CHAPTER 10:
PERFORMANCE REALIZATION OF BRUCE PENNYCOOK'S *PRAESCIO IV*

My current concert realization of *Praescio IV* is based on Pennycook's latest version of the performance software, with a few of my own modifications and additions. Pennycook's system interprets MIDI input and, in response, executes live MIDI processing routines according to a set of event list files. Pennycook's software does not include pitch-tracking or sound generation facilities, which, along with live system control via triggers and pedals, were originally implemented using external devices. These elements must be simulated in software or replaced with updated equipment. The following is a description of a complete concert realization of *Praescio IV*, including the stage setup, the equipment used, the essential components of the system software, and the synthesizer orchestration.

10.1 EQUIPMENT & STAGE SETUP

The sound system and stage setup for *Praescio IV* are relatively simple. The interactive system software currently requires an Apple Macintosh G3 computer running MacOS 9.2. Besides the computer and a clarinet, *Praescio IV* requires external control devices (foot pedals and switches), a microphone (for pitch tracking), computer interfaces for MIDI and audio, a synthesizer (which may be either an external device or integrated into the system software), and a sound reinforcement system (amplifier and loudspeakers).

10.1.1 Input Devices

Live input to the interactive MIDI system is provided in two ways: a) through direct MIDI control via footswitches and pedals, and b) through detection of clarinet pitch and amplitude, translated into MIDI note messages. Two footswitches (trigger and sustain) and a foot pedal (volume) provide direct MIDI control via the computer's MIDI interface (MOTU Fastlane USB). A contact microphone (mounted on the clarinet mouthpiece) is connected to the computer's audio interface (M-Audio FireWire 410), providing an input signal for the pitch tracking software.

Foot Controls. I use an integrated foot-controller device (DigiTech RP-20 floor-mounted effects processor/MIDI foot controller) to send MIDI signals to the system software. The RP-20 provides twelve programmable footswitches and a single volume pedal. As with the setup for *Narcissus*, I use a MIDI merge device to add a piano-style sustain pedal. For *Praescio IV*, the RP-20 "Bank Down" footswitch (closest to the volume pedal) is programmed to transmit MIDI controller 65 (event trigger), the volume pedal transmits controller 7 (volume), and the piano-style pedal transmits controller 64 (sustain).

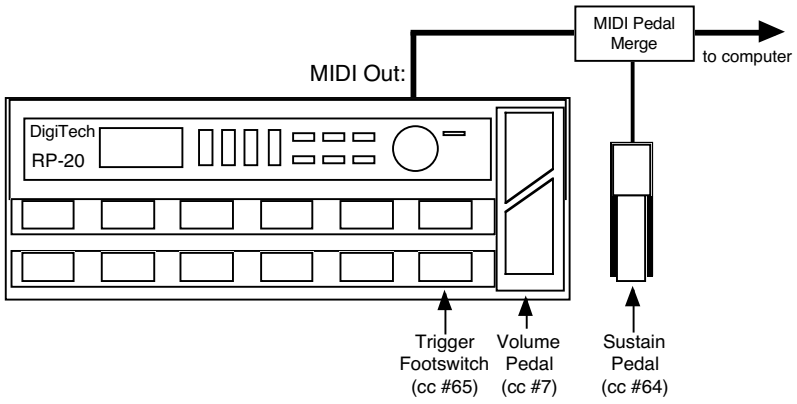

Figure 10.1. DigiTech RP-20 effects processor and MIDI foot controller

Pitch Tracker. Previous realizations of this work used an external pitch-tracking device (IVL PitchRider) that sent MIDI note data to the computer. In order to avoid altogether the need for external pitch tracking hardware, I have embedded pitch-tracking functions into the performance software.

My software implementation of the pitch tracker is a simulation of the IVL PitchRider. It analyzes incoming audio for pitch content and amplitude (using the fft-based *fiddle~* object in Max/MSP) and generates MIDI "note on" and "note off" messages that follow the notes played by the clarinetist. Note data from the pitch tracker is used to trigger selected score events and controls the "coloration" (doubling or parallel harmonies generated by the sound module) of the clarinet in designated "THRU" events.

In order to generate discrete MIDI note events with clear start ("note on") and stop ("note off") points, input is passed through compressor/limiter before going to the pitch tracker. Sound below a given amplitude threshold is not tracked. Once a sound is detected above the threshold, the pitch is analyzed to generate a note number and amplitude is measured to generate a corresponding note velocity (attack volume) value.

Once the input sound drops below the threshold or the pitch changes, a corresponding "note off" message is generated.

The pitch-tracking module shown in figure 10.2 is embedded as a Max/MSP "sub-patcher" in the system software. Pitch data, used to control event progression and coloration, is made available to the rest of the program through the send object "pitchvel" (note number/velocity pairs) available to any identically named *receive* object elsewhere in the program.

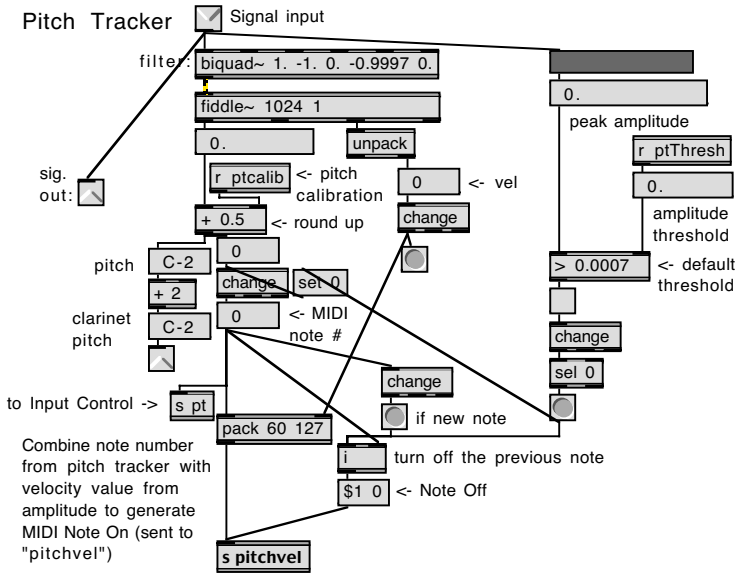

Figure 10.2. Software implementation of the pitch tracker

10.1.2 Output Devices

The interactive system generates MIDI data intended to control a multi-timbral modular synthesizer. I am using an E-Mu Proteus 2000 sound module (a later model of the Proteus I synthesizer used in the first performances). The composer leaves the choice

of synthesizer to the performer, but *Praescio IV* requires a synthesizer that can play distinct sounds on at least sixteen separate channels. The Proteus 2000 is more than adequate, with thirty-two available channels. Complete descriptions of the voice programs used for each Proteus 2000 channel are given in detail in table 10.4 below.

Physically, the MIDI inputs of the Proteus 2000 are connected to the outputs of the computer's MIDI interface. Audio outputs from the synthesizer are then connected to the sound reinforcement system consisting of a small mixing console (12 input/2 output) and a pair of self-powered loudspeakers.

10.1.3 Overview of Stage Setup

Pennycook recommends a compact stage setup, with the loudspeakers placed as close as possible to the performer in what he describes as a "chamber trio" configuration. The complete stage setup for the current realization of *Praescio IV* is shown in figure 10.3.

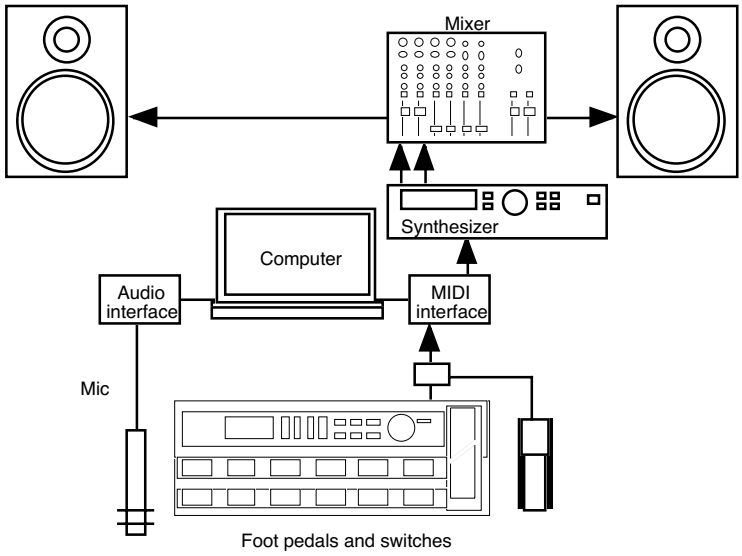

Figure 10.3. Complete Stage Setup for *Praescio IV*

10.2 INTERACTIVE MIDI SYSTEM

The interactive MIDI system processes live MIDI input and controls sequence playback and synthesizer output. In addition to the pitch-tracking functions described in section 10.1.1 (above), I have modified the performance software to provide flexible setup of external MIDI devices and a more convenient graphical user interface, shown in figure 10.4.

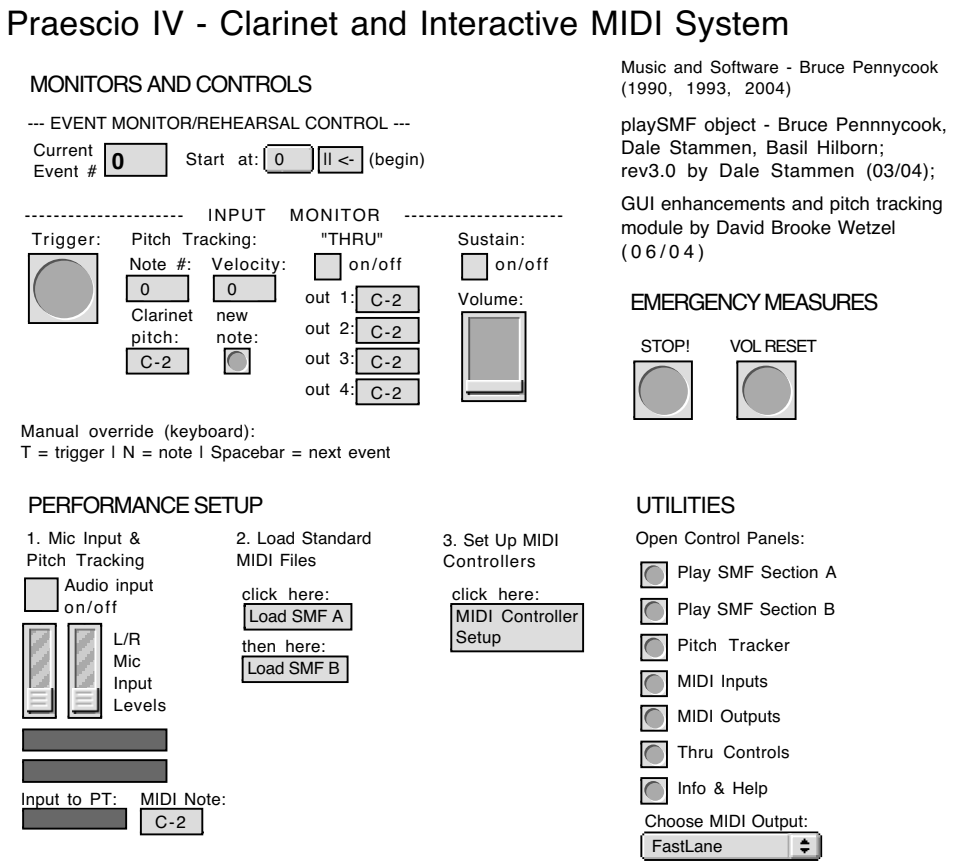

Figure 10.4. Interactive MIDI system software user interface

10.2.1 Prepared Data

The interactive system functions on the basis of prepared data files that are specific to the *Praescio IV* score. There are two categories of prepared data: event lists and MIDI sequences (SMFs).

Event Lists. The master event list is included for reference in Appendix B. In the current software implementation, event list processing is split into several stages. The result is a separate event lists database for each major function of the live processing program: input

control, SMF playback processing, and "thru" event processing. The first is a list of the trigger signals required for advancement to each score event. The second is a pair of lists of *PlaySMF* parameters for each event that includes MIDI file playback ("play" events). The third is a list of parameters for each of the designated "thru" events. Each subset of the event list is stored as a text file specially formatted for the Max/MSP *coll* object. Each line of the data file contains an index number or symbol, followed by a list of parameter values. A message received by the *coll* object that matches an index number or symbol stored in the file causes output of the associated list of values. Excerpts from the necessary *coll* files are shown in tables 10.1, 10.2, and 10.3, below.

The event control data file "p4-inputs.col.data" determines which trigger signal to look for before advancing to the next score event. The first number on each line is the index value, representing each score event in *Praescio IV*. The two numbers following the comma on each line determine the input signal that will advance the system to the event indexed. If the first value is 1, the system waits for the footswitch trigger. The second value, if non-zero, gives the MIDI note number to look for from the pitch tracker. As shown in table 10.1, events 1 and 2 are triggered by the footswitch. MIDI note numbers 50, 64, and 59 trigger events 3, 4, and 5 respectively.

Table 10.1. Excerpt from *coll* file 1: event control

```
1, 1 0;
2, 1 0;
3, 0 50;
4, 0 64;
5, 0 59;
```

Two data files, "p4col-A.data" and "p4coll-B.data" govern play event parameters. Each play event number is followed by a list of PlaySMF control parameters, including track number, playback channel, transposition level (semitones), tempo, and velocity scaling. Some pre-processing is included in the control module (discussed in section 10.2.2 below) to simultaneously play multiple sequence tracks during a single score event. For example, event 3 initiates playback of tracks 3a, 3b, and 3c, but with individual playback parameters.

Table 10.2. Excerpt from *coll* file 2: play events

```
3a, play tr3a chan 1 trans 10 vel 1.4;
3b, play tr3b chan 3 trans -2;
3c, play tr3c chan 12 trans –2;
4, play tr4 chan 4 11 trans –2;
5a, play tr5a chan 6 trans -4;
5b, play tr5b chan 1 trans -2;
```

A third data file, "p4col-thru.data," stores the list of "THRU" events and associated parameters. Event numbers are followed by a list of seven parameters: on/off (1/0), harmonizer 1-4 (transposition of the clarinet pitch; 50 = off), program change, and velocity scaling (as a percentage of the value from the pitch tracker). Table 10.3 shows parameters for THRU events 16-22 (no change to thru parameters in event 20). Event 18 turns the THRU module off; event 19 turns it back on, with synthesizer playing a four-note chord (tracked pitch + 0, 2, 7, and 12) for each note played by the clarinetist.

Table 10.3. Excerpt from *coll* file 3: THRU events

```
16, 1 0 50 50 50 65 100;
17, 1 0 -12 50 50 99 100;
18, 0 50 50 50 50 18 100;
19, 1 0 2 7 12 18 100;
21, 1 0 10 16 19 18 100;
22, 1 –1 -2 -4 50 18 100;
```

Standard MIDI Files (SMFs). Pennycook's MIDI sequences are currently formatted as two separate type-1 (multi track) standard MIDI files: "SMF-A" and "SMF-B." Each sequence is stored as a track, named for the event in which it is played back (as indicated in table 10.2, above). The playSMF object accesses and processes tracks separately, as if they were individual MIDI files. "SMF-A" contains 53 tracks, used in events 1 – 48 and "SMF-B" contains 34 tracks, used for events 49 through the end of the piece. The MIDI files are provided with the system software and are loaded into the playSMF object during the system setup routine prior to performance.

10.2.2 Event Processing

The system software contains three real-time data processing modules that correspond to the event list data files described in section 10.2.1 above. The event list control module manages advancement through the score in response to event trigger signals (footswitch and pitch tracker signals). The playback control module contains instances of the playSMF object and executes sequence playback at the appropriate points in the score. The THRU event control module manages MIDI note output, velocity

scaling, and program change messages in response to parameters given for each THRU event.

Event List Control. The event list control module compares incoming footswitch trigger and note data (from the pitch tracker) to the required conditions for advancement to the next system event. For example, event 2 (as shown in table 10.1, above) requires a signal from the footswitch to execute. At the start of the piece, the current event is 1 and the event list control module looks ahead to the set condition for triggering event 2 (specified in the data file "p4-Inputs.col.data"). The system then waits for the footswitch to be pressed before sending out the number '2' to the other processing modules. The event control module then sets the trigger for event 3 (MIDI note 50, or clarinet low E) as the condition for sending the number '3' to the next stage for processing. The complete event control module implementation in Max/MSP is shown in figure 10.5.

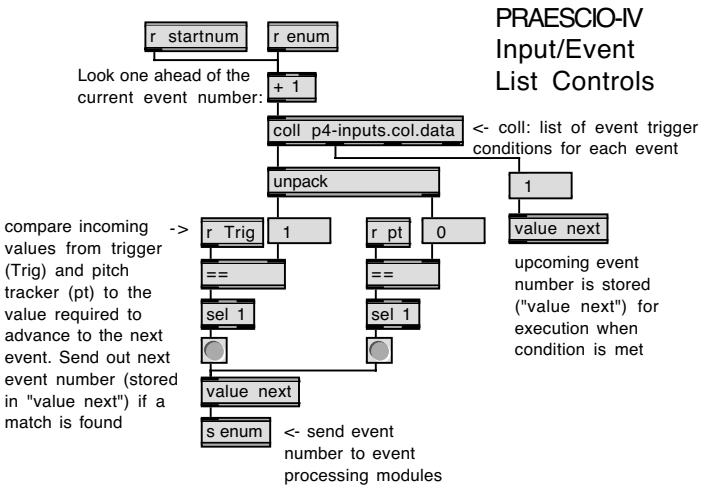

Figure 10.5. Event List control module

Play Events. The playback control module receives updated event numbers from the event list control module as signals to execute play events. If a number received matches an event number stored in the play event lists ("p4col-A.data" or "p4col-B.data"), the parameters stored at the appropriate index point(s) will be sent to the playSMF object, initiating MIDI sequence file playback. The output from playSMF is a stream of MIDI data that controls the external synthesizer.

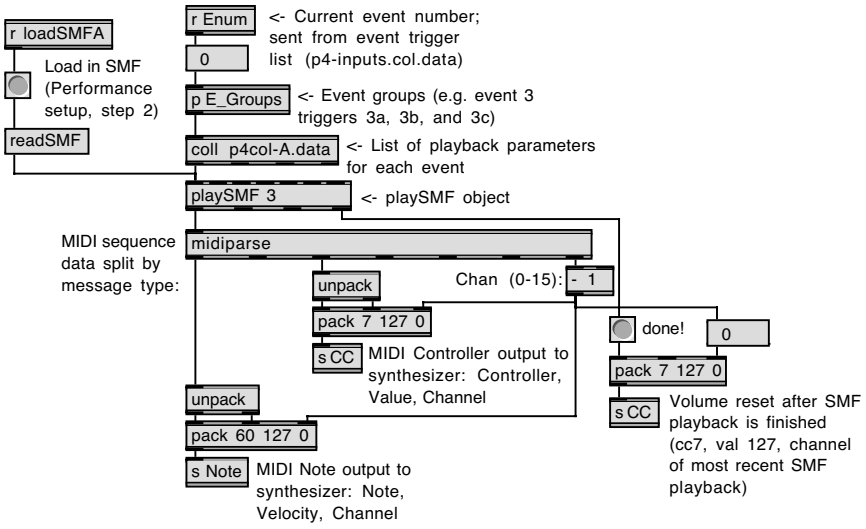

Figure 10.6. Play Event control module

THRU Events. Event numbers are sent to the THRU event control module and compared to the index field in the file "p4col-thru.data." A match with an event index in this list triggers output of THRU parameters (as shown in table 10.3 above) for further processing. A value of 0 in the channel field closes off output from the THRU module; a value of 1 enables MIDI output to the synthesizer. The four "harm" fields specify transposition intervals (from –20 to +49, 50 = off) to play based on the note value

received. The “program change” field specifies a change to a new synthesizer program for THRU notes. The “velocity scaling” field specifies a percentage by which to multiply velocity values of notes received from the pitch tracker. The complete Max/MSP implementation of the THRU event control module is shown in figure 10.7.

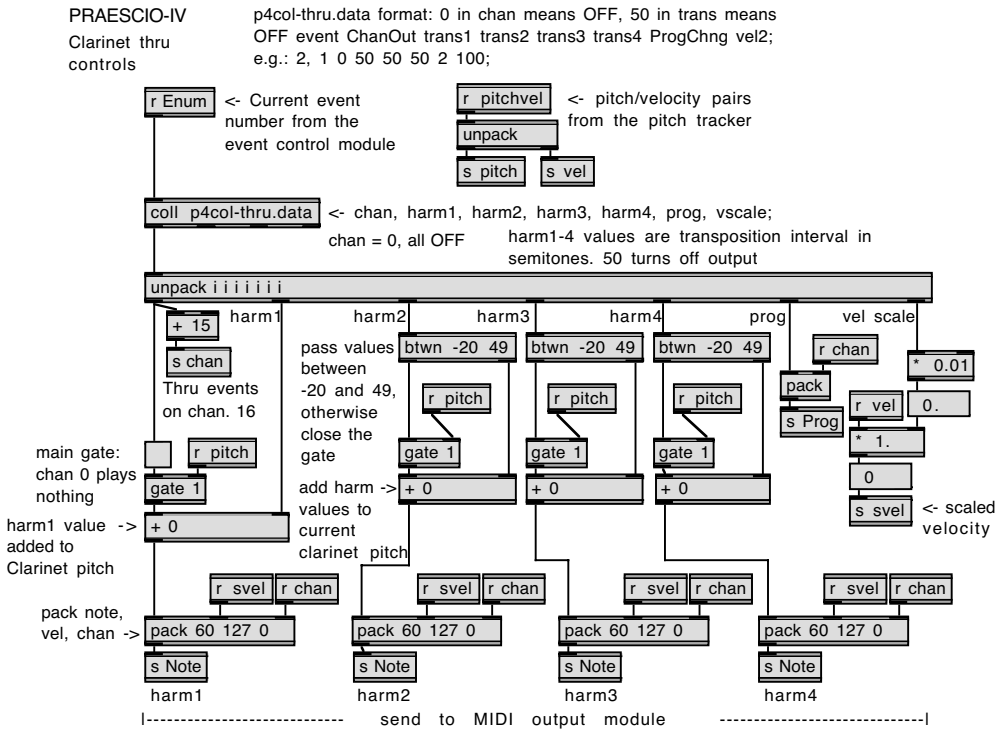

Figure 10.7. THRU Event control module

MIDI Output. The MIDI output module routes MIDI data generated by the play event and THRU event control modules to the external synthesizer. The MIDI output module is shown in figure 10.8.

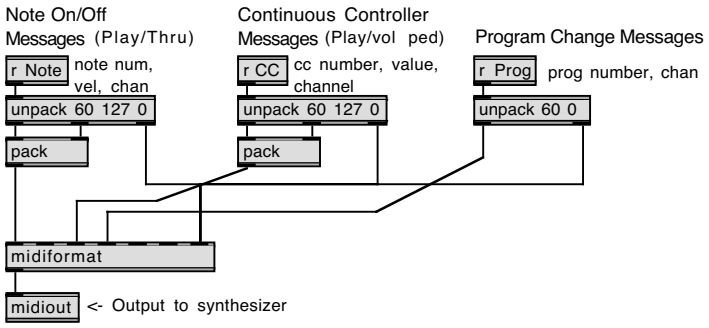

Figure 10.8. MIDI output module

10.3 SYNTHESIZER

I have opted to use an E-Mu Proteus 2000 external modular synthesizer for my performance of *Praescio IV*. The Proteus 2000 is a more recent generation of the synthesizer Pennycook and Boisvert used in the original 1990 production, although the available sound set is completely different. Therefore, the synthesizer part must be completely re-orchestrated according to the guidelines given by the composer and presented in chapter 6. Subsequent realizations are likely to use a different synthesizer and therefore, a different sound set. Table 10.4 shows my Proteus 2000 orchestration for this realization of *Praescio IV*. Program assignments for each of the 16 MIDI output channels are given with brief descriptions of the sound.

Table 10.4. Proteus 2000 sound set for *Praescio IV*

Channel	Proteus 2000 program	Description
1	VS37 (custom)	Layered program: organ, noise, rattling metal samples, filtered, with diffuse attack and long decay
2	Stereo Piano (preset)	Sampled stereo piano
3	Silk OBX Saws (modified preset)	Large string-like analog saw wave sound (emulation of the classic Oberheim Expander) with long decay and smoothed tone (low-pass filtered)
4	Analog (preset)	Large analog synthesizer string section
5	Chime (preset)	Organ layered with high bells
6	SawSweep Comp (preset)	Brass-like analog synth sound with sharp attack and filter sweep under LFO control
7	Dynamic Grand (preset)	Sampled grand piano
8	P5 Brass (preset)	Synthesized brass section (analog emulation)
9	Rich Analogs (preset)	Synthetic string section (analog emulation) with flange effect
10	PianoString 1 (preset)	Piano with diffuse string background (long decay)
11	Chime (preset)	Same as #5
12	Breather (preset)	Synth flute, sampled flute, pan flute and wind noise; diffuse attack and long decay
13	Breather Horn (custom)	Same processing as #12, but with brass samples replacing the flutes
14	Rich Analogs (preset)	Same as #9
15	Cast Teller (preset)	Electric guitar, as close to the Stratocaster as the Proteus 2000 will allow
16	THRU Cl Sust (custom)	Digital strings, chime, metal noise, IP wave with filter sweep

10.4 SUMMARY

The interactive MIDI system presented in this chapter is a re-implementation of the original MIDI-live version of 1991. Any substantive changes to the sequences, event lists, or functionality of the core MIDI processing software, as compared to the original version, were made by the composer and should be considered as mild revisions of the work itself. However, new implementations of the interactive software system, including

updates to the processing modules, the use of new tracking or control equipment, and the use of alternate synthesizers do not represent fundamental changes to the composition, so long as the functionality of the system is the same as presented here.

The software provided by the composer assumes a hardware setup that matches the original performances. Therefore, the performer is at this point responsible for supplying pitch tracking and synthesis modules as well as physical control devices (footswitches and pedal). Future versions of the performance software may incorporate these functions in a way that allows flexibility and ease of use for a wider pool of interested performers.

At the present time, *Praescio IV* is still somewhat limited to a specific set of equipment. The essential Max/MSP object playSMF is compiled so far only for Apple Macintosh computers running MacOS 9.2. This operating system has been outdated since the introduction of MacOS X in 2000 and is currently unsupported by Apple Computer. Future performances using this software will require either another update of the playSMF object or a work-around solution that emulates its functions in another way.