Douglas Geers

Research: Ripples

NOTE: Please also refer to the three other examples of my research shown on this website: Appliance, Juicer, and Treembre.

To the reader: This material is an excerpt from a larger paper, "Oblique Strategies," written in 2001 for *Current Musicology,* vol. 67-68 (in press.) Since this article was directed at a musical but non-computer music audience, this text exhibits a rather conversational style and explains even basic computer music concepts.

In the spring of 1997 I composed a computer music piece entitled *Ripples*.During the composition of *Ripples* I purposely decided to play with two ideas pioneered by composer Iannis Xenakis: stochastic algorithms and a "granular" approach to musical texture (both explained below.) Freely adapting Xenakis' concepts, I decided to create and implement an algorithmic system that would produce music according to rising and falling waves of note densities. Meanwhile, quite unlike Xenakis, I decided to map these ideas to a stable major scale pitch set, clearly pulsed rhythmic material, and a sound world highly influenced by techno music.

Put simply, a stochastic algorithm is a mathematical procedure that utilizes probability theory to influence its decisions. In music, this means that a composer can set up situations that require choices, and allow the computer to make these choices by following statistical distribution rules. For example, one could generate a primitive stochastic melody by instructing the computer to choose every subsequent note via the following rules: 40 percent of the time move down a step, 40 percent of the time move up a step, and 20 percent of the time repeat the same note.

For *Ripples*, I decided to use stochastics to control rising and falling densities of notes in time. From previous experience with computer music I knew that as a regularly pulsed stream of notes gets faster (denser in time), eventually one's mind switches from perceiving it as a series of individual notes to hearing it as a single continuous sound. This notion intrigued me, and I decided to explore perceptions of this boundary in *Ripples*. I conceived of each rise and fall of note densities as a wave of musical energy, and determined that during the piece these would grow and diminish in their extremity; thus the title *Ripples*. As I will explain more below, the final composition was created by connecting and layering many instances of this basic process, run with different input.

I wanted the music in *Ripples* to be pulsed and, at least initially, feel "performable." To accomplish this, I implemented the density changes by creating more and then fewer equal subdivisions ("tuplets") per beat, beginning with quarter notes, then eighth notes, then triplets, sixteenth notes, quintuplets, sextuplets, and so on. After the process reaches its "peak" number of tuplets, it then progresses back from the highest tuplet level to quarter notes once more. Each of these processes, from quarter notes to some x-tuplet to quarter notes again, would constitute one "ripple" (Figure 1.)

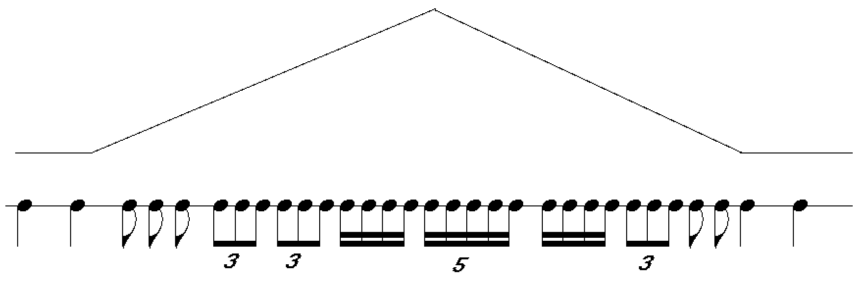

Figure 1: One simplifed, hypothetical instance of the basic *Ripples* algorithm

To realize the *Ripples* algorithm, I wrote my own software in the computer language C that did the necessary calculations and wrote a "score," which I then synthesized into audio using the computer music composition environment. Each time I ran my program, I set values such as tempo, the highest tuplet level to which I wanted the music to progress, a dynamic level for the beginning, a dynamic level to progress to, and the amount of time to get there and back. To give the ripples more shape, I soon altered the original plan so that each "ripple" would contain two density rises and falls: first to the specified peak tuplet level and back, and then to a second peak level and back again. I also added instructions for the music to move back and forth across the left-right stereo field as it played.

Rather than merely having one pulse of notes following this algorithm at a time, I devised my software to create a musical texture of sixteen musical lines, each at a different pitch level, all following the density wave simultaneously but stochastically offset from each other. In other words, I turned my ripple instrument into a sixteen-member ensemble of "ripplers." Moreover, since to me part of what makes instrumental music interesting is the fact that an ensemble of live musicians cannot possibly play perfectly in time together, I employed stochastics so that each rippler would perform slightly differently than its peers. For instance, although technically all the ripplers begin at the same moment, I actually wrote my program so that each one would manifest small random time offsets. In addition, I employed stochastics to individualize each of these ripplers even more, allowing each one to choose its own subset of pitches from the scale (described below), choose a unique time within a specified distance from the "ideal" time to reach its peak tuplet level, and similar stochastic values for the tuplet level to which it progressed, its dynamic levels, and how often it would insert a rest rather than playing a note.

I employed an E-flat major scale as my pitch class set throughout *Ripples*. Over the course of the two rising and falling density waves of each iteration of the ripple algorithm, each rippler chose notes from the scale four times. First, during the acceleration towards the first peak tuplet value, each rippler could choose only two pitch classes. Next, while slowing back from this peak to quarter notes again, each rippler was free to use all seven pitch classes. Then during the rise to the second peak, each was limited to three pitch classses, and finally during the second fall back to quarter notes each could choose from all seven notes once more. Each time the software chose subsets of the scale it do so entirely at random, though it did check so that it would not choose the same pitch class more than once for each rippler at any given time. Within each segment of music, once a rippler had chosen a set of pitches to use, the software chose from those at random. Thus, although the pitches were those of an E-flat major scale, the use of them was not designed to emphasize tonal relationships. As a result, sometimes one may perceive it as in E-flat major, at other times as in F Dorian, or even at times as C Aeolian.

I mentioned above that each of the ripplers was placed at a different pitch level, and would like to explain a bit further how this was done, and the consequences of my choices. The sixteen ripplers' pitch levels were spaced at the interval of an octave, from the octave C -3 (seven octaves below middle C) to octave C12, eight octaves above middle C. This pitch range may sound a bit crazy, and maybe it is; but it also demonstrates the joys of working with computer music, where one need not limit him/herself to what conventional instruments can do. The lower limit of human hearing is generally between C0 and C1, and the upper limit somewhere between C10 and G10. So my piece was constructed to go beyond these limits, and this produces interesting results, as I will describe next.

Because of the limitations of the 44,100 audio sampling rate I used for *Ripples* and a technicality of digital sound known as *foldover*, all pitches that were supposed to sound from around E-flat10 and above are misrepresented as lower non-tempered pitches (so, no, *Ripples* will not drive dogs crazy.) On the other end of the pitch continuum are notes that are too low for humans to hear; but as these notes play faster and faster tuplets and faster and faster tempos, they begin to be perceived as pitches in relation to the rate they occur (this happens at about 16-20 beats per second.) For instance, if a pitch of eight Hertz (approximately C —1) played sixteenth notes at quarter note=480--which is entirely possible with the *Ripples* software--then a listener would hear a pitch of 32 Hertz, which is very near C1. While I do not have the space to discuss these effects in more detail here, suffice to reiterate that they added interesting, relatively unpredictable pitch and timbral information to the music of *Ripples*.

Beyond this, I let me mention two other ways that I composed the timbre of *Ripples*. First, during my experiments early in the process of composition I intuitively created a particular spectrum that I found pleasing. Later, when the *RTcmix* software synthesized my scores, I instructed it to create each individual note of *Ripples* using this spectrum and an amplitude envelope.

Although my use of one spectrum for all of the notes in *Ripples* might seem simplistic at first, I knew that this would not be a problem because of the second additional way that I compose timbre in the piece. This method is directly linked to the "ripple" process that pervades the piece , and operates as follows: While *Ripples* begins with clear successions of notes, as the music proceeds the basic ripple process repeats and moves to more extreme realizations–faster and faster tempos. As the density of notes increases, eventually the sense of successive notes, each with its own timbre, dissolves into the perception of a single, complex evolving timbre, now the "line" of the music. In fact, the tempo of *Ripples* eventually rises to a maximum level of quarter note=720, so that even when each of the ripplers is only playing quarter notes it is already performing twelve notes per second!

Music such as this, in which thousands of very brief individual notes are combined linearly to form large scale audio events, is known as "granular synthesis." A simple analogy is that in granular synthesis each note has a role similar to that of each grain on a beach: it is an individual, yet a tiny element of a much larger structure. The concept of granular sound was first employed in composition by Iannis Xenakis, and has been used by many computer music composers during the last quarter of the twentieth century. My interest in *Ripples* was to in effect build a piece around the concept of granularity, allowing the music itself to trace a path from distinct single notes toward an increasingly granular manifestation and back again.

The basic shape of *Ripples* as a whole is the same wave of increasing and decreasing density realized in each individual ripple. As such, the essential slow-fast-slow pattern is easy to discern at the end of the work, and for the most part the musical development within seems smooth and organic. However, while the finished piece seems to flow quite naturally, it is actually the fruit of an involved compositional process.

To create the final composition, I developed my *Ripples* software until it gave me musical output that I found intriguing and satisfying. Then, I ran the program time and time again, altering the settings of the initial values repeatedly. The most significant settings that I altered were the tempo and peak tuplet levels for each ripple, and the piece employs materials at tempos ranging from quarter note=20 to quarter note=720. Each time I ran the program the software realized music within the parameters I had set, using stochastically weighted randomization so that no two runs of the program, even with the same settings, were exactly the same. I listened to these segments of music carefully and adjusted my program settings repeatedly, fine tuning the settings to move the results closer and closer to what I had in mind, until I found just what I wanted, or--sometimes--something even better.

Eventually I was able to create a large number of musical segments embodying a wide range of realizations of the *Ripples* algorithm, and chose to use only those I found most successful from them. I then ordered the chosen segments, edited them, and layered them into a composite mix according to my overall formal plan. I arranged the connections between them so that in the final piece not all of the sections manifest the entire rising and falling density pattern. In fact, most often the connections between sections are quite blurred, and a new iteration of the algorithm, at a new tempo, begins before the previous one has ended. Moreover, as in *Reality House*, sometimes processes are unexpectedly truncated, such as in the abrupt transition back to the opening material at 5 minutes, 35 seconds into *Ripples*. Finally, I processed each segment with several kinds of audio signal processing software (reverberation, flanging, etc.) to create more and more subtle timbral variations among the sections of the piece.

As the preceding paragraphs illustrate, my creation the *Ripples* software was clearly part of the act of composition, since the design of the program implemented elements of the work's form, its rhythmic and melodic/harmonic material, tempo, number of voices, and the relation of these voices to one another. However, it is also important to note that even though on one level the composition is intensely algorithmic, I exercised "rigorously intuitive" discrimination regarding which materials to choose and how to deploy them in the final composition. Among other things, I chose the tempo settings for each section, the peak tuplet values, the dynamic levels and their changes, the ordering of the sections, how the sections would join one another, when and how much segments would overlap, and how much of the entire sparse-dense-sparse process each segment would manifest in the finished piece.

Another aspect of *Ripples* worth mentioning is the harmonic structure. Essentially, the harmony in *Ripples* is static, employing a single set of seven pitch classes, those of the E-flat major scale, throughout (although as I mentioned earlier, factors such as *foldover* add unexpected pitches.) I composed *Ripples* this way both for practical and aesthetic reasons. First, my software had no mechanism to automatically change the pitch collection. To add planned harmonic changes to the piece would have required either significant further time programming the software or that I run it separately for each harmony, which would have greatly increased the time needed to realize the piece and would have disrupted the continuity of the musical processes. However, a further reason for the static harmony relates to my own interest in musical styles with little or no harmonic motion, including minimal pieces and Indian classical music. Since the concept explored in *Ripples* is the rhythmic/timbral progression, I felt that complex harmonic relations were not necessary and that the relatively stationary harmony was satisfying to me.

Other examples of my research include Appliance, a system for interactive performance with sculptures; Juicer, a realtime DSP instrument; and Treembre, a SGI/Linux application. The Treembre program, which is still under development, is an attempt to hierarchically organize elements of timbre and animate them in time.

- A brief bio of Doug
- Go to Doug's front page