Aesthetics, Score Generation, and Sonification in a Game Piece

Cristyn Magnus
CRCA, Cal(IT)^2, Department of Music, University of California, San Diego
La Jolla, Ca. 92093-0326
cmagnus@ucsd.edu

Abstract

This paper explores the motivation behind vs. computer*, a game piece for computer and solo percussion. It discusses the compositional decisions that were required to meet the aims of the piece. It also describes the technical implementation of the game itself, sonification of aspects of the game, and automatic score generation for variable instrumentation.*

1 Introduction

Vs. computer is a game piece for solo percussion and computer. It grew out of an interest in developing interactive pieces with emergent formal properties. This interest sprang from a joint background in cognitive science and music.

When I began thinking about *vs. computer*, I had just completed a piece for percussion and two vocalists. That piece, *Pantomime Grasshopper*, explored the changing relationships between performers as a basis for compositional structure. I wanted to explore similar dynamics in a performer versus computer situation. By removing other people, it becomes a situation of performer versus self—success is measured by personal satisfaction with the attempt; usually against previous attempts.

2 Game

My first decision was that the piece needed to be robust. I needed to design a sonic interface that would run without a second person watching the computer to make sure the patch worked. The moves also needed to be musically meaningful, since I wanted formal aspects of the piece to rise out of the act of game-playing. With these considerations in mind, I chose two reliably detectable dimensions: loudness and speed. The binary combination of loudness and speed give the game four possible moves: fast and loud, fast and soft, slow and loud, slow and soft.

I worked extensively with percussionist Robert Esler to find a viable game. We tried a four-move strategy game before settling on an arcade-style game. The four moves are mapped onto up, down, left, and right on a two-dimensional game field.

The game itself is very simple. The performer's task is to use his/her interpretation of the algorithmically generated score to move a green square around the game-field, picking up black squares before they disappear. The performer gets a point for each square he/she picks up and the computer gets a point for each square that fades out before the performer can pick it up. When squares are picked up or face out, new squares replace them at random locations. In a typical game, about ten squares are placed each minute.

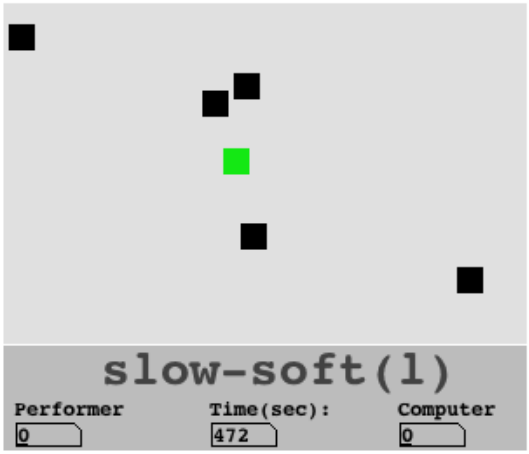

Figure 1: The game-field.

Since the piece is substantially less interesting if the performer can win too easily or can't win at all, he/she can adjust the difficulty to reflect his/her current skill. The game should pose sufficient challenge that the performer can win occasionally with significant effort. The default difficulty sets the life of the black squares to fourty-six moves; this is the minimum number of moves it takes to cross the game-field diagonally.

Moving works like a switch joystick rather than a variable resistance joystick. It doesn't matter how fast or slow the performer plays, the green square moves across the screen at

the same rate. No amount of louder or softer, for instance, will change the rate at which the green square moves. Even if it did, this would be irrelevant, since the life-span of a black square is counted in moves, not in time.

As soon as the computer detects a different movement direction, it will change direction regardless of when it was scheduled to repeat its last move. Since speed is calculated based on the time between notes, there can be a significant lag in determining a transition from fast to slow. It's a bit like driving a boat. I considered re-coding the slowness-detector to change direction as soon as a pause crossed a threshold, but I liked the notion of slowness really having a tangible effect instead of merely being another direction.

3 Score Generation

Another disideratum was instrumental flexibility. The exact instrumentation is unspecified for two reasons. First, the practice among percussionists is to modify instrumentation as part of the interpreting a piece (Hennies 2002), (Esler 2006), (Manzanilla 2004). It makes more sense for me to describe the features I want highlighted than to ask for specific instruments without explaining my motivation. Second, I want to allow versatility in performance. This piece is designed to work with instruments on hand. It can expand to an array of larger instruments at home or map onto a small setup on the road.

My thinking here was heavily influenced by Ivan Manzanilla's project of coming up with a versatile set of instruments to specialize on and to commission pieces for that setup. This would allow him to specialize as non-percussion instrumentalists do rather than use a different setup for each piece. Moreover, I wanted the piece to be playable on any percussionist's preferred improvisation setup. *Vs. computer* allows anywhere from three to twenty-two instruments. Although performers will probably tend towards somewhere in the 7 ± 2 (Miller 1968) range for individual instruments, this leaves open the possibility of playing on the piece on a single keyboard instrument and mapping the keys to instruments.

Since the moves are made by playing the score fast or slow, there is no way of knowing how much notated material will be required for a given game. If I had chosen to use a fixed score, there would have been two options: either allow the score to loop or make it so long that the performer could never conceivably reach the end. A looping score would have to lack direction so that looping would not disrupt the structure, or I would have to accept the possibility of the piece ending at an awkward structural point. Generating a score so long that it would never be played in its entirety would be a ludicrous waste of time, both for the composer and the performer learning the piece. The problem would be compounded because I would have to create scores for all possible numbers of instruments or sacrifice the goal of flexible instrumentation.

To acommodate the requirement for instrumental flexibility and length variability, the *vs. computer* score is generated algorithmically. First, a *seed score* is generated by nested markov processes. Then, a developmental algorithm produces new material by sampling and modifying earlier parts of the score.

The seed score is generated hierarchically. First, the number of *phrases* is determined. Gestures are ordered within phrases by a markov process. Finally, each gesture is itself algorithmically defined.

There are five kinds of gestures. Since tempo is one of the dimensions of game control, I wanted each gesture to articulate time in a different way. The two extremes are outlined by single-note gestures. The longest space between notes is twenty times the length of the shortest space between notes. The other three gestures are multi-note gestures. First, a "melodic" gesture of several notes whose randomly-selected length is twice the length of the shortest possible note-range. Second, an n-tuplet gesture plays several notes of comparable length that are of equal length. Third, an entire *accelerando* gesture lasts as long as the longest possible notes, but its constituent notes move from rather long to extremely short (or vice versa) over the course of the note.

In both single-note and multi-note gestures, notes are assigned instruments and loudness by markov chains. The instrument and loudness markov chains are independent. Each gesture-type has a unique probability table. The gesture's probability table determines the first note of the gesture based on the last note of the previous gesture, even if the previous gesture is of a different gesture-type. In order to apply instrument choice to a variable number of instruments, the probabilities are defined in terms of large proximity categories rather than a different probability for each instrument. Each note has some probability of doing one of four things in relation to the previous instrument. It can *repeat*, pick a *neighbor*, pick a *near* (non-neighboring) instrument, or pick a *far* instrument.

The seed material is developed by a process of sampling and altering earlier parts of the notated material. Sampling accounts for the hierarchical structure, but is not completely subject to it. Preference is given first to beginning and ending at phrase boundaries, then gesture boundaries, then note boundaries. There is a slight possibility that a sample will interrupt a note. The sample is appended to the end of the score with randomly selected notes altered in length, instrument, or loudness.

The developmental process has two effects. First, each time a portion of notated material is sampled, its odds of be-

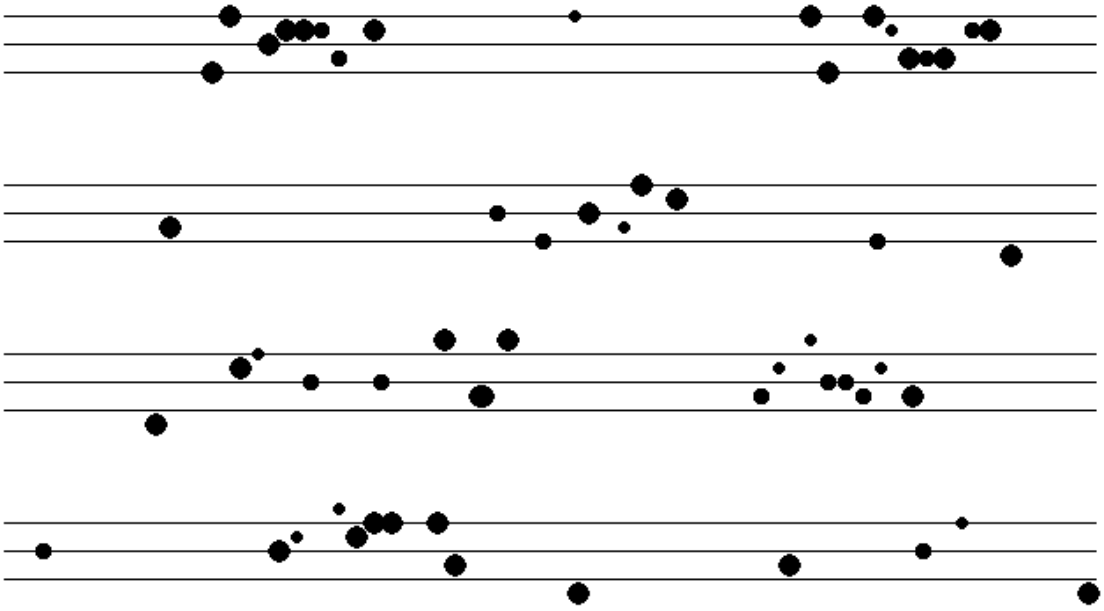

Figure 2: A few lines of sample score generated for seven instruments.

ing sampled again are higher relative to unsampled material. This is because samples come can come from any existing material, not just the seed material. This has the effect of elevating some materials to recurring, developing themes, while other materials become subservient. Second, phrase and gesture boundaries are priviledged but it is possible to have samples that interrupt notes or gestures and define them as new phrases. This means the material will become increasingly fragmented over the course of the piece. The combined effect is like reexamining a memory: examining it over and over, obsessing on some details and forgetting others until they achieve a signification they did not originally posess.

4 Sonification

The game is not visually projected for the audience, since this would be distracting. Instead, the state of the game is reflected in electronic processing. Each time the performer gets a point, a sound that rapidly increases in pitch is played; each time the computer gets a point, a sound that rapidly decreases in pitch is played.

The point distribution is reflected in the amount of processing the performer's sound undergoes. If the performer is well ahead of the computer, subtle effects like reverb are applied to his/her sound. As the computer does better, the amount of reverb will increase. Then increasingly intrusive effects are added; first pitch shift, then delay (again, of increasing intensity). If the performer loses, the settings on the processing will become sufficiently extreme that the performer's sound will be swallowed up in gritty, aggressive sounds.

If the performer wins, the computer's processing will be completely turned off. The performer will then continue to play the next portion of the score, free from the constraints required to make moves in the game. He/She should play until he/she reaches a satisfying gesture on which to end the piece. Unless the game ended at the end of a page, he/she should find an end point on the current page. Otherwise, he/she should end somewhere on the next page.

The time left in the game also affects processing. There is a multiplier that reflects the amount of time left. There is always less processing early in the game, and processing reflects the point distribution with increasing accuracy over the course of the game.

The game-field itself is projected spatially onto the audience. The processed sounds are spatialized to reflect the location of the green square. The sounds made by the computer getting points are spatially located analogously to the location of the black squares that just faded out. For flexibility, the piece works with 2, 4, 6, or 8 speakers. Note, however, that a stereo projection will lose the vertical dimension of the game.

5 Interaction between the game and notation

The form of the piece emerges from the interaction between the performer, the notation, and the game. This piece is a form of algorithmic composition in which the algorithm explicitly requires distributed cognition (Hutchins 1996). In this case, the algorithm is distributed between the computer and the performer—each is following the rules for its rôle in the game, but the outcome is music. It is not structured improvisation, since a performer playing the game in good faith will not be improvising. Rather, he/she will be making the most strategic choice possible at any given moment. As Xenakis notes about algorithmic composition: "In the long run [it] will follow the laws of probability and the performances will be *statistically* identical with each other" (Xenakis 1992). That is, because of the constraints provided by the game, roughly the same form will emerge at each performance.

The form of the piece will always be blocks of juxtaposed material that are produced by the slow/fast soft/loud binary pairs. Because the game-field does not wrap around, the performer can never get away with avoiding a particular way of playing. Because new tokens are respawned whenever old tokens are removed from the game, there are always new reasons for the player to keep moving and changing direction. The electronic portion of the piece will follow a more varied trajectory, depending on the difficulty the game is set to. But the general form will usually emerge in which processing begins gently, then increases in degree, varies in either direction over the central portion of the piece, then either stops (if the performer wins) or takes over (if the computer wins).

The difficulty in the game lies in interpreting the score so that the computer understands the performer's moves. The score deliberately contains extremes to create tension between the notation and the move the performer is trying to make. To play slowly, the performer must play slowly enough for the densest passage is perceived as slow; to play quickly, the performer must play fast enough for the sparsest passage to be perceived as fast. The performer isn't allowed to change his/her speed and loudness mappings once he/she has chosen to move in a particular direction. For instance, he/she shouldn't compensate for dense passages by playing a *ritardando*. The performer either needs to explicitly change his/her movement direction and play slower next time he/she attempts to move or accept any incidental direction change the computer might perceive and continue with the same mapping. The performer is allowed, however, to fool the computer by playing anything that is legal under the notation. For example, if the performer finds that a sustained attack on a particular instrument is often picked up as multiple, fast attacks, he/she could play a sustain during an extremely sparse section of notation to trick the computer into thinking he/she is playing faster than the notation suggests.

6 Acknowledgements

Vs. computer was written in collaboration with Robert Esler, who is not only an extremely talented, committed, and insightful performer, but also infinitely patient in the process of developing the technological aspects of the piece. I also owe a great deal to P.D. Magnus, off of whom I bounced most of my ideas and from whom I got a great deal of useful feedback. I would also like to thank Philppe Manoury and Miller Puckette for comments while I was writing the piece. *Vs. computer* was implemented using Miller Puckette's PD software (Puckette 06).

References

Esler, R. (2005–2006). Personal communication. `http://robertesler.com`.

Hennies, N. (2002). Personal communication.

Hutchins, E. (1996). *Cognition in the Wild*. Cambridge Massachussets: MIT Press.

Manzanilla, I. (2004). *Programming and performance in contemporary percussion music : a performer[']s exploration of how, why, and when*. Ph. D. thesis, University of California, San Diego.

Miller, G. A. (1968). The magical number seven, plus or minus two: Some limits on our capacity for processing information. In R. N. Haber (Ed.), *Contemporary theory and research in visual perception.*, pp. 187–202. New York: Holt, Rinehart, & Winston.

Puckette, M. (downloaded 2005–06). Pd–0.39–0 and pd–0.39–2. `http://crca.ucsd.edu/~msp/software.html`.

Xenakis, I. (1992). *Formalized Music: Thought and Mathematics in Music*. Hillsdale, NY: Pendragon Press.