Distance Music I–VI
For any Number of Programmer/Performers and Live, Programmable Computer Music Systems

LARRY POLANSKY

Contents

Introduction

The following pieces are intended for any group of performers/programmers, each with sound-generating, fully programmable general-purpose computer systems. In these pieces, the programmers are the performers as well, and the instruments used are small, sound-generating computer systems of any kind.

Each of these pieces may be altered, embellished, or enhanced by the programmers/performers to incorporate their own ideas, as long as the general intent of the piece is left intact. These pieces are intended as algorithms for a small "class" of formal musical events, each involving in some way the notion of deliberate, machine-aided morphological transformation.

The pieces in *Distance Music* require some form of "coordinated" programming between the performers. That is, each performer must be aware of the details of the other performers' software realization of the idea, and may have to alter their own software to interact in as coherent a way as possible with the other software. What is desired is a unified group realization that may or may

not involve different software on different machines, not several independent realizations of the idea played simultaneously.

Technical Requirements

The minimal requirements for the computer-music systems used are:

—Simple programming capability in some high-level or assembly language

—Some form of real-time sound generation, in which amplitudes, frequencies, and some aspects of timbre (but not necessarily complex envelope or steady-state wave generation) can be controlled. Any synthesis facility may be used, from the simplest to the most complex.

—Some form of user-definable input device (computer or instrument keyboard, mouse, joystick, knob, etc.)

—and optionally:

—simple data-passing facility between the performers (MIDI, serial or parallel port, and so on). High-speed (audio) data passing is not necessary, but if this is technically possible, this facility may be exploited in many of the pieces by a simple conceptual extension of the formal idea (explained in the pieces themselves).

Definitions

Shape: any ordered set of data, in n dimensions. A shape's dimensions are generally considered to correspond to musical parameters; e.g pitch, loudness, duration, and so forth. However, dimensional data might also represent more subtle parameters (for example "brightness"), indices to a finite set of musical event data (e.g. a list of possible pre-defined envelopes), or musical information on a larger level (such as a profile of distances from some source shape).

Metric: a function which takes two arguments and returns one positive number, indicating some concept of "distance" between the two arguments. The standard mathematical definition of a metric is used here (including symmetry, the "triangle inequality," and that "the distance between x and y equals 0 if and only if $x = y$)." In *Distance Music*, metric functions are often used between shapes. For example, a simple such function between two one-dimensional shapes (X, Y) of equal length

might be the average of the absolute values of the differences between corresponding intervals in the two shapes:

$$d(X,Y) = \frac{|(|(X_i - X_{i-1})| - |(Y_i - Y_{i-1})|)|}{n}$$

—where X and Y are two one-dimensional shapes with length n; and X_i and Y_i are the ith points in each shape.

I. "The Metric System" (for Charles Ames)

Pick a common "melody," some n–dimensional shape, called the source shape whose dimensions are musical parameters—durations, pitches, loudnesses, timbres, etc.

Each performer designs a distance function, or metric, which accepts two shapes as input, and returns some measure of perceptual distance between those shapes.

Each metric should define some class of perceptual equivalencies, or invariants, under its determination of distance. For example, a simple (but powerful) metric might only consider the *direction* of successive intervals along a given dimension of shape. For this metric, all shapes whose points "moved" up and down in the same order in that dimension (regardless of the magnitude of these intervals), would be regarded as perceptually equivalent.

Two shapes might be considered to be equal under one metric, and radically different under another. Thus, a metric, to a large extent, defines the perceptual space in which we are to judge the distances on shapes. The performers should design their own metrics in this piece so as to contrast, coincide, or complement the perceptual effects of the other performers' metrics.

Metrics may take into consideration any or all of the shape's dimensions (musical parameters). They may integrate any or all of the dimensions of the shape into one measure of multi-dimensional distance, or they may ignore all but one dimension.

All of the performers begin with the same source shape. Over the course of the piece, the performers successively generate shapes which are at a greater and greater distance from the source shape, under their individually designed metrics. At some point, on a visual cue, the generated shapes begin to move closer and closer to the source. Thus, the form of the piece is a simple arch, a morphological transformation from, and returning to, a common shape.

Generated shapes may be randomly generated in real-time, or chosen from a pre-compiled list. If computing shapes in real-time involves substantial delays between the sounding of those shapes, those pauses may be considered part of

the piece. If several performers are involved, some sort of distributed processing system might be used to generate new shapes according to the metrics in real-time, with one computer generating the information for the other computers to play.

The piece ends on a visual cue after all performers have arrived back at the source shape.

One possible variation on the form of this piece is as follows:

Ia. "THE METRIC SYSTEM" (VARIATION)

Performers begin with shapes distant from the source, and move towards the source. This version is, then, the second-half of "The Metric System" (I), in terms of form but not necessarily duration.

II. "CONCEPT TRACES" (FOR DAVID ROSENBOOM)

Each performer picks a different source shape, and designs a metric so that successive transformations of the source shape yield shapes which are closer to the source shape of another performer. That is, the newly generated shapes must satisfy two criteria—they must be further and further from the performer's shape, and closer and closer to the shape of another performer. All other aspects of this piece are as in I ("The Metric System").

The piece ends when all performers have arrived at the shape of another performer.

III. "DRAWING UNNECESSARY CONCLUSIONS" (FOR PHIL BURK)

This piece is a variation on II.

Each performer picks a shape, and graphically "draws" it on the computer monitor. Some simple method of altering that shape graphically in real time is needed (cursor keys, mouse, A-to-D conversion, even simple use of the keyboard), so that the performer can slightly "redraw" the shape while it is being sounded. If multidimensional shapes are used, there must be some method of selectively displaying and simply editing the various dimensions of the shape.

Each performer draws a picture of his or her own shape (on paper), and gives it to one of the other performers. Shapes may be n–dimensional, and the "hard-copy" pictures should show, in some simple fashion, all the dimensions of the shape.

Over the course of the piece, each performer continually and slowly redraws

their shape on the screen until it looks like the shape of the performer whose picture they have on paper. The piece ends when all performers have arrived at their new shapes.

IV. "The Roots of Learning" (for John Bischoff, Jim Horton, and Tim Perkis)

The performers pick a common *found* data source, such as a segment of computer memory whose contents are fixed for the duration of the piece (like library files, the operating system, ROM, and so forth). This data source is sonically interpreted in some simple fashion (e.g., sets of four bytes are used for pitch, duration, loudness, timbre). Another example would be to modulate a simple waveform by the data source. All performers should play this data in as similar a way as possible, so that the resulting sound is a kind of unison, or as close to one as possible given differing machine configurations. The manner of playing the data should be as simple as possible, and should avoid any attempts at making the sound more "musical." If possible, all machines should use the same data; e.g. one computer's ROM could be used as a common lookup table.

The piece begins on a visual cue, the performers attempting to synchronise the machines manually as much as possible. By some input device (keyboard, mouse, and so forth), or algorithmically, each performer then changes the *starting location* of the data list throughout the piece, so that although the same information is being scanned by all performers, a canon is caused by different lengths and inceptions of lists. When the end of the list is reached, the sonic interpretation returns to the current, computed starting location of the data.

Two endings are possible for the piece. In the first, the performers eventually return, in some fashion, to the original starting point of the list, and end on a visual cue. In the second, one performer signals the end at a point of considerable disjunction between the various performers.

V. "The World's Longest Melody" (for David Feldman)

The performers play a shape whose next parametric values are always determined as simple increments or decrements, or ratios, from the previous ones. In other words, a generalized "step-wise" melody. The magnitude of the interval, in each dimension, should be fixed (whether it is ratiometric or linear). However, the intervallic magnitude in each dimension is left up to the individual performer. For example, in a pitch dimension, it might be a tempered semitone, a just ratio of any size, or a constant frequency value (say, 30Hz).

Whether the value in a given dimension is *incremented* or *decremented* by this

interval to produce the next value, is determined by a probability that the next step will be in the same direction as the previous. That is, if the pitch rose for the previous event, this probability determines the likelihood of it continuing to rise. One might refer to this probability as the "up/down" probability along a given dimension.

A probability of 1 means that the "sign," or direction of an interval in one dimension will always be the same (for example, the pitch continues to rise, or durations get longer and longer). A probability of .5 means that there is equal probability of the interval being in the same or different direction as the previous. A probability of 0 means that there is no probability of the interval being in the same direction as the previous.

The up/down probability begins at .5 and gradually moves to 1 over the course of the piece, increasing at a steady rate. Thus, the piece moves toward a monotonic directional motion. Each performer may assign a directional meaning, for each dimension, to the probability of 1. The dimensions may be independent. For example, the pitches might get higher while the durations get longer, or vice versa.

The performers should try to apply this algorithm to as many parameters of the sound as possible, including (but not limited to): duration, pitch, loudness, steady-state waveshape deformation (either point by point or by some algorithm which works on spectral amplitudes), time values of envelope transients, and indices and frequencies of steady-modulators.

Three simple formal variations are:

1) the probability begins at 0 (no probability of the same direction) and moves to .5 (equal probability), and then to 1 (certainty of same direction) over the course of the piece;
2) the probability begins at 1 (certainty of same direction) and moves to .5 over the course of the piece;
3) the probability begins at 1, and moves through .5 to 0 over the course of the piece

If several performers are involved, the choice of variation, parameters, and interval magnitudes may be completely independent.

VI. "Simple Actions" (for Daniel Kelley)

Each performer designs several very simple ongoing musical events, called "actions." The criteria for simplicity is something like: "the sonic process may be completely described in about one sentence." Thus glissandi, crescendi, simple stochastic melodies (or simple stochastic generation of parameters in any

dimension), and random waveform generation, are all examples of "simple actions." Actions should be designed so that they can be turned on or off, but once on, they execute repeatedly until turned off.

These simple actions should have relatively simple "control" variables associated with them. For example, the speed at which an action executes (such as the general tempo of a stochastic melody) should be changeable by the performer in real-time, by some simple stimulus-response aspect of the program (mouse, cursor keys, knob, and so forth). As many simple control variables should be associated with as many simple actions as possible. These controls might include: range and means of stochastic event generation, tempi, probabilities for accelerandi and decelerandi, depth of modulation, and directional probability of glissandi and other dimensional interpolations.

(Note: Obviously, for this piece, some sort of simple software multi-tasking must be devised, so that each action executes continuously. This might be as simple as embedding all actions in a BEGIN ... UNTIL structure, and having each action check some stimulus activated variable for its on/off execution status. This piece was originally designed to run in the PERFORM stimulus/response environment of the computer music language HMSL, developed by the composer, Phil Burk, and David Rosenboom. In HMSL, complex activation, deactivation, and control of actions is possible in real-time. However, "Simple Actions" is quite possible to implement using a very simple scan-loop type of software structure).

A limited number of sound generation "channels" or *voices* is used in each computer. Each action should be possible in all voices, but the controls should be applicable to a single voice. That is, it might be possible to have a glissando or a random melody occurring in a voice, but the tempo control, associated with that voice, should affect the rate of both of these actions, in that voice only. Each action should be exactly the same in each voice.

The software is written such that it is always possible to activate any or all existent actions for a given voice. In other words, the performer should be able to initiate, say, both a glissando and a stochastic melody in one voice. However, the software should deliberately avoid trying to compensate for "interference" between two actions in one voice. The actions, when superimposed, should distort each other freely, in the following way.

Actions of the same voice share data as much as possible. If two actions have an "idea" in common, they use the same data. For example, both a glissando generator and a random interval generator would likely need a variable for "previous pitch" (so that the next pitch could be derived). This variable should be shared by both actions. This means that whenever two actions are executing simultaneously, there is as much "parameter passing" between them as possible. Another way of stating this is that each action's parameter passing to *itself* should be interfered with as much possible by any other action in that voice. Data should not be shared, however, between voices.

The programmer should have three goals in the designing of the software for this piece:

1) that there be as many actions as possible

2) that the actions be as simple as possible; that is, actions should be able to be trivially explained ("rising pitch," "random waveform," "pick one of two loudnesses randomly," . . .)

3) that the actions within a given voice be as *synergetic* as possible; no action should have any of its *own* data, and the more shared data in the network of actions in a voice, the better.

The programmers may use as many voices as the machine allows.

The piece consists of the performers turning on and off actions, and changing their controls. It may be of any length. A musical intent of the piece is to investigate the ways in which these simply definable "critters" (thanks to Robert Marsanyi for this term) alter each other's behavior, and combine into more complex musical organisms. Over the course of the piece, the performers should explore the informational and sonic ecology of the configuration of actions as much as possible.

One possible variation of the piece is as follows:

VIa. "SIMPLE ACTIONS" (VARIATION)

If the various computers can communicate with each other, actions between computers should share data as much as possible in an extension of the way these actions do within voices. One method for doing this is to have each computer identify its voices with that of another, so that a given voice might be shared by every computer in the performance. For example, if each computer can generate four voices, that is how many *total* voices should be used in the piece. All data applicable to "voice n" should be shared by any action on any computer in "voice n." Once again, a maximal synergy between actions is desired.