and capable of representing any desired pattern. While control over the weight of every single beat division for every single instrument (as in a step sequencer) would allow for the creation of arbitrary rhythms, such a representation would not be a very capable or interesting space for pattern interpolation.

We have thus developed a representation wherein composers create polyphonic rhythmic patterns by manipulating a set of high level musically-descriptive parameters, each of which varies beat division weightage and timings across the entire pattern. The parameters used in the Rhythmorpher are familiar musical concepts and are described below.

3.1. The metrical hierarchy

The basis for the rhythmic pattern representation used in the Rythmorpher is the "metrical hierarchy" [13] in western music which is made up of a pattern of beats and beat division, each accorded a different weight in significance and generally in amplitude level. A series of 16^{th} notes within a 4/4 metrical context can be represented as in Figure 1 with the hierarchy index number 1 identifying the downbeat, 2 identifying the half-measure position, 3 identifying the second and fourth beat and 4 and 5 identifying the half-beat and quarter-beat positions respectively. The height of the graph points denotes arbitrary weights (here represented as velocity). This representation will be referenced throughout the paper.

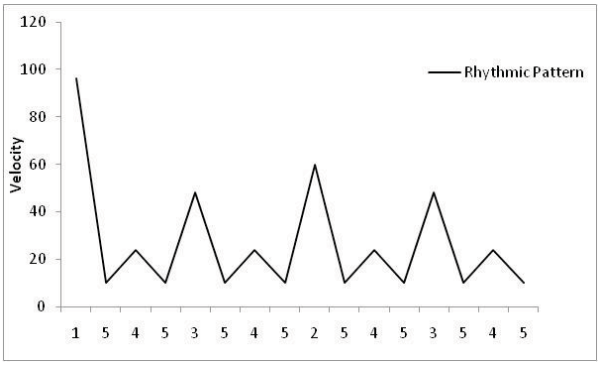

Figure 1. Numerical representation of the metrical hierarchy for 4/4 meter.

3.2. Per-instrument Controls

This section introduces the parameters that apply to the rhythmic patterns for individual instruments.

The parameter "Straight" influences the downbeat velocities of a rhythmic pattern, "Offbeat" influences the upbeat velocities of a rhythmic pattern, and "Syncopation" influences beat divisions that have onsets occurring at a low metrical level in the hierarchy (high number) and are sustained through to a beat in the metrical hierarchy that has a higher metrical level (low number). Figure 2 shows the difference in beat patterns that these parameters influence.

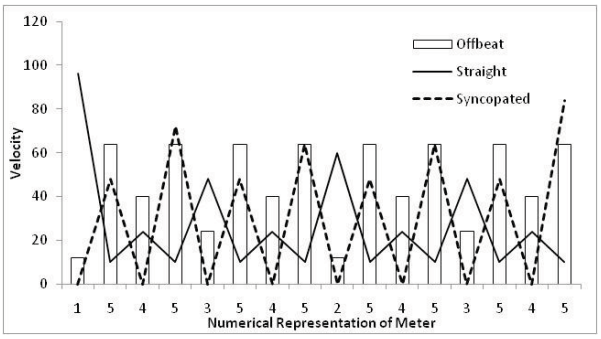

Figure 2. Beat emphasis patterns represented by the Straight, Offbeat and Syncopation parameters.

Figure 3 illustrates how the "Phase" parameter shifts the numerical representations of the metrical hierarchy with respect to the beat cycle. "Random" adds or subtracts a random velocity from each beat in the meter while "Mean" adds or subtracts velocities by a predefined amount across the whole pattern. "Ramp" as its name suggests, gradually increases and then decreases (or vice versa), the velocities of each beat over the course of a bar. "Threshold" specifies a minimum velocity beats in an instrument pattern must possess in order to be played by the system.

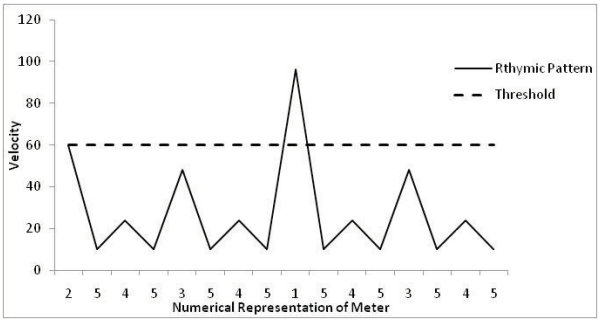

Figure 3. Phased rhythmic pattern with velocity threshold of 60. Note the shift in representation of meter compared to Figure 1.

3.3. Ensemble rhythmic pattern controls

Parameters described in this section influence the rhythmic pattern of the entire ensemble of instruments.

Time signatures can range over a variety of simple and compound signatures. A "Swing Ratio" parameter affects the relative timing of adjacent pairs of metrical durations that together constitute what we refer to as the "swing unit" by extending the first element and shortening the second while holding the duration of the whole swing unit constant [10]. Studies on swing [10, 11] have shown that the long/short ratio typically ranges from 1:1 to 3.5:1 with slower tempos tending to have higher ratios. Ratios in this range have been used in generative systems for creating swing [14].

The Rhythmorpher provides users with possible swing ratios ranging from 1:1 to 5:2 and the ability to choose swing units corresponding to eighth, quarter, half and whole notes. While it is uncommon for composers to choose swing units that are not factors of the time signature, for example a swing unit of a half note in a bar of 5/8 meter, we permit this in the Rhythmorpher by ignoring bar divisions if this case. Aside from providing more options to the composer, this simplifies interpolation between rhythmic patterns of different time signatures during morphs.

3.4. Applications of morph

The main feature of the Rhythmorpher is its ability to generate music that allows for seamless transitions between two rhythmic patterns that we call a "morph." Morphing refers to the process of generating a rhythmic pattern by interpolating between two sets of high level parameters (the original rhythmic patterns) and is different from a crossfade which is the gradual increase and decrease of volume between two audio signals. By moving the Morph parameter from one endpoint to the other, a transition between any two rhythmic patterns can be made over any transition duration.

3.5. Morphing ≠ Crossfade

Due to the interpolation in the space of the high-level parameters representing patterns and the non-linear relationship between the parameter values and velocities, a morph is significantly different to crossfade. For example, morphing sometimes results in transitions containing instrumental events that were not in either of the original rhythmic patterns. This is because the representation of velocities for all instruments across all beat divisions as well as the threshold for playing are in constant relative flux during a morph.

3.6. Morphing between different swing pattern

Adjusting individual beat durations is another factor which differentiates a rhythmic morph from a crossfade. If two preset patterns to be interpolated have the same swing unit (e.g. quarter note), then the swing value can just be interpolated over the values of the morph. A particularly interesting timing issue is the case of a "compound swing" that arises when morphing between two rhythmic patterns that swing over different swing units. In this case, the interpolated rhythm pattern timing cannot be computed directly from an interpolated swing ratio. Instead, time values of the events in the larger swing unit must be computed first. Once the longer unit event times due to swing distortion are known, the swing distortions for the finer beat divisions can be computed. Figure 4 illustrates how the morph is computed between patterns with swing ratios being interpolated on two different swing units simultaneously.

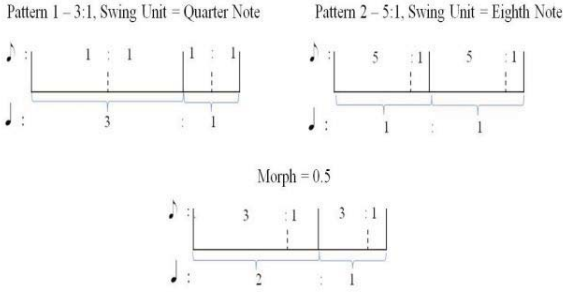

Figure 4. Pattern 1 has a swing ratio of 3:1 on the quarter-note unit only, while Pattern 2 has a swing ratio of 5:1 on an eight-note swing unit only. The compound swing time distortions for both the quarter-note and the eight-note swing unites are shown at each endpoint, and in the middle of the transition when the Morph parameter = .5.

3.7. Morphing between different time signatures

The Rhythmorpher also allows morphing between rhythmic patterns with different time signatures, but the implementation differs from the way other parameters are handled since interpolation of time signatures doesn't make sense.

Instead, two separate "morphed" rhythm patterns are computed across the parameters for which interpolation does make sense – one pattern for each time signature. Although the two different time signature patterns will be at constantly changing phases with respect to their

beat and bar locations, events are always synchronized at 16th notes, the smallest time division in the Rhythmorpher. Once the two different parametrically morphed patterns have been computed, velocity values can be interpolated between the two resulting patterns to generate a value for the final output. Although the sense of bar is lost in the middle of a morph between two time signatures, some sense and feel of the different time signatures does remain, and is correlated with the position along the morph.

4. DISCUSSION AND FUTURE WORK

Although we have been able to generate a wide variety of rhythmic patterns of different styles using the high-level parametric representation described, we have not yet been able to quantify the actual space of possible rhythms the representation covers. It would also make the Rhythmorpher more useful if patterns designed with common step sequencers could automatically be put into the representation used by the system. Finally, the next obvious major step would be to integrate the representation and generation of harmonic and melodic content so that full musical transitions could be automated.

5. CONCLUSION

We have designed and implemented a system where polyphonic rhythmic patterns can be created and seamless transitions be made between them despite radically different musical characteristics. The transitional patterns take on features of precomposed rhythmic patterns as they move from one to the other. Most importantly, the music produced during the automated transitions is never jarring and can be set to occur over any duration in real time.

6. ACKNOWLEDGEMENTS

This work was supported by project grant from the GAMBIT program of the Interactive and Digital Media Project Office, Media Development Authority of Singapore.

7. REFERENCES

[1] "Fmod - interactive audio middleware" http://www.fmod.org/.

[2] JET http://www.sonivoxrocks.com/jet.html.

[3] Slipgate Ironworks, "*A Generative, Adaptive Music System for MMO games*", Proceedings of the Game Developers Conference, Austin, 2008.

[4] C.Bajakin. "*Adaptive Music: The Secret Lies Within Music Itself*", Proceedings of the Game Developers Conference, San Francisco, 2010.

[5] J. Harlin. "Ahead of the Curve" *Game Developer magazine*, October 2008.

[6] A. Pazos, A. S. del Riego, J.Dorado, J. R. Cardalda. "Genetic music compositor", *Proceedings of the Congress of Evolutionary Computation 1999* (CEC'99) pp. 885-890, Washington, 1999.

[7] N. Tokui and H.Iba. "Music Composition with Interactive Evolutionary Computation", *Proceedings of the 3rd international Conference on Generative Art,* Milan, 2000.

[8] A. Eigenfeldt. "The Creation of Evolutionary Rhythms within a Multi-Agent Networked Drum Ensemble", *Proceedings of the International computer music Conference*, Copenhagen, 2007.

[9] A.R. Brown. "Exploring Rhythmic Automata", *Applications on Evolutionary Computing, volume 3449,* Berlin, 2005.

[10] A. Friberg and A. Sundstrom. "Swing Ratios and Ensemble Timing in Jazz Performance: Evidence for a Common Rhythmic Pattern," *Music Perception: An Interdisciplinary Journal* 19, no. 3 (April 1, 2002): 333-349.

[11] H. Honing and W. Bas De Haas, "Swing Once More: Relating Timing and Tempo in Expert Jazz Drumming," *Music Perception: An Interdisciplinary Journal* 25, no. 5 (June 1, 2008): 471-476.

[12] J. A. Biles. "GenJam: A Genetic Algorithm for Generating Jazz Solos", *Proceedings of the International Computer Music Conference*, San Francisco, USA, 1994.

[13] D. Huron and A. Ommen, "An Empirical Study of Syncopation in American Popular Music, 1890-1939", *Music Theory Spectrum* 28, no. 2 (October 1, 2006): 211-231.

[14] F. Gouyon, L. Fabig, J.Bonada, "Rhythmic Expressiveness Transformation of Audio Recordings: Swing Modification", Proceedings of the International Conference on Digital Audio Effects, London, 2003.