About This Issue

Constraint solvers are software systems that find solutions to constraint-satisfaction problems (CSPs). Many familiar rules of musical composition, such as those of species counterpoint or twelve-tone serialism, can be expressed as CSPs, but so can new rules devised by a composer for a new piece. A musical constraint solver finds solutions to problems in which certain elements of a musical score are left unknown; i.e., they are variables. The set of all values that may be assigned to a variable is known as its domain, and the rules governing allowable assignments in specific musical contexts are called constraints.

This issue contains articles about two musical constraint solvers. In the first case, the software, called PatchWork Musical Constraints (PWMC), is written atop the visual programming environment PWGL, which has been described in previous issues of this journal. By virtue of this environment, PWMC allows the user to specify constraints through Max-like graphical patches and to produce music notation as the output. In the second case, a software environment called Strasheela written in the multiparadigm programming language Oz, the user interface is textual. Strasheela allows the user to define not just constraints but also constraint applicators: higher-order functions that can traverse a score in arbitrarily complex ways, mapping arbitrary constraints to selected items in the score.

The next two articles concern new interfaces for controlling the production of sound. The two have a common goal of providing intuitive and easily manipulated sonic emulation of instruments that in their original form are not particularly easy to learn to play. In the case of the first of these articles, the emulated instrument is electroacoustic: the turntable. Through extensive practice, turntablists develop gestural nimbleness in manipulating vinyl discs and mixer controls. The authors of this article (Kjetil Hansen and Roberto Bresin) address the question of how software can emulate a turntablist's sounds while providing a control layer that insulates the user from the physical device's learning curve (and also from its fragility and weight, especially considering the many discs in a typical DJ setup). The abstraction of this layer permits non-real-time control, facilitating composition with such sounds. Additionally, this layer allows the sound production to be controlled in real time by other devices. In the latter regard, the article summarizes musicians' experimentation with the Reactable and the Radio Baton as controllers for virtual "scratching." By providing control at a higher level than a simple one-to-one mapping of user gesture to emulated turntablist gesture, the software offers new techniques that exceed the physical instrument's limitations. Thus it not only makes scratching accessible to novices but also potentially provides new sonic resources for professional DJs.

In the case of the second of these articles, by Steven Gelineck and Stefania Serafin, the emulated instruments are acoustic musical instruments, and, by extension, other physical sound generators. Here the question is how to control physical models for sound synthesis in ways that both encourage creative exploration of the models' sonic potential and also provide some physicality for a more intuitive "feel" than software alone can provide. The authors present a device called the PHYSMISM that incorporates four different physical controllers, including musically atypical ones such as a crank. The controllers, which are operated by blowing, rubbing, grinding, or hitting, manipulate four corresponding physical models: turbulence, stochastic, friction, and impact models. For greater diversity, the models can be interconnected. The authors summarize user feedback, and they also propose seven directives to be followed when designing systems that, like the PHYSMISM, are intended for exploration of sound synthesis by physical models.

The final article, by researchers Chrisoula Alexandraki and Demosthenes Akoumanakis in Crete, presents software for networked musical performance. The authors aim to provide a customizable framework that is useful for many different scenarios involving geographically distributed musicians: not only live concert broadcasts, but also rehearsals, music lessons, master classes, jam sessions, and collaborative composition sessions. The software permits a variety of network topologies, including client–server (a star pattern), peer-to-peer, and combinations of these. The system architecture contains a server for streaming audio and video, a collaboration server,

Front cover. An illustration from the article by composer Örjan Sandred, showing his PWMC software for musical constraint solving.

Back cover. Two screen images from the article by Chrisoula Alexandraki and Demosthenes Akoumanakis on DIAMOUSES, their framework for networked music performance.

components for each distributed musician, and a digital TV broadcast center. The article presents results of experiments with musicians in different scenarios and musical genres, pointing out the diversity of requirements that musicians have for networked performance. These experiments focused on technical quality-of-service measurements and user feedback, but the authors also stress the importance of studying the different collaboration practices that their software facilitates.