ABSTRACT: Sound synthesis using physical models provides many expressive possibilities, with an associated cost that all of the parameters must be controlled, and sometimes parameters interact in ways which are not obvious. The way that a human player solves many of these problems is by listening to (and feeling) their instrument and constantly updating the parameters. Here a software architecture and system is described which contains various objects encapsulating the instrument physics, and the physics, expertise, and perception of the player. Inter-object connections to modify instrument control parameters "on the fly" yield more consistent performances.

1. Introduction

Control parameter values for physical modeling synthesis are difficult to obtain by linear or non-linear system identification techniques because of the rapidly varying nature of transients. Further, since many interesting physical models contain a non-linear oscillator, parameters which work well once may not work the next time the same performance is attempted. The way that a human player solves these problems is by setting up instrument control parameters from memory, then adapting the control parameters based on auditory feedback from the instrument. Synthesis by physical modeling can also benefit from a feedback control system such as this, and can further benefit from a more complete simulation of the physics and expertise of the player. A software architecture, evolved from the experience of creating several physical models and controllers (Cook92,92b), will be described which contains various objects encapsulating the instrument physics, player physics, player expertise, and simple aspects of player perception.

2. Objects In a Complete System

Figure 1 shows a general Performer/Instrument class hierarchy. Blocks will be discussed with examples.

A. Instrument Family Acoustics: The physical acoustics characteristics of a particular instrument family, such as the brass family, bowed string family, etc. All Acoustics of Specific Instruments within a family, containing all the relevant tone-producing musical acoustics. These are waveguide synthesis models (Smith87), such as Clarino, Hoisell, Flutea, and DSPGinger objects used in the Clarino, Hoisell, SlideFlute, and SPGinger programs.

B. Instrument Performance Physics Model: Models the physics, other than the acoustics modeled in A, of the instrument. Examples include the mass and damping of the trombone slide or trumpet valve (dynamically modeled), or the length of the guitar fretboard (lookup table) to be used in calculating how long it should take a human arm to slide from one position to the other. The bandwidth required for truly coupling the instrument physics to the performer physics could be quite high, although it is unlikely that this level of simulation must take place at audio rates. All These model a specific member of the family.

C. Performer Physics Model: Contains physical limitations of the player, such as the mass/spring/ damping of the arm and hand (Jaquity et al. 94). Also contains rules for perceptual/physiological guidance of articulators to targets, using instrument physical descriptions acquired by querying B. For example, the jaw can drop only so quickly, and the tongue body moves slower than the tongue tip. The Singer program now includes mass and damping on each articulator (Cook93), so that each articulator drives to the target point according to an individual time constant. Another case might involve modeling the time required for a flute player's finger to move, possibly modeling the interaction of fingers within the hand. All If desired, the specific instrument case could be addressed, noting that a skilled player develops muscles and strategies
unique to their specific instrument.

D. Expert Instrument Family: Contains an expert's knowledge for controlling instruments of the same family. Examples of such families include bowed strings of the same tuning characteristics, three-valved brass instruments, etc. Such knowledge might include: coarse settings for lip position and valve position to form particular notes on brass instruments. The object MIDIController within the HostPlayer program contains a lookup table of slide and lip parameters which give a coarse setting for each MIDI note number.

E. Expert Specific Instrument: Detailed settings for the specific member within the instrument family, such as the lip and breath offsets which differentiate a tuba from a trumpet. Dij: The fine details that cause a good player to be comfortably proficient on a very specific instrument. This part should probably be adaptive subject to feedback (Szilas et al. 93)(Wessel91), so that the player object can quickly adjust to a new instrument. Some functionality could be implemented by neural networks (Lee et al. 92).

Perceptual Model: "Listens" to output of physical model and drives control parameters to achieve the desired output. This object should have the capability of a 'musically astute' listener, but not necessarily any knowledge of the specific instrument. The Expert Objects (D) could be sent messages specifying that the sound is flat, shrill, or too loud. The necessary changes to the control parameters would be processed with input from the Expert Objects, then appropriate messages would be sent to the physical model. The Perceptual Model could be composed of simple time domain pitch detector coupled to a 'classical expert system, or a full blown model of the ear and brain (when such exists).

3. Implementation of a Simple Player/Instrument System

Figure 2 shows the software and connection architecture of a simple system including many of the elements described above. The PlayerController block does all decision making and processing, and various other blocks respond to queries from the PlayerController. The Expert is a simple lookup table, the Perceptual Model is a simple pitch and level detection program (Cook et al. 92). The line from Player to Instrument is accomplished via a MIDI connection, over which simple MIDI Control Change messages are passed. The Instrument consists of a DSP synthesis instrument, controlled by an InstrumentController, whose purpose is to convert MIDI messages into parameter changes for the synthesis instrument. Figure 3 shows the user interface panels for the programs implementing these Performer and Instrument functions.

4. References


(Cook et al. 93) P. Cook, D. Morrill, and J. Smith, "An A MIDI Control and Performance System for Brass Instruments," ICICM, Tokyo.


(Janovsky et al. 94) J. Janovsky, M. Karjalainen, and V. Vainio, "Intelligent Synthesis Control with Applications to a Physical Model of the Acoustic Guitars," ICICM, Athens.


