reprinted from *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics* vol. 3, pp. 2772-77

Interfacing sound synthesis to movement for exploring high-dimensional systems in a virtual environment.

Insook Choi and Robin Bargar
National Center for Supercomputing Applications
Beckman Institute, 405 N. Matthews, Urbana, IL 61801, USA
email: ichoi@ncsa.uiuc.edu

ABSTRACT

An auditory feedback plays an important role in our cognitive process. We want to make use of this sensory feedback when exploring high-dimensional systems. We present a *manifold interface* as a generalizable tool featuring three human performance abilities, to see, to move, and to listen. We visualize control parameter space the domain of which corresponds to the domain of input gestures. The gesture input is extended to a 3D visual representation of the control space in which one can efficiently apply multiple parameter variation techniques by performing movements. The consequences of parameter variation are computed in real-time sound synthesis with respect to the systems under study so that one can listen to the changes in the systems as they vary their motions. The paper introduces the working vocabulary *control path, window space, out-of-time displacements,* and *in-time displacements.* Applications of the interface are briefly discussed.

1. INTRODUCTION

Sounds as an auditory percept are little understood in terms of their potential to play a significant role for research and creative projects particularly involving high-dimensional systems. The applications of sound in interface design include voice recognition, teleconferencing, audio archiving, sound localization [1], audio alarms, audio cues, earcons [2], and data sonifications. Many of these applications serve the purpose of enhancing visualization or compensating for visual overload. For example audio cues are sounds for location identification guiding visual search for a point of interest. Among these applications data sonification comes close to utilizing auditory percepts for enhancing an understanding of data.

To bring auditory percepts into research projects, we investigate (1) designing sounds for an optimal representation of systems' behaviors, and (2) incorporating sounds in interactivity. As a system output, sound functions as an auditory feedback, linking full circle in an exploration process for observers to monitor their own interaction as well as the behavioral changes of systems under study. For exploring systems observers often encounter cumbersome tasks such as entering control data by typing or creating input files. Then output data are observed often in the form of numbers or graphic representations. When exploring high-dimensional systems one seeks for alternative ways of interacting with the systems. An efficient method for entering control data with real-time observation of the consequences are keys to an intuitive exploration. The use of sounds has been observed to offer efficient and perceptive learning in massive parameter space. We will discuss this application in section 4.

The unique characteristics of sounds lie in the omnidirectional characteristics of acoustic signals. This characteristic can be understood in two ways. First, the obvious meaning of "omnidirectional" refers to the way sounds propagate in space. This accounts for the physics of sounds such as diffusion, reflection, and diffraction as well as our perceptual ability to process the spatial distribution of sounds. Secondly, the term "omnidirectional" can be understood from a compositional point of view focusing on acoustic materials or elements, their pitch and rhythmic relationships, their sizes in units and groups. In other words, we can also apply "omnidirectional" to refer to classes of sounds within a *material differentiation space*. By listening to the way classes of materials are derived from an original set and developed through or without transitional states, one achieves a dynamical observation. An example can be found in [3] where the acoustic material differentiation is based upon the content area of an "information space".

2. ARCHITECTURE AND IMPLEMENTATION

The manifold controller (MC) is a set of C++ classes linking graphics, hardware input devices, and sound synthesis engines. MC can be defined as an interactive graphical sound generation tool and composition interface involving computational models; computational models may be sound synthesis models, composition algorithms, or any other numerical models such as chaotic systems. Its application is scalable from immersive virtual environments to desktop workstations. The manifold interface provides graphical lines and surfaces as an interface to manifolds of greater than three dimensions. The interface allows us to navigate in a high-dimensional parametric space from a visual display having a continuous gesture input system with at least two degrees of freedom. Our current implementation includes 3D gesture input and 3D display. For workstations supporting 2D controllers and 2D graphical display the references can be scaled down.

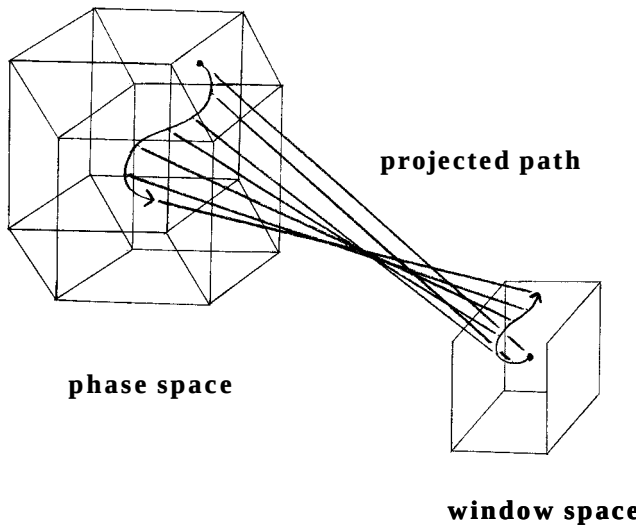
The graphic interface is linked to the NCSA Sound Server [4] which provides a real-time sound synthesis and algorithmic composition environment as well as non-real-time synthesis for demanding computational tasks. We focus on implementing functionalities for experimental production rather than for analysis. The architecture of the Sound Server allows the interface to be used concurrently for low-level control of synthesis engines and high-level control of composition parameters.

2.1 Organization and representation of control parameter space

Control parameters and all combinations of them involving computational models such as complex systems present a massive space to explore. We seek for efficient system access by organizing control parameters so that one can easily manipulate them into different combinations with rapid annotation capabilities to keep track of sequences of actions. Also we want the representation of the systems to have visual simplicity while maintaining an accuracy of its relationship to the actual states of the systems. This visual simplicity is an important factor to engage observers in an intuitive exploration.

2.1.1 Control, phase, and window spaces In organization and representation of control parameter space we distinguish three spaces; control space, phase space and window space. We use the term control space on a conceptual basis to implicitly refer to both phase and window space as a couple, whereas the terms phase space and window space have special meanings in terms of technical relationships. By the *phase space* of a system we mean the traditional n-dimensional Euclidean space where points--n-tuples of real numbers--correspond to states of a parameterized system. The phase space is all the permissible combinations of parameter values of an algorithm where trajectories of input gestures are encoded. A literal presentation of high-dimensional phase space will be visually undifferentiable resulting in the loss of orientation. Thus we need a representation space with data reduction from arbitrary high-dimensional phase space to 3D space in perceptible form. We call this represented phase space a *window space.* The window space defines how a three-dimensional visual representation is embedded in the high-dimensional phase space (Figure 1). We conceive of a three-dimensional visual display as a window onto the manifold so that an observer inputs changes to the system through the window space. An observer may effectively control the window space by panning and zooming in phase space.

Figure 1.
Embedding a window space



In our implementation the window is displayed in the CAVE environment [5] or a similar 3D view on a workstation. The cursor is placed in window space and it responds to gesture-input devices such as a wand or mouse, and to voice and keyboard commands. A window space provides a domain for generating and modifying classes of control point sets. These points represent combinations of parameter values as user-specified, and they are associated with particular sounds. This association of the sounds in conjunction with positional orientation in window space enhances the ability to identify boundaries where character shifts occur in states of the system.

2.1.2 Window space To define a window space users choose a small set of points in a phase space. We call these points *generating points.* We want to be able to visit these points and move smoothly between them. Since phase space may involve twists and bends during the embedding process we want the embedding to be continuous and "simple" while preserving a maximum amount of information. For data reduction from phase space to window space, we use a genetic algorithm (GA) to find a near-optimal window space by starting with a random population of possible solutions and allowing the better solutions to "multiply" and create offspring. For maximal representation of the structure in a phase space, in the region of generating points we apply a fitness function and a bit-representation of a solution as we interpret "structure" as the matrix of Euclidean distances between points. The states the GA explores are sets of points in the window space, represented as vectors of fixed-point numbers; the fitness function measures the error between the original distance matrix and the matrix for a particular set of points in window space.

The image of the generating points in the window space is extended to a 3-dimensional lattice where lines through the generating points are more or less parallel to the principal axes of the space. All points in the lattice are then used in a reversal of the previous GA to produce a corresponding lattice of similar geometry in the phase space. To map one point in the window space to the one in phase space, first the lattice cell where the point belongs has to be searched. Then its coordinates in the cell is found based on a tetrahedral decomposition of the cell (Figure 2). The corresponding cell and coordinates in the phase space define the resultant point in the phase space. The inverse map is computed similarly. As a point's cell-coordinates exist and are unique under certain conditions which the cells satisfy (convexity, noncoincidence of corner vertices), this map from one space to cell-coordinates and back to another space exists and is bijective. As the map is a patch of linear functions continuously connected, it is continuous as well.

To smooth out the map's nondifferentiable "edges", we are investigating the use of high-dimensional splines, in particular, cubic B-spline volumes built on a perturbation of the 3-dimensional lattice in the product of the phase and window spaces. In a Euclidean space, given a sequence of control points $\{p_0, \ldots, p_n\}$ and an index parameter *u*,

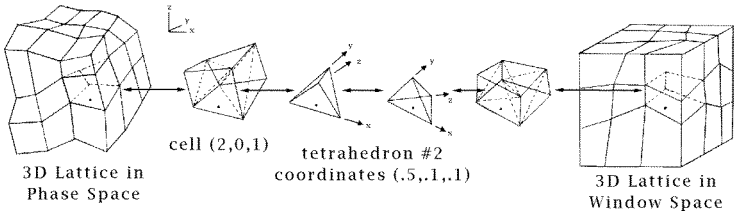
$$P(u) = \sum_{k=0}^{n} p_k N_{k,t}(u)$$

defines the B-spline curve for these control points, where $N_{k,t}$ are the standard B-spline blending functions, polynomials of degree $t-1$ [6]. We use cubic splines, hence $t = 4$. Given a 3-dimensional lattice $\{p_{j,k,l}\}$ of control points, we define its associated B-spline volume by

$$P(u,v,w) = \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} \sum_{l=0}^{n_l} p_{j,k,l} N_{j,4}(u) N_{k,4}(v) N_{l,4}(w)$$

over the index parameters *u, v, w.* Since we want generating points to map onto their images in the window space, we perturb the original lattice in the product of the phase and window spaces with another GA to find a lattice whose use as a set of control points for a B-spline volume will yield this exact mapping. This search takes a long time to compute, because the GA's fitness function evaluates this spline equation for many values. The inverse computation is slower still, that of finding index parameters *u, v, w* which correspond to a given point in the product space (equivalently, in one of its two component spaces). However, once these indices are found they immediately provide the mapping between the component spaces without any linearizing steps such as the tetrahedral decomposition of a lattice cell. We are searching for ways to do this in real time.

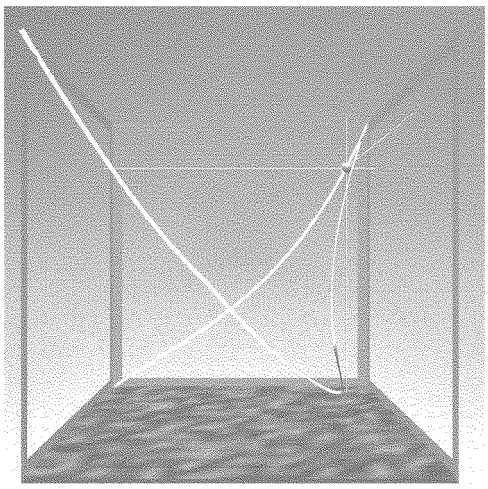
Figure 2.
Bijective map between Phase space and Window space.



2.2 Paths and data management features

Using a hardware input device such as a wand permitting three degrees of freedom in VR, making arm gestures an observer may draw traces in window space (Figure 3). We will refer to these traces as *paths*. The path is a sequence of positions of a cursor in the window space which correspond to the movement of a wand, thus scheduling the state changes in the system. The cursor position in the window space then maps to a point in a phase space through a callback function. A path through a phase space is a mapping from some time interval $[0, t_{Max}]$ to the phase space. This map need not be bijective or continuous; a path can cross itself, or make abrupt jumps. The path is stored in the phase space, not in the window space. Thus a sequence of points of the path is defined with respect to the high-dimensional manifold, and its projection is defined with respect to the particular window space being used.

Figure 3
A view of a path in a window space



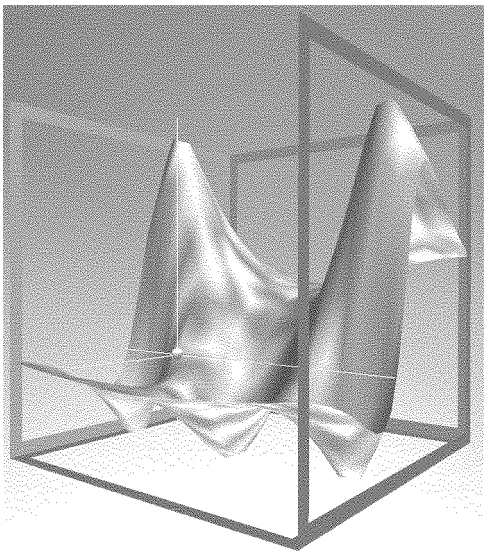
A path is initially recorded as a set of (n+1)-tuples, points in the Cartesian product of the n-dimensional phase space and one-dimensional time. This raw data is smoothed prior to be stored as a C++ path object. The smoothing is done by approximating the original path through this (n+1)-space with a sequence of spline curves. These splines are also in time as well as in "spatial" dimensions and are computed in the high-dimensional space. This smoothing is also done with a GA, where the bit vector representation of a sequence of spline segments is a vector of fixed-point control points and the fitness function approximates a least-squares error measure integrated over the original path.

In short we can say the path is drawn through a window space and encoded in a phase space. One of the important attribute of paths is a record/retrieval functionality which stores and returns temporal information as well as positional information from the user's activities. This enables users to reflect and revisit the previous decisions or movements in a time critical manner.

2.3 Surfaces and fiducial points

On a workstation where a desktop mouse is inherently incompatible with three-dimensional control, we draw surfaces in the window space and constrain the cursor to the surface, thus compromising with the locally two-dimensional behavior of the mouse (Figure 4). Paths can then be recorded on the surface by gestures in two dimensions. The concept of *surface* is also useful in a 3D environment to provide regional differentiation with explicit geographical representation of subsets of control space.

Figure 4.
A view of a surface in a window space

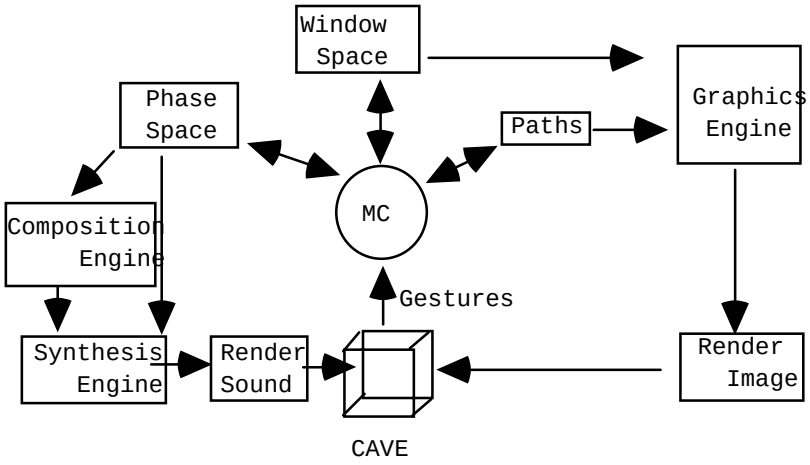


In order to create surfaces we first investigate the window space and find a position producing interesting results. This position is an initial *fiducial point* which can be linked to other fiducial points in the window space by generating a surface through the points, creating a terrain of control values. The terrain may be altered in shape by trimming edges, changing the position of a fiducial point or adding or removing points. Currently we constrain a plane to displacements in the direction perpendicular to its surface, this being the most direct conversion of the control region of a 2D mouse to three spatial coordinates. More complex surfaces containing folds or intersections may be incorporated as control surfaces by enhancing the 2D navigation constraints of the mouse, for example with keyboard commands for navigating in a third dimension. However, hybrid interface solutions that require interrupted movements in hand and arm gestures are cumbersome and intrusive to a user's concentration.

Several methods may be employed for creating a smooth surface among a set of fiducial points. Splines are familiar tools in the graphics industry for constructing desired curved lines or surfaces. One drawback to their application for manifold surfaces is their potential need for multiple control points for assigning smoothing coefficients to a curve. By adding control points we lose the one-to-one relation between fiducial points and surface-defining points. To preserve this relation we use an averaging-

displacement method for arriving at a smooth surface, passing through all the fiducial points and requiring no other surface-defining points. Beginning with a set of fiducial points defined by two planar dimensions and one perpendicular dimension, we apply the perpendicular displacement value of each fiducial point at regular intervals across the planar surface. The displacement at each interval across the surface is determined by weighting the displacement according to the distance from each fiducial point to a surface location, and averaging all of the weighted displacements for that location. This procedure is repeated at regular intervals across the surface. This procedure is not applicable if we permit two fiducial points to share the same planar coordinates with different displacement values. The architecture of the MC control flow can be summarized as a double-loop feedback and rendering cycle (see Figure 5).

Figure 5.
The control flow of the MC environment.



3. EXPLORATION MODES

The model of an observer in exploration modes with the MC is a model of a performing listener. Listening to sounds generated by her or his preceding movements, an observer's cognitive responses are in turn reflected in her or his next movements. Thus each movement is an articulation of decision-making based upon the evaluation of a preceding acoustic consequence and a proposition for the next.

Navigating a high-dimensional manifold with the MC provides an alternative approach to conventional scientific investigations where all but one dimension is held constant while the remaining dimension is varied. An alternative paradigm for exploration modes in a high-dimensional manifold is in several ways akin to learning a musical instrument. (1) Nonlinearity--the interfaces such as a keyboard on a piano, or valves and fingerholes of wind or brass instruments have nonlinear relationships to the vibratory system states as well as the acoustic results, yet one can learn to perform control gestures to extract desired results. (2) Intuitive orientation--explorers do not need to attend in detail to the dimensions being varied and how, since this information is encoded by the window space embedding prior to exploration. They can concentrate on grasping an intuitive orientation with respect to the control space. (3) Applicability for unpredictable skills--musical instruments are available for those whose skills vary from novice to virtuoso. A virtuoso is an expert of an instrument by an understanding of its physical properties. She or he knows how to enter motion control to the system in order to achieve desired sounds as well as how to apply acquired listening skills to continuously diagnose the states of the system. (4) Global orientation--it is worthwhile to note, when observing novice performers' learning processes, it is more efficient for them to learn an instrument by grasping its whole physical space rather than trying to gather a performance sense by investigating one key or one type of bow stroke at a time. After this global orientation there will be time for refining individual movements in relation to particular keys or strings for extracting desired tone quality. An easy scalability of control parameter space enables explorers to choose their own orientation scope until they acquire the ability to rapidly fine-tune relations among control variables to achieve desired system states.

3.1 Orientation, experiment, and production

The maturity stage of an observer's interactivity with the systems can be described by three stages: orientation stage, experiment stage, and production stage. Each stage has its heuristic value and an observer gains an insight and understanding of the systems while stepping through the stages. Descriptions of these stages are based upon our experiences and reports from beta tests, and we wish these are suggestive to adopt alternative and creative ways of exploring computational models.

During *orientation stage* explorers investigate the whole control space by assigning attributes to the axes of the window space. Finding a region of interest, she or he refines the scope of the window space by specifying minimum and maximum boundary values of the attributes. Once a good scope of window space is decided an observer can experiment with fine details of the space by choosing generating points, by specifying surfaces and fiducial points, and by encoding paths.

In *experiment stage* explorers learn the temporal sensitivity in state changes of the systems with respect to the sense of speed of their own motions as well as the spatial sensitivity affected by resolution according to the size of the grid in control space. Having found acoustically relevant regions and paths at this exploratory stage, the paths can then be subjected to rigorous experimentation. Gaining a certain degree of familiarity, one can pursue unusual tasks for intermediate experiments. displacements can be performed on source paths by applying transformations such as translation, rotation, augmentation, and diminution. Translation and rotation affect the values of parameters, not the temporal content of the source path. Augmentation and diminution will affect temporal content as well as parameter values, altering either the rate of change or the duration.

Quick and systematic generation of displacements can be performed in two ways. (1) Out-of-time displacements can be achieved by applying transformation rules to the source paths to generate batches of files in non-real time. The results are available for real-time review through window space. (2) In-time displacements are generated by real-time encoding along with the source path. While a source path playback is initiated as an accompanied event, one can detach the cursor from the path and use it to send additional control messages to generate a variation to the original. This is analogical to the way a jazz musician generates material in jazz performances. Only, in jazz one cannot backtrack whereas here we can backtrack all the sources and origins and their relations. For other examples of unusual tasks, MC provides functionality to bundle several paths and initiate them simultaneously so that an observer may experience polyphonic retrieval of her or his previous gestures. During this

retrieval one may also record yet another path and study the acoustic deviations. By the time an observer steps through all these stages she or he is an expert of the window space and ready to go to *production stage* where she or he decides what data and paths to keep or to discard, documents them, and script them as desired. These are subjected to further refinements and analysis.

Whereas out-of-time displacements offers a systematic approach to generate variations and real-time reviews, in-time displacements offers a large variety of playfulness. We find the latter case as informative as the former: imagine an explorer starts a second path in conjunction with a source path, and applies displacements as time passes and observe the differences of the two paths in duets while controlling the degree of deviation. This would be a powerful way to generate a pair or a group of modifications with intended degrees of deviation since our ears are good at evaluating fine details of deviations and variations.

4. APPLICATIONS

In this section we discuss three applications: a physically-based models, a simulated resonance, and an algorithmic musical pattern generation.

4.1 Multi-dimensional bifurcation scenarios in a simulated chaotic circuit

Traditional studies of chaotic systems observe bifurcation scenarios by identifying a single bifurcation parameter and varying its value for generating changes in a state of a chaotic system. The Chua's circuit belongs to the class of simplest electronic circuits which can generate chaotic signals [7], and is one of the few known experimental chaotic systems which can be modeled numerically and in computer simulations [8]. Following preliminary experiments with a Chua's circuit for observing acoustic properties of attractors [9], we observed many states producing interesting sounds can not be achieved by the variation of a single parameter. Using a numerical simulation of the Chua's circuit implemented as real-time oscillator in the NCSA Sound Server, a multiple-parameter variation technique can be applied from the MC to continuously vary the voltage values of simulated circuit components [10]. The resulting trajectories of parameter values generate bifurcation scenarios producing acoustic signals that are informative concerning the state of the circuit and are potentially interesting for musical composition. The MC may also be applied to an experimental voltage controlled Chua's circuit for generating composed sequences of states and bifurcations to produce signals for real-time musical performance [11].

4.2 Dynamically controlling vowel synthesis

CHANT synthesizes sound from a description of frequency spectrum characteristics and a simulation of the output of an excitor-resonator system [12]. CHANT waveforms require the specification of 7 parameters for each formant in the spectrum. For best results the spectrum should vary over time. We installed the CHANT libraries in the NCSA Sound Server, allowing the manifold interface to generate CHANT sounds in real time. To define a window space we associate specific sounds with specific locations - generating points - in the window space. Configuring a window space for rendering a CHANT waveform required 4 steps:

1. Identify sets of formant parameter values for specific vowel sounds.
2. For each vowel, associate its formant parameter set with a unique 3D position in a window space, creating a generating point.
3. Compute the embedding such that all points in the window space have acoustic properties consistent with those of the generating points (smooth transitions occur between generating points).
4. For the examples in Figure 6, create a path in the window space that visits each generating point.

For these examples we rendered three formants, requiring 21 parameters. We decided to hold some parameters fixed while others varied along the control path. For each generating point we defined 8 parameters: the center frequency and bandwidth of the first formant, and the center frequency, bandwidth and amplitude of formants two and three. Four generating points were created; each was assigned a unique vowel sound (/u/, /i/, /e/, or /a:/) and each vowel point was positioned at a unique corner in the window space. Amplitude is measured in dB and center frequency and bandwidth in Hz.

Using the same points as path control points, a path was created passing once through each of the vowels. Signals from five locations on this path are presented in figure 6. Intermediate positions on the path produce intermediate vowel sounds, such as the /U/ which occurs in a location toward the center of the window space. In figure 6 the cursor on the floor is positioned so that its vertical axis intersects the path at the point of the intermediate vowel, /U/.

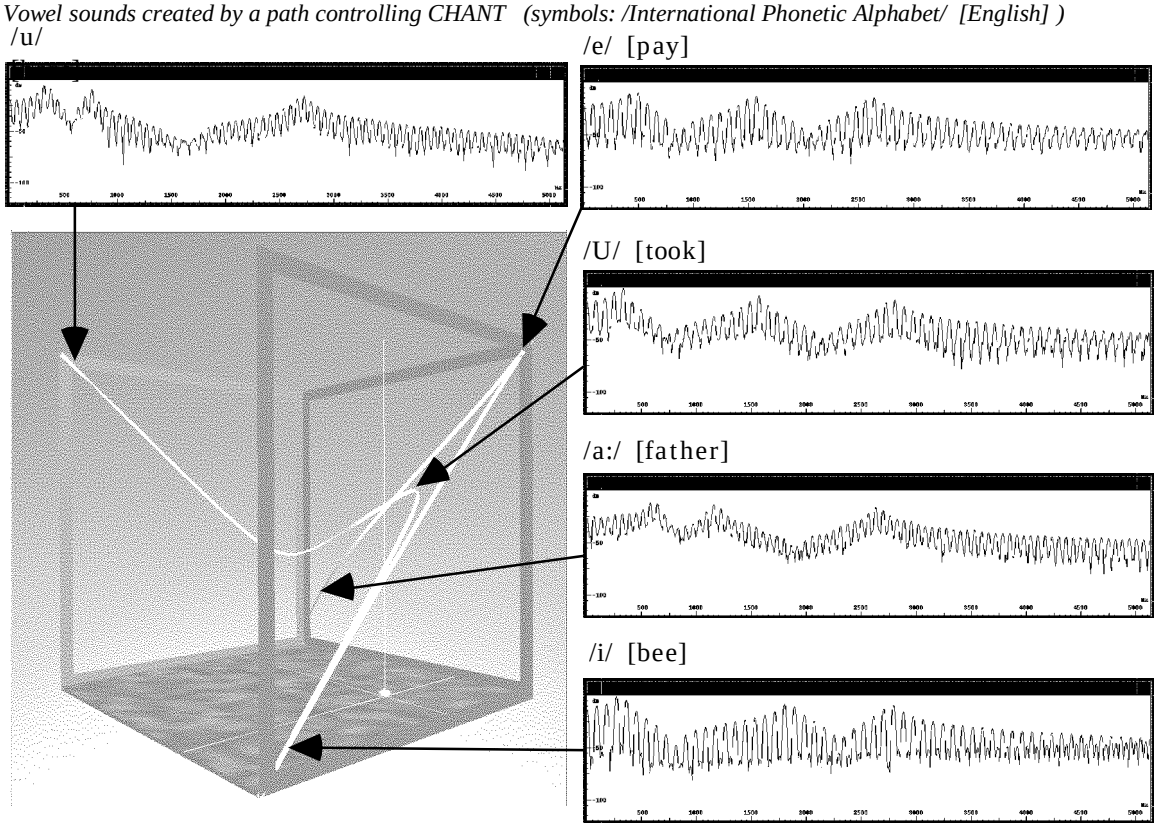
4.3 Transformation of musical patterns

Our previous examples control simulated signal generators for producing complex tones. We may also apply the control in larger scale to signals structured of series of discrete acoustic events. Messages such as music or speech organized in streams which we parse into phrases, words, motives, and notes, are a class of signals to which we devote most of our listening attention in daily life. Composed patterns provide an auditory interface with 2 desirable features: an acoustic background helping listeners to make comparisons among auditory messages, and (2) a balance of repetition and change helping listeners to remain attentive to significant changes without tiring their ear by attempting to give equal attention to every detail. In a recent interface prototype we associate positions in 3D space with transformation of musical materials determining rhythm, pitch and instrument according to positions on three axes [4]. With the MC, musical patterns from the 3D location interface can be further differentiated into the six axes: melodic pitch sequence, melodic rhythm sequence, melodic instrument, tempo, accompaniment, harmonic sequence, accompaniment metric pattern. These elements are independently transformed and combined to create unique patterns at regularly-spaced intersections in the six-dimensional space.

5. CONCLUSIONS AND FURTHER PROJECTS

Freed from complicated control tasks, explorers can concentrate on observing system behaviors as well as their interactivity with the system. Control paths provide a form of gesture-based notation. We can treat them as virtual control signals, exported from the manifold interface and stored in files. Thus the concept of a human movement as an exploration of a system can be formalized in a data structure. A capability to encode and formalize control signals suggests a study of control signals with respect to the systems in which they are generated. Such a study

Figure 6.
Vowel sounds created by a path controlling CHANT (symbols: /International Phonetic Alphabet/ [English])



may result in further definition of the human inquiry process in an interactive interface.

A good definition of the window space is critical to all aspects of the Manifold Controller. There is an inevitable information loss as we reduce dimensions. The nature of the information loss affects the size and shape of the manifold region we can control from a window space. As this problem is difficult and impossible to solve precisely (at least when the phase space has more than 3 dimensions), we want to pursuit ways to define maps which retain more information than our current choice of a GA and fitness function.

6. ACKNOWLEDGMENTS

We wish to thank to Camille Goudeseune for his implementation of the GA in section 2, Bryan Holloway for the CHANT implementation, and Ami Choi for figure 5.

7. REFERENCES

1. E. M. Wenzel, "Localization in Virtual Acoustic Displays", *Presence: Teleoperators & Virtual Env.* V. 1, No. 1, 1992, pp. 80 - 107.
2. M. M. Blattner, D. A. Sumikawa, and R. M. Greenberg, "Earcons and Icons: Their Structures and Common Design Principles", *Int. Human Interface*, 1989, No. 4, pp. 11 - 44.
3. R. Bargar and I. Choi, "Sound Assemblage for Navigating Distributed Information", this volume.
4. R. Bargar, I. Choi, S. Das, C. Goudeseune. "Model-based Interactive Sound for an Immersive Virtual Environment." *Proceedings of the International Computer Music Conference*, Aarhus, Denmark, Sept. 1994, pp. 471-474.
5. T. Defanti, C. Cruz-Neira, D. Sandin, R. Kenyon, and J. Hart. "The CAVE". *Communications of the ACM*, V. 35, No. 6, June 1992.
6. Hearn, D., and M. P. Baker. *Computer Graphics*. Englewood Cliffs: Prentice-Hall, 1986.
7. L. O. Chua, C. W. Wu, A. Huang, G. Q. Zhong, "A universal circuit for studying and generating chaos, part I: Route to chaos," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, V. 40, no. 10, October 1993.
8. P. M. Kennedy, "Three Steps to Chaos, Part II: A Chua's Circuit Primer," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, V. 40, no. 10, October 1993, pp. 657 - 674.
9. G. Mayer-Kress, I. Choi, N. Weber, R. Bargar, A. Hübler, "Musical Signals from Chua's Circuit," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, V. 40, No. 10, October 1993, pp. 688 - 695.
10. I. Choi, "Interactive Exploration of a Chaotic Oscillator for Generating Musical Signals in Real-Time Concert Performance." *Special Issue: Chaos and Nonlinear Dynamics, Journal of the Franklin Institute*, June, 1995.
11. G. Q. Zhong, R. Bargar and K. S. Halle, "Circuits for Voltage Tuning the Parameters of Chua's Circuit: Experimental Application for Musical Signal Generation," *Special Issue, Chaos and Nonlinear Dynamics, Journal of the Franklin Institute*, June, 1995.
12. X. Rodet, Y. Potard, and J. Barriere. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General." *Computer Music Journal* V. 8, No. 3, Fall 1984, pp. 15-31.