Ali Momeni* and Cyrille Henry†
*Center for New Music and Audio Technologies
University of California at Berkeley
1750 Arch Street
Berkeley, California 94709 USA
www.cnmat.berkeley.edu/~ali
ali@cnmat.berkeley.edu
†11 Rue Jaques Kablé
75018 Paris, France
www.chdh.net
cyrille.henry@la-kitchen.fr

Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis

The work we describe here is motivated by a desire for intimate and expressive control over creative processes implemented in real-time performance software. We seek a manner of control that offers a "low entry fee with no ceiling on virtuosity" and allows expressive control of musical and visual control structures (Wessel and Wright 2001). Like many colleagues, we believe that the answer lies in enriching the approach to mapping (Winkler 1995; Rovan et al. 1997; Arfib et al. 2002; Hunt et al. 2002).

Correspondence between sound and image is an incredibly rich area of exploration ranging from psychoacoustic research to cinema and most recently to the world of computer-based media. Our interest in the latter category motivated a survey of what repertoire and tools explore interaction between sound and image in real-time work. Most works can be characterized as (1) sound-to-image, (2) image-to-sound, or (3) concurrent generation of sound and image. The first two categories represent a unidirectional relationship between sound and image. In sound-to-image applications, audio analysis provides control information for image synthesis and manipulation. Perhaps the most common examples of such an approach is the music visualization feature present in many commercial music library managers, such as Apple's iTunes (www.apple.com) and Nullsoft's Winamp (www.winamp.com). In programs such as these, analysis of audio streams creates often psychedelic moving images.

In image-to-sound applications, image analysis provides the means for synthesis and manipulation of sound. Pioneering works in this field include that of Lesbros (1996) on image-to-sound mappings and the software application Metasynth (released in 1997 and most recently updated for Mac OS X; see www.uisoftware.com/PAGES/acceuil_meta.html). Metasynth uses a mapping scheme between sound and image that considers time on the horizontal axis and frequency on the vertical axis of the source image. Manipulating the source image—a kind of score for the generated sound—results in changes in the synthesized audio. Many other interesting examples of image-to-sound and sound-to-image paradigms are found in areas ranging from computer-based installations and performances—including Golan Levin's *Messa vi Voce* (http://tmema.org/messa/messa.html), Mark Domino's recent contributions to the Silk Road Project (Domino 2004; www.silkroadproject.org), Bas Van Koolwijk's *FD-BCK AV* (www.clubtransmediale.de/index.php?id=1090), and Tanaka's *Bondage* (www.xmira.com/atau/bondage)—to data-mining applications (Hunt 2005) and medical applications like Meijer's *vOICe* system for the visually impaired (www.seeingwithsound.com/voice.htm). Our research falls in the last category: the concurrent generation and control of audio and video from a third, independent and variable data set.

This article summarizes our efforts to ease the exploration of rich gesture-mapping techniques applied to creative work with generative real-time audio and video instruments. We describe an approach to mapping that involves an independent layer of algorithms with time-varying behavior that is affected, explored, or observed in some way with gesture. We begin by introducing the concept of dynamic, independent visual-mapping layers; we then step back and briefly discuss the roots of this work to clearly contextualize it and its future directions. We conclude by discussing some of the advantages of our proposed systems and providing a number of examples.

Computer Music Journal, 30:1, pp. 49–66, Spring 2006
© 2006 Massachusetts Institute of Technology.

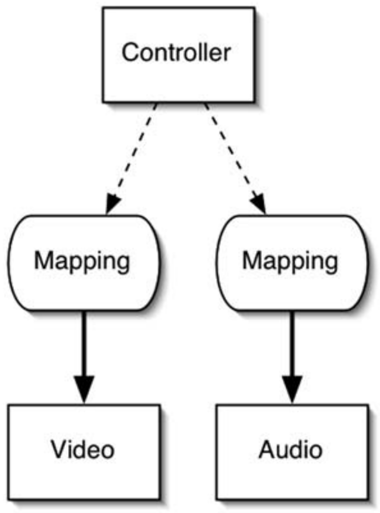


Figure 1. Block diagram of the flow of data from a controller, through a mapping stage, to the audio/video synthesizer.

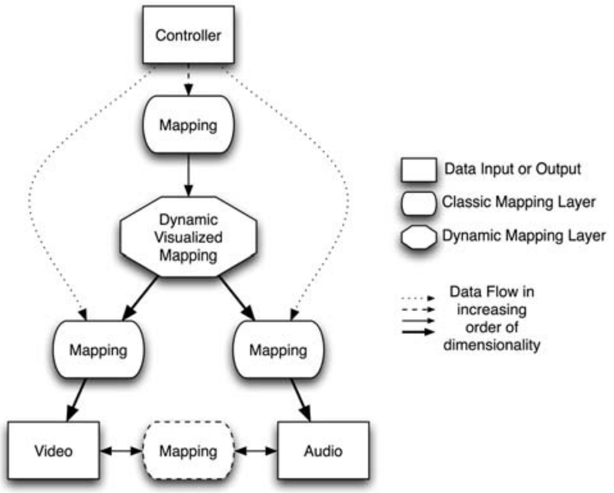


Figure 2. Block diagram of the data flow from the controller to the dynamic mapping layer and on to the audio/video synthesizer. Classic mapping layers connect the controller to the dynamic mapping layer and the mapping layers' outputs to the audio/video synthesizers. Because the dynamic mapping layer concurrently drives the audio and video instruments, an intrinsic correspondence is created that can be explored in the classic mapping layers.

Description of a Dynamic, Independent Visual Mapping Layer

Our notion of a dynamic, independent, visual-mapping layer concerns any independent system with time-variable behavior that takes data from the user and produces output to drive audio/video synthesis. This modification can be a change of dimensionality as well as what is commonly considered "mapping"—namely, changes in numerical ranges, interpretation of "triggers" for events and mathematical analysis, and modification of the input, be they one-to-one, convergent, or divergent (Rovan et al. 1997). This modification, however, can be more complex if the mapping system is dynamic (i.e., changing over time). Notably, the internal behavior of the system can produce output variation without variation in the input. The system is visual, because we first choose mapping spaces that have clear graphical foundations. In the case of the two examples presented later in this article, mass-spring physical models and interpolations systems in perceptual spaces, both have clear visual interpretations that we believe are a significant strength of this approach. Consider a classic block diagram of the role of the mapping layer between the controller and the software instrument (see Figure 1).

By contrast, our dynamic visual mapping layer contains an additional block inserted between the controller and the instrument (see Figure 2).

We emphasize the autonomy of this layer by maintaining independent mapping blocks before and after our dynamic layer. We consider the pre- and post-mapping layers shown in Figure 2 as *classic mapping layers* (such as mathematical analysis and modifications for matching the outputs of a controller to the inputs of an audio/video synthesizer). On the other hand, the dynamic, independent mapping block in the center is a generative system whose inputs and outputs are not specific to any particular controller or audio/video synthesizer. Furthermore, whereas the mathematics inside a classic mapping layer have some knowledge of what numerical values arrive from a controller and what values are needed to drive our synthesizer, the independent layer has no knowledge of our controller or our instrument. It is rather a modeled dynamic system that has a particular behavior over time. Another way of describing this layer is a technique of controlling a virtual object that in turn controls the synthesis, as described in Mulder et al. (1997).

We offer two primary examples of dynamic, independent systems: mass-spring physical models and dynamic-interpolation spaces. In the sections to follow, we present visualizations and complete real-time instruments that use one or both of these techniques. However, the choice of systems is not limited to our two examples: we believe that other approaches like cellular automata or controlled stochastic systems may also be used for the same task.

First, we consider modeled mass-spring structures that are manipulated in real-time and behave according to Newtonian laws of forces applied to masses by springs. Our implementation is essentially based on the Cordis-Anima system (Cadoz et al. 1993; Cadoz et al. 2003). It allows modeling of virtual masses with visco-elastic links among them to create mass-spring networks. Like its predecessors, Cordis-Anima's GENESIS and MIMESIS environments, it offers a graphical environment for designing mass-spring models, discussed later in this article, and a real-time simulation engine that calculates the behavior of the system over time. Specialized externals were created for the Pure Data (Puckette 1996) and Max/MSP/Jitter (www.cycling74.com/products/maxmsp.html) environments that perform all the calculations for modeling an arbitrary physical model.

The choice of a standardized platform presents a significant advancement from previous implementations of physical-modeling simulation environments. We have found that Pure Data and Cycling 74's Max/MSP represent the most common software tools used by the community engaged in exploration, experimentation, and performance of computer-based creative work. The degree to which these software environments have been integrated into the field is evident in any cursory survey of academic programs that teach such topics, production houses that produce interactive/new media works, research centers that use and develop such software, as well as artists who create works and use these tools daily. Therefore, the availability of a generic physical-modeling environment in Max/MSP and Pure Data allows an appreciably larger group of explorers.

Furthermore, we emphasize our interest in using physical-modeling-based mapping layers to map any input to any form of audio/video synthesis. Because the inputs and outputs in these environments can simply be any other Max/MSP or Pure Data patch, other users can easily attempt to use our mapping layers for control of their audio or video synthesizers. We use a controller to affect the mass-spring model in a rather direct way (e.g., by moving around a mass), and we use the parameters that describe the model's state (the position, velocity, and forces of each mass) to generate the parameters for audio/video synthesis. When a force is applied to a mass-spring system, the structure changes, thus varying its behavior with time. With increasing complexity in the mass-spring models, the description of the structure's physical state becomes a high-dimensional vector that can be assigned to a rich audio/video synthesizer with many controllable parameters.

Our second proposed dynamic layer is an interpolation/extrapolation system that allows the performer to move about in a low-dimensional perceptual space (Momeni and Wessel 2003), where subjective similarity is inversely related to distance. This relationship can be based on a Euclidean measure of distance (or any other linear or non-linear function that one may wish to use, for that matter). The perceptual space is dynamically filled with parameters for the audio/video synthesizer. Snapshots of the audio/video synthesizer's state are placed in various locations in the perceptual space as high-dimensional parameter vectors. Low-dimensional output of a controller is then used to navigate the perceptual space and perform a weighted interpolation among all parameter vectors. Additional operation modes allow various types of extrapolation and gradual randomization of the parameter vectors. Furthermore, desirable interpolations that are discovered can be placed anywhere in the space as a new point in real time, thus promoting intuitive exploration of a high-dimensional space using low-dimensional controller data.

To contextualize this research and draw attention to our particular contribution to the literature, we make note of seven areas of research that inspire our work and lay the foundations of this field. At the same time, we try to provide a concise literature review of some rich and approachable resources for mapping and real-time computer instruments, especially for artists working with gesture and real-time software instruments.

Perceptual Spaces as Mechanisms for Control

Perceptual spaces represent an intuitive way to explore and master high-dimensional parameter spaces. The literature on this topic is quite rich and has been thoroughly reviewed in a previous work by the first author (Momeni and Wessel 2003; Momeni 2005) and referred to throughout the present article. This area of research continues to receive attention from researchers, as is evident in recent contributions by Bevilacqua et al. (2005) and Bencina (2005), as well as the integration of this research into some existing and upcoming commercial software packages, such as that of Place (www.electrotap.com/hipno). All of this research seeks techniques for controlling a large number of parameters with only a few input parameters and is thus closely related to the work we present here.

Physical Models for Sound Synthesis

The literature on computer-based musical instruments is rich with uses of physical models. Physical modeling is typically used in the field of audio synthesis. We differentiate the focus of our research from these applications: we attempt to use physical models to make mappings between gesture and any audio- and video-generation technique. That is, we are interested in physical models as control structures, not as audio-synthesis engines; this distinction will be clarified in great detail in the sections below. Nonetheless, audio synthesis through physical modeling relates to our research in two ways: first, in its use of some similar paradigms (namely mass-spring models), and second, in its desirability as a rich and multi-parametric synthesis technique that can be challenging to control in real time (Momeni and Wessel 2003). Although examples of audio synthesis through physical modeling are too abundant to allow a complete bibliography here, we note what we think are some outstanding contributions to this historically rich and well-established field. In particular, we find the Synthesis Toolkit (Cook and Scavone 1999), along with its corresponding port to the Max/MSP environment by Trueman and Dubois (http://music.columbia.edu/PeRColate) and its port to Pure Data by Matthes (www.akustische-kunst.org/puredata/percolate), to represent some of the most approachable avenues for exploration of physical models in sound synthesis today.

In the research/academic domain, IRCAM's Modalys software remains one of the richest environments for low-level design of physically modeled instruments. We must also mention here the tremendous amount of work done at the Association pour la Création et la Recherche sur les Outils d'Expression (ACROE) during the last 30 years. Their simulation environment Cordis Anima (Cadoz et al. 1984; Cadoz et al. 1993; Cadoz et al. 2003) is closely related to some of the work presented in this article. In the realm of commercial audio software, programs by Applied Acoustics Systems (www.applied-acoustics.com), including Tassman Sound Synthesis Studio and their latest program, String Studio, represent some of the best examples of sound synthesis through physical modeling for the consumer market (i.e., low cost, available as VST plug-ins, multiplatform). As for free software, the *dmiflute* (http://dmi.smartelectronix.com/flute.html) and *dmihammer* (http://dmi.smartelectronix.com/hammer.html) are quite convincing and playable instruments.

Gestural Control of Physical Models

Another area of research important to our work addresses the idea of controlling physical models in real time with user input. For a well-presented review, refer to Cook (2001). The literature in this area includes examples of physically modeled instruments that are performed in real time with a gestural controller: consider Trueman and Cook's BoSSA (1999), or the bowed violin model of Serafin et al. (1999). The aforementioned research at ACROE also addresses gestural interaction with simulations of physically modeled structures (Cadoz et al. 2003). In these works, user interactions generally affect the audio synthesis generated by the structures. Finally, we should note the technique known as *scanned synthesis* (Verplank et al. 2001) and its Max/MSP implementation provided by Courturier (2002). The technique involves "scanning the slowly varying shape of an object and converting the shape to samples of a sound wave" (Verplank et al. 2001). By allowing a performer to change the shape of this object with a gesture, scanned synthesis allows "direct dynamic control by the performer over the timbre of sounds" (Verplank et al. 2001). This technique is also related to our work in that it aims to allow gestural control of a synthesis process; however, the technique is quite specific to the generation of timbral variations as a result of waveshaping.

Concurrent Audio/Video Synthesis

Driving audio/video synthesis from a separate and unique data set provides the potential for a rich and perceivable link between the generated audio and video, a goal for any undertaking dedicated to sound-image correspondence. We also gain certain liberties in the realization of this link between audio and video. Because there is only one source of data for both, we are able to begin with complete synchrony and then reduce it to a desired level of abstraction. Noteworthy among previous works in this realm are the experiments of Kirk and Hunt (1998) on real-time control of audio/video synthesis. From their early experiments with the MidiGrid system (Hunt and Kirk 1994) to the later evolutions of their performance instruments embodied in the MIDAS system (Hunt et al. 1994; Kirk and Hunt 1998), their work concerns computer-based instruments with interactive graphical interfaces. These graphics were used initially as user interfaces (with common objects like knobs and sliders), but they were later extended to generate "dynamic graphical artifacts as a part of the performance output" (Kirk and Hunt 1998) controlled by the user's gestures.

Virtual Instruments as Mapping Layers

The word "mapping" is perhaps the most all-encompassing keyword in our research; our faith in the importance and potential for this stage of the work with computer based media, gesture, and performance cannot be overstated. Within this area, we draw attention to the body of work motivated by viewing the act of playing an instrument like the act of manipulating a virtual object, a concept well-described theoretically in Mulder et al. (1997) and put into good practice in Sergi Jordà's FMol (www.iua.upf.es/~sergi/FMOL), Nao Tokui's Sonasphere (www.sonasphere.com), Jordà et al.'s Reactable (www.iua.upf.es/mtg/reacTable), and AudioPad (http://web.media.mit.edu/~jpatten/audiopad/index.php), among others. We extend this idea by describing two generalized approaches to mapping that can be used to control any software audio/video instrument that can be viewed as virtual objects. The behavior of these objects—affected in real time by data from a controller—defines the behaviors of concurrent audio and video synthesizers.

Physical Models for Gestural-Control Feedback During Performance

We believe that objects—virtual or real—that react to user input in physically natural, perceivable, and informative ways present a good interface for mapping and real-time control. Some recent examples of work similarly concerned with the physical interaction with an object in the creative computer-based domain include instruments like that of Sinyor and Wanderley (2005)—a controller built to exploit the physical behavior of a simple dynamic system, namely a spinning wheel—and the "Pitch Stick" controller present on many of Clavia's Nord line of keyboard instruments (www.clavia.se)—a novel wooden, spring-loaded, pitch-bend controller). The most comprehensive survey and analysis of musical controllers to our knowledge is Jordà's doctoral dissertation submitted earlier this year (Jordà 2005), which includes many other examples of controllers with physically informative behavior.

Another important component of our proposed system is the role of the visual representation of the mapping layer as a form of feedback to the user/performer. In this regard, we note other works that propose to use physical models as mechanism for feedback during performance. The bulk of this work addresses haptic feedback using physical models (e.g., Cadoz et al. 1984; Luciani et al. 1994; Florens and Henry 2001; Howard et al. 2003).

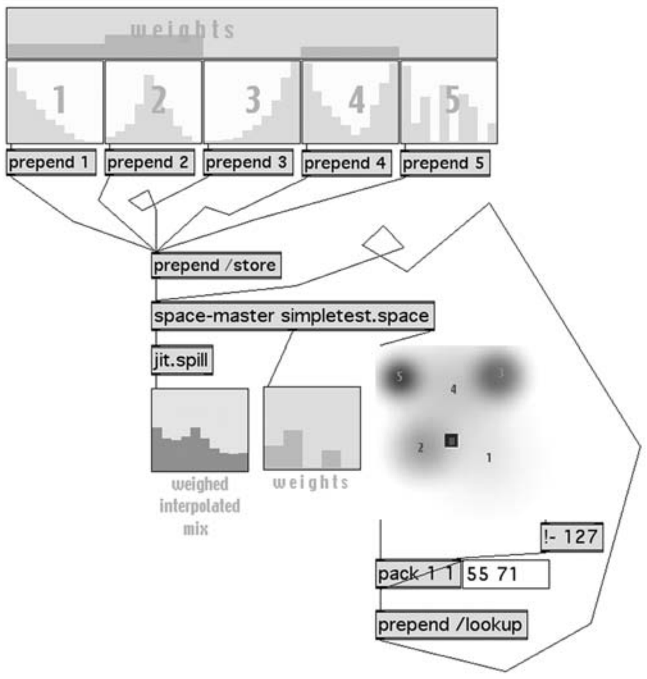


Figure 3. A Max/MSP patch that stores five lists into `space-master` and then uses a two-dimensional slider to interpolate among the lists.

Mass-Spring System for Real-Time, Audio-Visual Work

Finally, we focus specifically on previous systems that use mass-spring physical models for real-time audio-visual work. Most notable are the long-standing projects at ACROE, mentioned above and described in some detail in next section of this article. The research team at ACROE has published a tremendous number articles (750 references currently, according to the Web page at www-acroe.imag.fr/mediatheque/bibliotheque/bibliotheque.php), nearly all of which are in some way related to the use of physical models for sound and image creation. The research we present in this article owes much to this line of work at ACROE.

We note also the mass-spring simulation system with audio-visual applications presented in SodaPlay (www.sodaplay.com) first introduced in April 2000. SodaPlay provides an all-inclusive environment for creating two-dimensional mass-spring structures and simulating the behavior with user interaction (usually with a mouse). The system also includes a sound mapping for the structure's movement. An additional component created by Muth and Burton (2003) allows the user to extract parameters from the simulation and send them to another environment for sonification using the OpenSoundControl protocol (Wright and Freed 1997).

Benefits of a Dynamic Visual Mapping Layer

We discuss below what we consider significant features of dynamic, independent mapping layers that render them a rich approach to dealing with gesture mapping.

Performing Low-Dimensional to High-Dimensional Mappings

The primary goal of our proposed model of mapping is to allow exploration and expressive control of a high-dimensional parameter space using a low-dimensional gestural controller. This involves some form of dimensional scaling that allows one to transform n inputs to m outputs, where m is often significantly larger than n. Our mapping layers are thus algorithmic systems that take some input and produce a large number of output parameters.

We consider a simple example of such an n-to-m mapping for each of our two proposed mapping systems. In an interpolation space, the input to the system is simply the location in the perceptual space, typically a two- or three-dimensional vector. (Of course, higher-dimensional spaces are also possible, but there we lose the intuitive notion of exploration based on similarities and distances.) The output of the system is a weighed interpolation between all the parameter vectors in the system based on their distance from the present point. In Figure 3, we have placed 100-member parameter vectors (five lists of 100 floating-point numbers each), in five locations on a two-dimensional interpolation space. As we move about in the interpolation space, the five predefined vectors are weighed and mixed to produce a smooth interpolation of all 100 parameters. Here, we map a two-dimensional input vector to a 100-dimensional output vector.

Considering the case of mass-spring physical

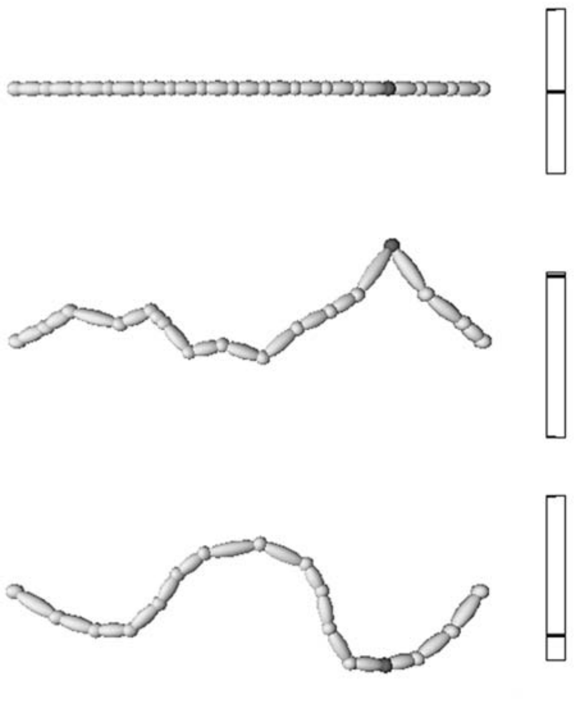


Figure 4. A mass-spring model of a string. Moving the single darker mass with a controller (whose value is shown in the slider to the right of each image) deforms the entire interconnected structure.

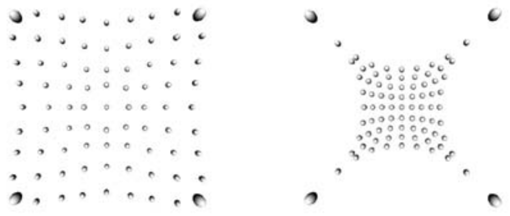


Figure 5. Two states of a mass-spring structure showing a change in the length and rigidity of the springs.

models, we use the simple example of a string, modeled by a series of 16 masses connected by springs in series (see Figure 4). We now control the vertical position of one of the masses directly with a controller. This results in the movement of not only this mass but all the other masses owing to the series of spring connections. In terms of our n-input-to-m-output system, we have here a mapping layer that accepts one input (the position of the mass we control) and produces 32 outputs (the two-dimensional position of all the linked masses). In our implementation we also calculate the velocity and sum of applied forces on each mass, as well as some high-level physical descriptors of the entire structure, thus generating about 100 outputs. These high-level descriptors are described later in this article the section "Mapping Techniques and Controllability."

High-Level Control

Because we attempt to control a complex independent system with low-dimensional input, it is imperative that the behavior of our algorithmic independent mapping layer be controllable at a high level. Consider the example of a mass-spring structure made of a two-dimensional grid of 81 masses arranged in a 9×9 matrix (see Figure 5). Adjacent masses in the x-y plane are linked with a spring, and the four corner masses are stationary, thus stretching the structure to form a kind of flexible surface. The most low-level manner of controlling the behavior of this system would naturally lie in controlling the positions of each of the 81 masses, thus giving complete control and deterministic results. A second manner, described in the previous section, would be to control one of the masses and allow its modeled physical interaction with the other parts of the structure to drive the systems' behavior. A third method—one that benefits from features particular to mass-spring models—is to control with four input parameters the resistance, length, and damping constants—one for each end of a spring—for all the springs in the system. This control, in conjunction with the direct manipulation of the position of one of the masses—or any other way of adding energy to the system—allows for control of dramatic changes in the structure's behavior. This changing behavior in turns produces a large number of varying output parameters that can control audio/video synthesis.

In the case of interpolation spaces, we consider a number of possibilities for high-level control. Most fundamentally, the high-dimensional output of the system is directly a translation of a low-dimensional coordinate in the interpolation space. We draw attention, however, to the effect of activating these interpolation spaces by dynamically displacing and changing the data in the space. By changing the data associated with one point in the space, one changes the interpolation result of the system everywhere in the space. Naturally, this effect depends on the level of influence that we have chosen for that particular point in the space. Furthermore, we also use extrapolation techniques that allow us to generate variations of a parameter vector (Momeni 2005). These variations are also produced by high-level control—a degree of extrapolation from what already exists in the space.

Time-Variable Behavior of Structure

We consider dynamic mapping layers—i.e., mapping algorithms whose behavior changes over time—to be a potentially rich approach to mapping for a number of reasons. The first goal of this approach to mapping is to allow exploration of a greater subspace of a rich audio/video instrument's complete parameter space. Imagine the simple case of driving an audio synthesizer with three parameters with two sliders as the controller. The parameter space of the instrument can be visualized as a three-dimensional Euclidean space where the *x-y-z* axes correspond to the three parameters for the instrument. The two sliders, on the other hand, allow the user to explore anywhere along a surface in the three-dimensional parameter space of the instrument. With a static mapping system, we are essentially limited to a subspace of the full parameter space, namely one two-dimensional manifold in the three-dimensional parameter space of our instrument. Now imagine that this two-dimensional manifold evolves in time owing to some internal algorithm that is intuitive and controllable. Our aim is to use such a dynamic system to intuitively explore a larger region of the complete parameter space.

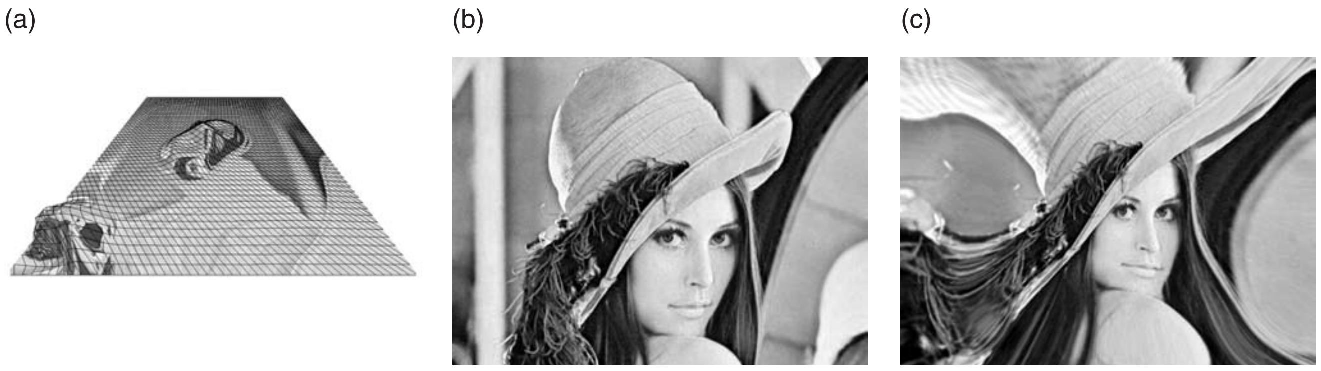
The second overarching goal is to integrate a sense of time in the mapping layer itself. A time-aware mapping layer has a number of interesting implications. First, a sense of continuation of a physical gesture is inherently present in the behavior of the mapping layer. An example is the gesture of throwing an object in the real world; the thrower's gesture during the act of throwing defines the trajectory that the object follows after it is released. The analogous example in the mass-spring modeling world shows similar behavior: the user puts energy into the system by affecting the structure in some way with a controller; once left alone, the structure's behavior is a natural continuation of its previously controlled behavior. Second, in the case of structure with no energy loss (i.e., no damping), we can create independent instruments with complex behavior, a common area of interest pursued notably by researchers interested in Artificial Intelligence, Artificial Life, and generative Hidden Markov Models. (See Visell 2004 for an impressive overview.) Lastly, the mapping layer offers an interesting environment for working with rhythm in the musical or visual domains.

Visual Mapping Layers

We are especially interested in dynamic mapping layers that have a graphical metaphor and thus a visual representation that can act as an interface for the mapping layer. It is important that this graphical representation of the mapping layer is a real indication of the internal algorithms of the mappings, as opposed to a subjective visualization of some time-varying data (such as found in the Apple iTunes visualizer). We believe that a visual paradigm and representation of the mapping layer eases exploration and expressive control of the system—especially if the mapping layer has a visual representation that is intuitively easy to understand and manipulate. Mass-spring models serve as a good example, because their behavior is reminiscent of many everyday objects.

An additional advantage of a visual mapping layer is its direct applicability to the video-synthesis portion of our instrument. Varying degrees of abstraction from the direct visualization of the mapping layer (e.g., masses and springs) can serve as a continuum of possibilities for video generation. In the example shown in Figure 6, we model a three-dimensional surface (see Figure 6a) with a matrix of interconnected masses and springs, and an image (see Figure 6b) is applied as a texture for this surface. After setting the structure into motion by applying a force somewhere, we map the *z*-axis displacement of each mass to a displacement of the image texture

Figure 6. Image transformation based on a mass-spring model of the deformation of a membrane. Shown here are (a) deformed membrane whose displacement from the z-axis was mapped to the repositioning of the corresponding pixel of the source image; (b) original photo (source image); (c) resultant deformed photograph.



for the corresponding pixel in the image used. The result is shown in Figure 6c. Because we are looking at the image from above, what we see is not an undulating surface, but rather a variably transformed version of our image.

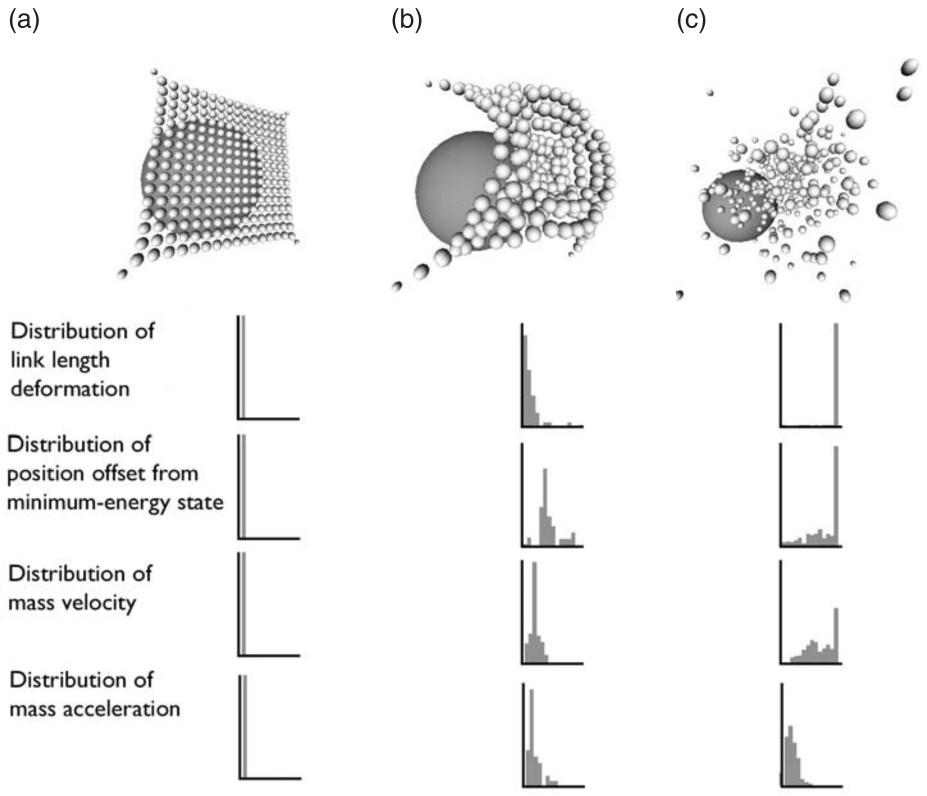
An Intrinsic Link Between Generated Audio and Video

Central in approach to expressive control of synthesis is the concurrent control of video and audio using a single independent mapping layer. We seek a rich relationship between generated audio and video. Work with real-time generation of video and audio often takes one of two approaches: audio-analysis for control and synthesis of video, or image/video analysis for control and synthesis of audio. Both approaches seek a connection between sound and image that is ideally rich, artful, and controllable. We propose an alternative that is concerned with sound and image generation from the same source of data—in our case a gestural controller. As Figure 2 shows, we view the audio and video synthesizers as two parts of the same instrument, and we attempt to control them concurrently with one independent mapping layer whose output is subsequently adapted to the inputs of the two branches. This approach allows an intrinsic connection between the audio and the video that is a direct result of their common sources of control. We then explore this connection by changing the classic mapping layers between our independent mapping layer and audio/video generators.

Mapping Techniques and Controllability

We now describe a number of techniques for mapping the output of our mapping layers to software audio/video synthesizers. In most cases, we describe each technique by way of a performance instrument. We also address important concerns intrinsic to work with complex dynamics systems, namely the state-dependent reaction of the systems to outside input and their overall controllability. There is a common theme to our approach in mapping the output of a dynamic system to sound and image generation: we attempt to use interpretations and reduction of the system's raw output as data to control sound and image. We propose a number of techniques for deriving high-level descriptors of the state of our mapping layer. As an example, we might consider the overall kinetic energy of a physical model as opposed to the positions of the masses. Because our ultimate goal with the dynamic, independent mapping layer is to generate data that varies in interesting ways and that can be controlled intuitively (rather than simulating real-life behaviors with physical models), we take great liberties in the ways we manipulate and affect the system. For example, we exploit "unnatural" stretching or hastening of a physical model's internal sense of time; we allow sudden leaps from one state to another as well as outside influences onto the system that do not conform to Newtonian mechanics; we explore negative values for the "rigidity" of a link between two masses, etc. In short, we propose an approach in which the mapping layer is essentially "unitless" and arbitrary; it does not represent a "real" physical

Figure 7. Movement analysis of a mass-spring model membrane: (a) minimum-energy state; (b) somewhat excited state; (c) nearly unstable state. The topmost images depict the model at three different times. For each time, statistical analysis of four physical parameters for all the components of the system is shown as four histograms. From top to bottom, the first histogram shows the distribution of the deformation of link-lengths (a value related to the potential energy); the second shows the distribution of the distances of masses from their equilibrium positions; the third shows the distribution of the velocities of masses; and the fourth histogram shows the distribution of accelerations of all masses. The means and variances of these distributions give a great deal of information about the state and behavior of the entire structure.



system. It is at the user's disposition to impose any influence desired, and the system will behave according to its defining algorithm.

We draw special attention to instruments that concurrently use mass-spring models and interpolation spaces. The interlinked complexity of such mapping layers emphasizes viewing the dynamic mapping layer of an independent system whose behavior we aim to control expressively. The mixtures of these models also facilitate more predictable and repeatable mappings during performance by allowing the performer to influence, limit, or dictate the behavior of the mapping layer in a multitude of ways.

High-Level Physical Descriptors of Mass-Spring Models

Our implementation of mass-spring models allows us access to low-level descriptors of the structure at a given time. For each mass, it provides the position, the velocity, and applied force in one to three dimensions; for each link it provides the length. We have also implemented a number of high-level descriptors that describe the state of the entire structure in fewer parameters than the details of each constituent part. Specifically, we calculate the mean and variance of four physical parameters for the entire model: the displacements of the masses from their equilibrium positions, the lengths of the links (related to the potential energy of the entire structure), the velocities of the masses (related to kinetic energy), and the accelerations of the masses (see Figure 7).

Considering a Part of the Structure with a View Window

As a way of exploring the behavior of the independent mapping layer, two analogous techniques were used in limiting what we take into consideration from our mapping layer's scope. In the case of the

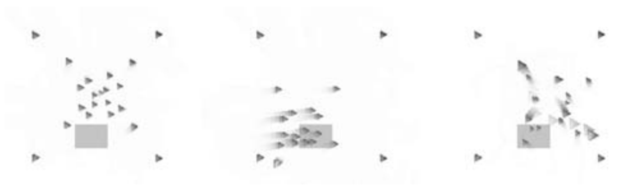


Figure 8. Three snapshots of a real-time, audio/video instrument. The small gray window allows the user to select the part of the structure that is considered for producing control information. By changing the size and position of this window, the user is able to select parts of the structure exhibiting different behaviors and to use the description of their behavior as control information for concurrent audio and video synthesis.

mass-spring models, an adjustable window is used to view only a part of the structure (see Figure 8). We then calculate the above-described low-level and high-level descriptors for only the selected part of the structure. Moving this window about is thus one low-dimensional manner of controlling the mapping layer.

Mixtures of Dynamic Mapping Layers

Before introducing the software implementation of this system, we discuss the interpolation space for physical-modeling parameters, interpolators used to mix predefined states of a physical model, the exploration of an interpolation space, and the use of physical models as modulators for the interpolator's inputs and outputs.

Interpolation Space for Physical-Modeling Parameters

An instrument was developed in which a complex mass-spring structure was manipulated in real time with a controller. This control is imposed in two ways: first, the position of the central mass in the structure is controlled directly by the controller; second, an interpolation space is used to generate parameters for all the physical constants for the model (i.e., spring constants, dampings, etc.). A Wacom drawing tablet (www.wacom.com) is used to concurrently control a pair of two-dimensional positions (the center mass in the mass-spring model and the location in interpolation space). The buttons on the Wacom tablet are used to add or move data in the interpolation space, as well as to impose a special case of non-damping anywhere in the structure (thus producing unending periodic movement in the structure). As a result, one is able to control quite intimately the behavior of the structure.

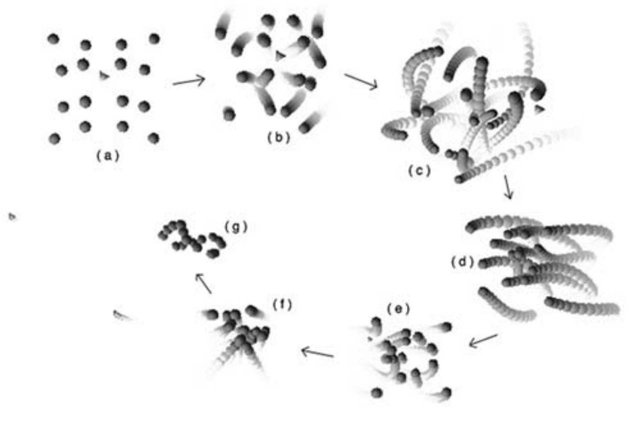
The use of interpolation spaces for the physical model's parameters is especially suited to instruments in which the visual appearance of the movement and behavior of the physically modeled structure provides a direct and intuitive link to understand and control its function as a mapping layer. Most mapping applications in which we consider the mapping layer as a "virtual object" to be manipulated by the user fit into this category. In these cases, we have found that the ability of the user to repeatedly produce a large variety of characteristic behaviors using the model is key to the success of the mapping scheme. Interpolations among predefined sets of physical parameters of the structure allow such control in a manner that is quite intuitive and variable. Returning to the question of controllability, we note that we have found it helpful to have among our predefined sets of physical parameters a few that produce quite predictable results; for instance, increasing the damping to high values will invariably stop the movement of the model, regardless of its previous state. Clearly, the ability to produce such predictable results enhances the functionality of the mapping layer.

Interpolators Used to Mix Predefined States of a Physical Model

An instrument was designed in which the behavior of the mapping layer is defined by a variable mixture of mass-spring models and an interpolator. Inspired by mixture-of-experts techniques, we model a mass-spring structure by calculating the position of each mass with our physical-modeling software; at the same time, an interpolation space is filled with parameter vectors that contain specific positions of all the masses in the structure (i.e., a particular arrangement of the modeled structure). As a result, the user can access the time-varying behavior of the complex mass-spring model, as well as known exact states of the system (see Figure 9). The interpolation space is filled in real time with snapshots of the structure by the user. It thus becomes a rich and varying space that can impose a regularity onto the physical-model's evolving output.

We find this type of mixture to be apt for mapping applications in which specific arrangements of a

Figure 9. Evolution of a mass-spring model in time and the use of two systems for calculating positions of masses. Images (a)–(d) show the mass-spring model in motion as a result of energy put into the system by movement of one of the masses. (The smaller triangle-shaped mass is controlled directly with a controller.) At the time of image (e), the controller is used in a different mode: to interpolate between the present location of all the masses and a number of predefined arrangements of the masses, as in image (g). Note that whereas the motion of the masses in images (b)–(e) is along curved trajectories, image (f) shows the transition between images (e) and (g) that involves each mass moving directly along a straight line to another position.



physical model's components serve as compositional elements. For example, imagine an instrument that translates the arrangement of masses in a three-dimensional structure to the harmonic content of synthesized audio. The composer for the instrument will might want to reproduce certain pre-composed harmonies during performance. In such a case, the interpolator will allow the performer to repeatedly find specific harmonies from the work's pallette, whereas the simulated behavior of the physical model will provide variation, rhythmic content, and extrapolations from the pre-composed harmonies.

Physical Model Output for Exploration of an Interpolation Space

An instrument was developed in which an interpolation space drives the input parameters for a number of software synthesizers based on physical modeling. Normally, a controller allows the performer to move about in the interpolation space, thereby continuously changing all the synthesis parameters. Here, a physical model with six linked masses was created; the positions of the masses were mapped to positions in the interpolation space. The six masses thus produced six interpolated synthesis parameters that were synthesized by a polyphonic synthesizer. The controller was then mapped to the position of one of the masses (and thus one of the voices of synthesis) as well as a second interpolation space controlling the physical parameters of the structure as described in the previous section. This creates a highly variable approach to polyphony that allows the exploration of various mixtures of homophonic and heterophonic textures. Depending on the physical parameters for the system, behavior similar to that observed in the flock-of-birds algorithm (Reynolds 1987) can be achieved. This provides an interesting approach to generating parameters for the various voices of some polyphonic system in which each voice is an audio or video synthesizer of some kind.

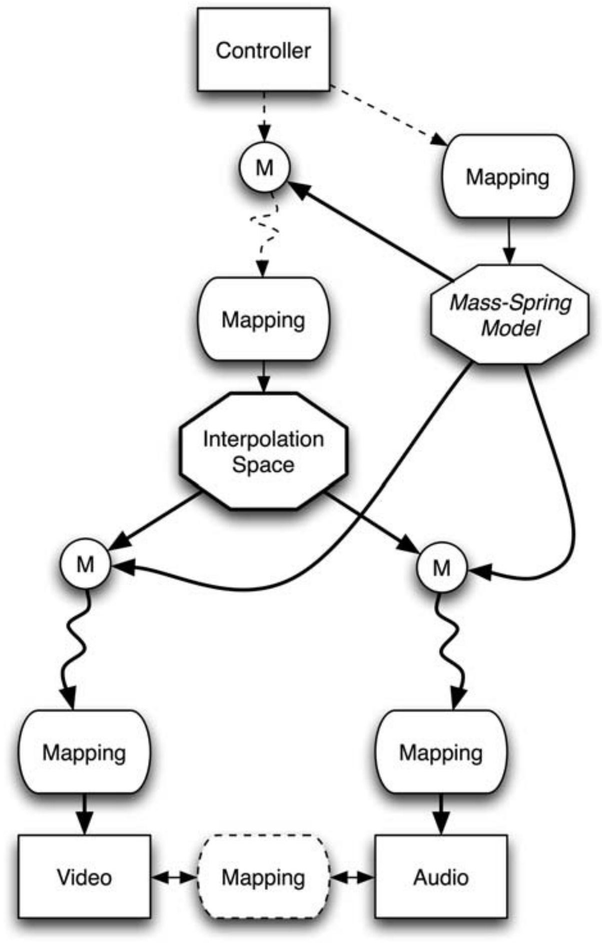
This type of mixture is also well-suited to explorative purposes in audiovisual instruments with very large parameter spaces, where a large number a interesting states have been composed and placed on an interpolation space. In such cases, the interpolation space is quite rich with potentially interesting mixes. Using a physical model as the input to the space allows one to explore continuous movements through many points in the interpolation space, as opposed to single static points. It also allows complex movements through the space with a sense of time inherent to the physical model's behavior. These features can greatly aid the process of finding these interesting mixes as well as potentially interesting transitions among them.

Physical Model as Modulators for the Interpolator's Inputs and Outputs

An instrument was developed in which frequencies and amplitudes of a "non-glissing" additive synthesizer (inspired by that of David Wessel in his 1980 work *Anthony*) were generated using an interpolation space. Moving about in this interpolation space provides a means for exploring a timbre space using a low-dimensional controller (Wessel 1979). Two separate mass-spring models were used to modulate the location-input and synthesis-parameter-output of the system. Owing to the rather large number of oscillators in the synthesizer (on the order of 1,000), variably modulating the frequencies allows control of a level of roughness in the timbre. The behavior of the physical model is thus a representation of the timbral variations in the synthesis.

This final type of mixture points to a more gen-

Figure 10. Block-diagram of an instrument exploiting interaction between a mass-spring model and an interpolation space. The controller's output is used in two ways: first to move about in an interpolation space, and second, to control a mass-spring model. The small circular blocks labeled "M" are modulation steps; these blocks show the integration of the mass-spring model as modulator applied to the input to the interpolation space, as well as its high-dimensional output.



eral use of the outputs of a physical model as modulations for control signal. Modulations of control signals often tend to be periodic functions, centered around a desired value and with a variable modulation depth (e.g., a low-frequency oscillator, or LFO). Both the periodicity and the variability of depth are features easily produced by simulating a mass-spring system in motion. In the example shown in Figure 10, a physical model's output was used to modulate the inputs and outputs of interpolation spaces. However, one can very well imagine attaching a physically modeled behavior to any control signal in a performance instrument in hopes that interesting modulations of that signal will produce compelling results.

chdh

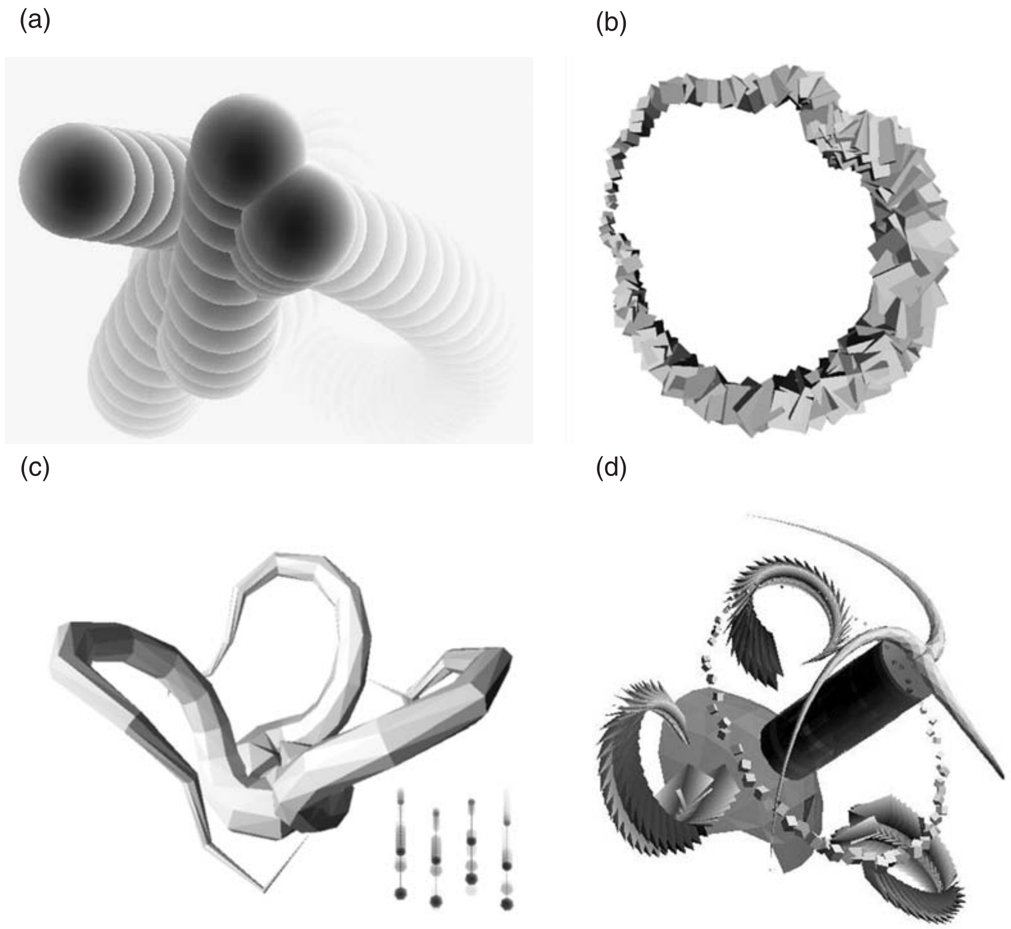
The chdh software environment (www.chdh.net) for live performance is primarily based on the use of physical models for audio and video synthesis. The environment allows the generation of sound and image from about 30 modeled structures at a time, each of which is an independent algorithmic system for controlling audio and video. The software allows one to sequence state changes for each of these structures. These changes of state in turn produce changing audio and video. In addition to the sequencer, the parameters for each structure can also be controlled in real time using faders and two-dimensional controllers mapped to interpolation spaces. Figure 11 shows some still-image extractions of the generated video from chdh.

The example shown in Figure 11a is composed of only three masses, each represented with spheres. The first mass is linked to the other masses and to the center of the screen. The user can apply a force to the first mass and change the rigidity and damping of all links independently. The corresponding audio synthesis uses three white-noise generators sent through three low-pass filters. The cutoff frequency and resonance of each filter is derived from the velocities of each mass and the force applied to it.

Figure 11b shows a physical model of a "circle" in a two-dimensional space, where each mass (represented by a cube with different size) is connected via a visco-elastic link to every other mass and to the center of the structure. The model is used to control a granular synthesizer: each mass represents a single grain. The angular position in the circle of the biggest cube controls the start position of the grains in the audio sample being granularized. The force applied to each mass controls the pitch transposition of its corresponding grain. The user can "play" the structure by moving the center or by changing some physical parameter like rigidity and damping of the links. The global movement can be controlled at a high level, providing easy and intuitive control of the sound and of the sound evolution, in addition to the relationship between the sound and the video synthesis.

In the example shown in Figure 11c, a three-dimensional model of three tentacles is animated

Figure 11. (a)–(d) Still images from a video produced by the chdh real-time performance environment.



by the output of a physical model. The physical model is composed of four independent systems of three masses each, connected in series with two springs (shown in the smaller image in the bottom right). The position of the topmost mass in each of these three-mass systems is modulated with noise, and the position of the bottom masses are observed. The user controls the behavior of the physical model by directly changing the rigidity and damping of the springs. The physical model's output (the observed positions of the third mass in each of the four independent physical models) is mapped to the *x* and *y* angles of flexion at the base and tip of each tentacle. As a result, the movement of the masses in the physical model is used to control movements of the three-dimensional object. The physical model can also drive the audio synthesis by changing the amplitudes of four corresponding oscillators with the positions of the four masses. Finally, Figure 11d shows a snapshot of a chdh performance that mixes multiple different instruments at the same time.

Implementation

We remain committed to implementations of our mapping systems for the Max/MSP and Pure Data environments for two reasons: first, their large following in the community, and second, their ability to allow users to map anything to anything else us-

Figure 12. A physical model of a string implemented with Pure Data. Cross-platform externals developed for Max/MSP and Pure Data allows the user to define an arbitrary mass-spring structure in one, two, or three dimensions by connecting mass and link objects and defining initial conditions.

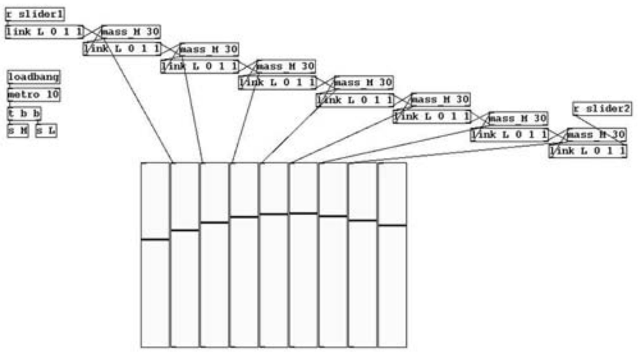
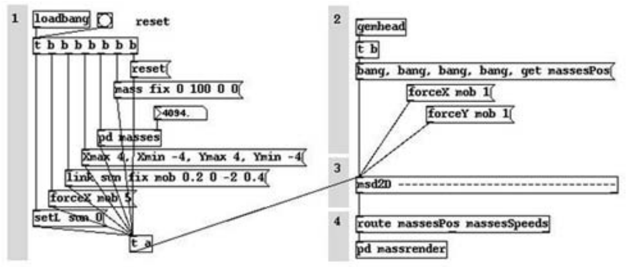


Figure 13. An example of a physical modeling simulation using `msd` *within Pure Data. The system simulates several thousands masses (4,094 to be exact) responding to an attraction to the larger central mass as well as various outside forces sent through additional messages to the* `msd` *object. (a) Structure of the Pure Data patch that performs the simulation. The first part of the patch is used to create the mass-*

(a)



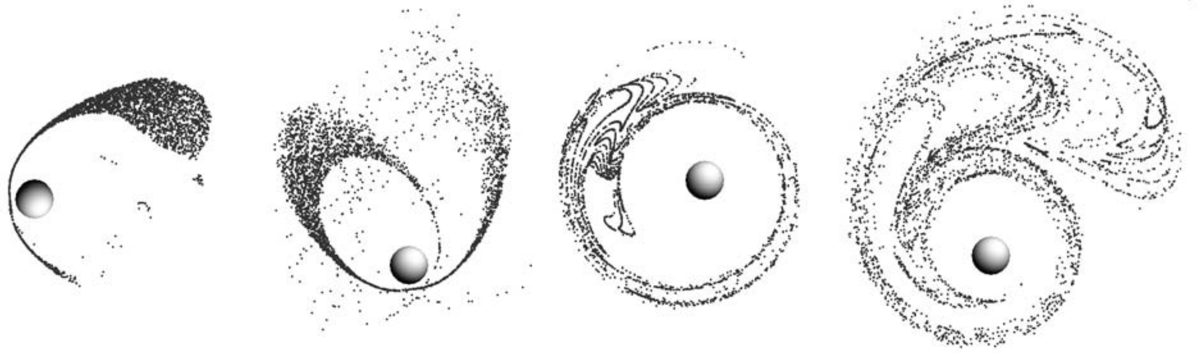
ing our system (most importantly, for mapping their controllers to their existing audio and video synthesizers). The interpolation system was initially implemented as a Max/MSP/Jitter patch (Momeni and Wessel 2003). This implementation has evolved to significantly improved versions now implemented as cross-platform JavaScript code that uses the OpenGL architecture for visualization. The new version allows a much more user-friendly mechanism for designing spaces with the mouse. It also links the entire space-interpolation world to the Max/MSP parameter store-recall system introduced with the `pattr` family of objects. The JavaScript code runs within Max/MSP's `js` and `jsui` objects. These objects are freely available as a part of `_aLib` (available online at http://cnmat.cnmat.berkeley.edu/~ali/share/max/_aLib.zip). Within Pure Data, similar interpolation capabilities were added to the `pbank` object by the second author.

The physical-modeling simulation environment currently exists in two implementations. The first, with which the examples in this article were created, is the set of Pure Data externals named `pmpd` (physical modeling for Pure Data). This implementation was created by the second author in 2004 and is available online at http://drpichon.free.fr/pmpd. These objects allow the modeling of virtual masses and visco-elsatic links among them to create mass-spring networks similar to those described by Cadoz et al. (1993). Basic objects that represent a mass, a link, or an interaction can be connected to compose a structure (see Figure 12). In this way, `pmpd` can be used to simulate a variety of systems, including dynamic physical systems, fluid motion, as well as human face animation (Lee et al. 1995). The `pmpd` externals were ported to Max/MSP by the first author in 2004, and they are available online at www.cnmat.berkeley.edu/~ali/share/max/pmpd.

The second implementation of our system is the collection of externals named `msd` (mass spring dashpot) written by Montgermont (2005) for Pure Data and Max/MSP. The goal of `msd` is very similar to that of `pmpd`, but its structure is more optimized: only a single object is needed to compute any simulation. This provides the capacity to simulate, for instance, a system composed of more than 10,000 masses and links at 20 Hz on an ordinary laptop, and many thousands more on faster machines, as illustrated in Figure 13. The user sends messages to the `msd` object to create or destroy virtual masses or links inside the `msd` objects. Users can also send messages to interact with the simulation or request the simulation's parameters, such as the position and velocity of masses. A graphical editor that facilitates the design of structures has also been implemented in the Pure Data programming environment by Frank Barknecht (http://footils.org/cms). `msd` is composed of three externals for simulation in one-, two-, and three-dimensional spaces. (The externals are simply named `msd`, `msd2D`, and `msd3D`). These externals were written using the flext C++ programming layer for creating cross-platform Max/MSP and Pure Data externals, written by Thomas Grill and available online at http://grrrr.org/ext/flext. The `msd` source code is freely available from the Pure Data developer CVS repository at www.sourceforge.net, available for use under the GNU Public License.

spring network by defining initial positions, arrangement of the connections and boundary limits. The second part allows users to interact with the simulation by imposing forces on the masses. The third part of this patch is the `msd2D` *object that makes all the calculations for the model. In the fourth part, information about the model is sent to another Pure Data patch for mappings to audio and video generation. (b) Four different states the system, visualized with GEM and openGL. At the initialization of the simulation, the different mass are released into their trajectories and slightly different times from one another, thus making their consequent reactions to other forces unique owing to their unique state.*

(b)



Conclusion

We described a technique for mapping the low-dimensional output of a gestural controller, to the high-dimensional input of real-time audio/video synthesizer. Our approach treats this mapping layer as an independent system with behavior that changes over time. We also described our interest in generating audio and video in real time using a single, complex mapping layer. We believe that this approach allows a rich and intuitive exploration and performance environment for work with audio and video generation. Furthermore, we believe that the use of a dynamic system for concurrent real-time control of audio and video creates an intrinsic link between the two produced media that is rich with potential.

Acknowledgments

We would like to thank David Wessel, Adrian Freed, and John MacCallum of CNMAT for their help in developing these ideas and reading this article before submission.

References

Arfib, D., et al. 2002. "Mapping Strategies Between Gesture Control Parameters and Synthesis Models Parameters Using Perceptual Spaces." *Organised Sound* 7(2):135–152.

Bencina, R. 2005. "The Metasurface: Applying Natural-Neighbour Interpolation to Two-to-Many Mappings." *Proceedings of the 2005 Conference on New Interfaces for Musical Expression.* Vancouver, Canada: University of British Columbia, pp. 97–100.

Bevilacqua, F., et al. 2005. "MnM: A Max/MSP Mapping Toolbox." *Proceedings of the 2005 Conference on New Interfaces for Musical Expression.* Vancouver, Canada: University of British Columbia, pp. 85–88.

Cadoz, C., A. Luciani, and J.-L. Florens. 1993. "CORDIS-ANIMA: A Modeling and Simulation System for Sound and Image Synthesis-The General Formalism." *Computer Music Journal* 17(1):19–29.

Cadoz, C., et al. 1984. "Responsive Input Devices and Sound Synthesis by Simulation of Instrumental Mechanisms: The Cordis System." *Computer Music Journal* 8(3):60–73.

Cadoz, C., et al. 2003. "ACROE-ICA: Artistic Creation and Computer Interactive Multisensory Simulation Force Feedback Gesture Transducers." *Proceedings of the 2003 Conference on New Interfaces for Musical Expression.* Montreal, Canada: McGill University, pp. 235–246.

Cook, P. 2001. "Principles for Designing Computer Music Controllers." Paper presented at the ACM CHI Workshop on New Instruments for Musical Expression, Seattle, Washington, 1 April.

Cook, P. R., and G. P. Scavone. 1999. "The Synthesis ToolKit (STK)." *Proceedings of the 1999 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 164–166.

Courturier, J. M. 2002. "A Scanned Synthesis Virtual Instrument." *New Instruments for Musical Expression.*

Proceedings of the 2002 Conference on New Interfaces for Musical Expression. Limerick, Ireland: Department of Computer Science and Information Systems.

Domino, M. 2004. "The Silk Road Project." Available online at www.fieldform.com/event.php?id=8.

Florens, J.-L., and C. Henry. 2001. "Bowed String Synthesis with Force Feedback Gesture Interaction." *Proceedings of the 2001 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 37–40.

Howard, D. M., et al. 2003. "Force Feedback Gesture Controlled Physical Modeling Synthesis." *Proceedings of the 2005 Conference on New Interfaces for Musical Expression.* Montreal, Canada: McGill University, pp. 95–98.

Hunt, A. 2005. "The Use of Sound, Graphics, and Tactile Feedback to Improve the Naturalness of Human-Computer Interaction." Available online at www-users.york.ac.uk/~adh2/Research/research_intro/intro_files/frame.htm#slide0003.htm.

Hunt, A., and R. Kirk. 1994. "MidiGrid: A Computer-Based Musical Instrument." *Journal of the Institute of Musical Instruments Technology* 1:3–13.

Hunt, A., et al. 2002. "The Importance of Parameter Mapping in Electronic Instruments Design." *Proceedings of the 2002 Conference on New Interfaces for Musical Expression.* Limerick, Ireland: Department of Computer Science and Information Systems.

Hunt, A. D., et al. 1994. "Evolution of Timbres Through the Use of Tabula Vigilans on the MIDAS system." *Contemporary Music Review* 10(2):201–210.

Jordà, S. 2005. "Digital Luthery: Crafting Musical Computers for New Musics, Performance, and Improvisation." Ph.D. Dissertation, Universidad Pompeu Fabra.

Kirk, R., and A. Hunt. 1998. "Computer-Assisted Performance Control for Audio-Visual Instruments." In R. Kopiez, ed. *Schriften Musikpsychologie und Musikasthetik,* Volume 12. Frankfurt, Germany: Peter Lang, pp. 149–158.

Lee, Y., et al. 1995. "Realistic Modeling for Facial Animation." *Proceedings of the ACM SIGGRAPH.* New York: Association for Computing Machinery, pp. 55–62.

Lesbros, V. 1996. "From Images to Sounds: A Dual Representation." *Computer Music Journal* 20(3):59–69.

Luciani, A., et al. 1994. "The CRM device: A Force-Feedback Gestural Transducer to Real-Time Computer Animation." *Displays* 15(3):149–155.

Momeni, A. 2005. "Composing Instruments: Inventing and Performing with Generative Computer-Based Instruments." Ph.D. Dissertation, University of California at Berkeley.

Momeni, A., and D. Wessel. 2003. "Characterizing and Controlling Musical Material Intuitively with Geometric Models." *Proceedings of the 2003 Conference on New Interfaces for Musical Expression.* Montreal, Canada: McGill University, pp. 54–62.

Montgermont, N. 2005. "Modèles Physique Particulaire en Environement Temps-Réel: Application au Contrôle des Paramètres de Synthèse." D.E.A., Université Pierre et Marie Curie.

Mulder, A., et al. 1997. "Mapping Virtual Object Manipulation to Sound Variation." *IPSJ Sig Notes* 97(122):63–68.

Muth, D., and E. Burton. 2003. "Sodaconductor." *Proceedings of the 2003 Conference on New Interfaces for Musical Expression.* Montreal, Canada: McGill University, pp. 222–224.

Puckette, M. S. 1996. "Pure Data." *Proceedings of the 1996 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 269–272.

Reynolds, C. W. 1987. "Flocks, Herds and Schools: A Distributed Behavioral Model in Computer Graphics." *Proceedings of the ACM SIGGRAPH.* New York: Association for Computing Machinery, pp. 25–34.

Rovan, J. B., et al. 1997. "Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance." *Proceedings of the AIMI International Workshop: KANSEI—The Technology of Emotion.* Genoa, Italy: Associazione di Informatica Musicale Italiana.

Serafin, S., et al. 1999. "Gestural Control of a Real-Time Physical Model of a Bowed String Instrument." *Proceedings of the 1999 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 375–378.

Sinyor, E., and M. M. Wanderley. 2005. "Gyrotyre: A Dynamic Hand-Held Computer-Music Controller Based on a Spinning Wheel." *Proceedings of the 2005 Conference on New Interfaces for Musical Expression.* Vancouver, Canada: University of British Columbia, pp. 42–45.

Trueman, D., and P. Cook. 1999. "BoSSA: The Deconstructed Violin Reconstructed." *Proceedings of the 1999 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 232–239.

Verplank, B., et al. 2001. "Scanned Synthesis." *Journal of the Acoustical Society of America* 109(5):2400.

Visell, Y. 2004. "Spontaneous Organisation, Pattern Models, and Music." *Organized Sound* 9(2):151–165.

Wessel, D. 1980. *Anthony.* Mainz, Germany: Wergo WER 2030–2032.

Wessel, D., and M. Wright. 2001. "Problems and Prospects for Intimate Musical Control of Computers." Paper presented at the ACM CHI Workshop on New Instruments for Musical Expression, Seattle, Washington, 1 April.

Wessel, D. L. 1979. "Timbre Space As a Musical Control Structure." *Computer Music Journal* 3(2):45–52.

Winkler, T. 1995. "Making Motion Musical: Gesture Mapping Strategies for Interactive Computer Music." *Proceedings of the 1995 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 261–264.

Wright, M., and A. Freed. 1997. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers." *Proceedings of the 1997 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 101–104.