CHAPTER 9:
PERFORMANCE REALIZATION OF THEA MUSGRAVE'S *NARCISSUS*

My primary motivation for pursuing the detailed analyses of the works discussed in chapters 5 through 8 was to prepare the way for actual performance realizations that would be faithful to the original intentions of the composer, regardless of the specific equipment or technology used. For the lecture-recital presentation accompanying this document, I have created a new realization of the digital delay system for Thea Musgrave's *Narcissus*. This realization is based on the delay system structure, variable parameters, and control values outlined in my analysis presented in chapter 5. The digital delay for my current realization is implemented entirely in software (using Max/MSP), with real-time system control via MIDI foot pedal board. The delay system software will run unmodified on general-purpose Macintosh or Windows personal computers. Furthermore, the graphical nature of the Max/MSP programming environment with which it was created makes the software itself relatively easy to understand. Future enhancements to the software's graphic interface may allow for more flexibility and ease of use by other performers interested in performing this work. The software presented in this chapter is a prototype for the purposes of demonstrating an implementation of my technical analysis. Besides its demonstration value, it provides the core functionality for my own performance and the potential for a subsequent version for public distribution.

9.1 SOUND SYSTEM & STAGE SETUP

The sound system used for this presentation closely follows the recommendations found in Musgrave's score. In fact, Musgrave's stage diagram is a fairly accurate description of my own setup with the exception that in place of three separate foot pedals I use an integrated multi-footswitch device. My performance setup includes a microphone pair (Shure SM-57 dynamic and AKG C-409 condenser), an Apple Macintosh G3 laptop computer (running the delay system software) with an external audio interface (M-Audio FireWire 410) and MIDI interface (MOTU FastLane USB), a MIDI footswitch controller (DigiTech RP-20), and a pair of self-powered loudspeakers. This setup is shown in figure 9.1, and is explained in further detail below.

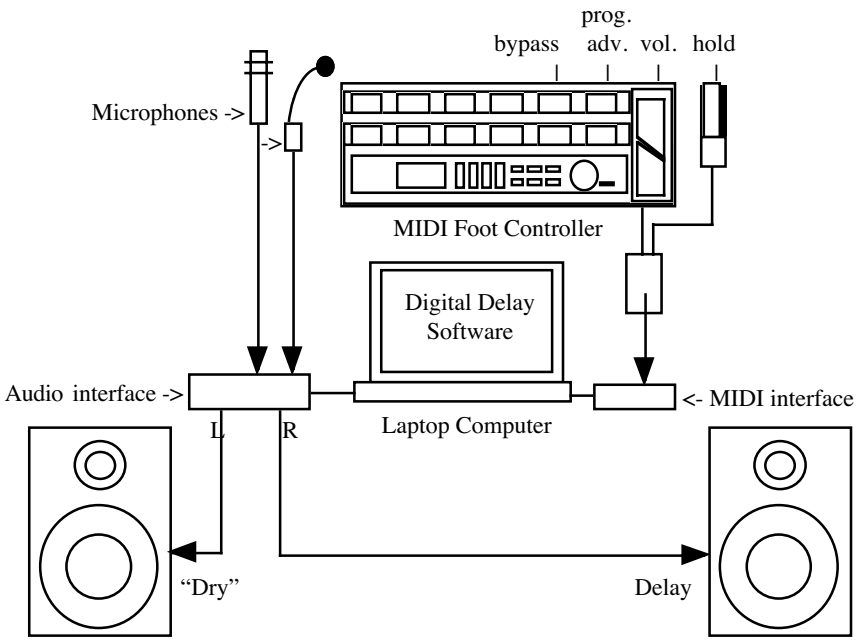

Figure 9.1. Stage setup for a new realization of *Narcissus*

9.1.1 Input

Microphones. Rather than using a contact microphone (as recommended by the composer), I have opted for the better sound quality obtained by a combination of a Shure SM-57 dynamic microphone (placed at the mid-point of the clarinet) with a clip-on AKG C-409 condenser microphone (placed below the bell). More care in speaker placement is required with this arrangement to avoid unwanted feedback, but I find the tradeoff is worthwhile to avoid the poor sound quality of a contact microphone for live sound reinforcement.

Computer Audio Input. Both microphones are plugged directly into the front panel of an M-Audio FireWire 410 audio interface connected to an Apple G3 Powerbook computer running Macintosh OS 9.2.[93] This same software will function just as well using alternate audio interface devices, including the computer's built-in audio inputs (controlled by Apple Sound Manager software). At this point, all further audio routing is done in the Max/MSP software application, described below.

[93] As of this writing, Mac OS 9.2 has been obsolete and unsupported by Apple Computer for over three years. However, my current computer runs MacOS 9.2, which is required for performance of Pennycook's *Praescio IV,* discussed in chapter 10. The realization of Musgrave's *Narcissus* presented here should run equally well on MacOS X, the current operating system from Apple Computer, as well as on computers running Microsoft Windows XP, though neither of these systems have been tested so far. The financial resources required for performers to keep up with rapid technology turnover is one of the motivating factors for this research, and is a fruitful topic for future discussions beyond the scope of this document.

MIDI Footswitch Controller. I am currently using a Digitech RP-20 footswitch controller to send MIDI control signals to the digital delay system software. The RP-20 has ten footswitches that can be individually assigned to transmit specific MIDI controller messages with values of 127 (on) or 0 (off). The RP-20 also has a volume pedal that transmits MIDI controller values in a continuum from 0 to 127. For the current realization of *Narcissus*, footswitches 9 and 10 are assigned to transmit MIDI controllers 64 and 60 respectively. Controller 64 is used to control Bypass, and controller 60 is used to advance through the list of score events, as described in section 9.3.3, below.

The RP-20's MIDI output is connected to the input of the computer's MIDI interface (MOTU Fastlane). A MIDI pedal-merge device is connected between the RP-20 output and the computer's MIDI input in order to accommodate a piano-style sustain pedal (transmitting MIDI controller 65), used to control the Delay Hold.

9.1.2 Output

Outputs 1 and 2 of the FireWire 410 interface are sent directly to a pair of self-powered loudspeakers, placed as indicated in Musgrave's diagram. Computer output 1 is the unaffected "dry" signal from the clarinet microphone, routed directly to speaker 1 (left), and computer output 2 is the digital delay signal, routed to speaker 2 (right).

9.2 DELAY SYSTEM

The digital delay is designed to emulate the functions of the Vesta Koza DIG-411 digital delay system used by the composer. Each of the main features of the DIG-411

called for in Musgrave's score is recreated in Max/MSP software according to the techniques and parameters described in chapter 5. Figure 9.2 shows the complete audio processing system used to implement the digital delay. The input signal from the microphone is sent to a multi-tap delay line, which is subject to dynamic modification by individual control modules for Delay Time, Feedback, Modulation, Hold, Volume, and Bypass.

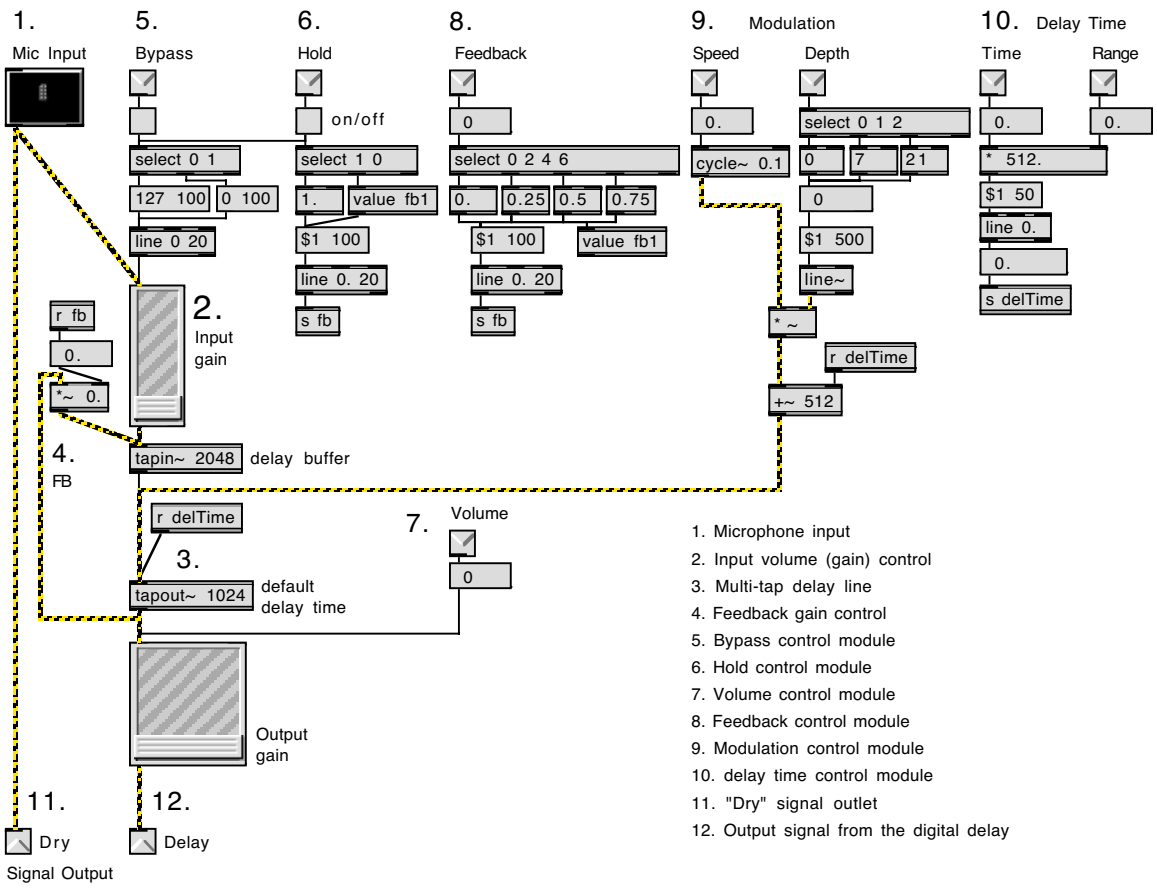

Figure 9.2. Software implementation of the delay system for *Narcissus*

9.2.1 Multi-tap Delay Line

The delay line is implemented using the Max/MSP tapin~ and tapout~ objects. These two objects are designed to work in tandem to create a "multi-tap" delay system. Sound is recorded into a delay buffer (tapin~) and delay time is set as a "tap" point in milliseconds (tapout~). My implementation of the delay line for *Narcissus* uses a delay buffer that is twice as long as needed (2048 milliseconds), and delay time is controlled by sending new time values to tapout~ (default 1024, or "512 x 2" as found in the score).

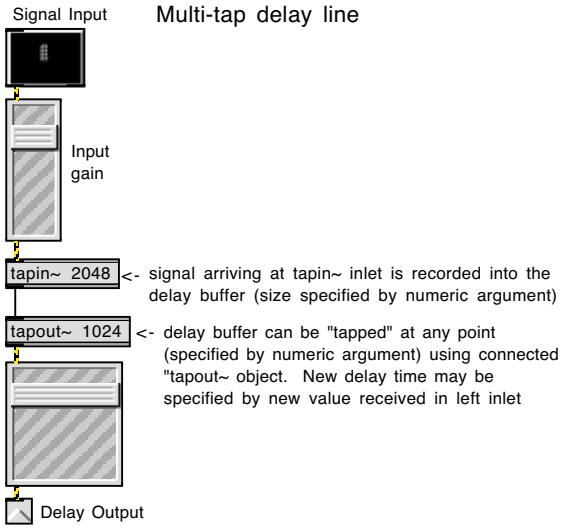

Figure 9.3. Multi-tap delay line

9.2.2 Delay Time

Delay Time is implemented in a way that reflects the functions of the DIG-411, in that separate values for "Time" and "Range" are maintained. Therefore, a score notation of "512 X 2" is implemented by actually multiplying the "Range" value of 512 by the

"Time" value of 2 (rather than simply using the value "1024"). I have done this in order to maintain consistency and avoid confusion when comparing the functions of the software implementation to the relevant events in the published score. The complete implementation of the Delay Time control module is shown in figure 9.4.

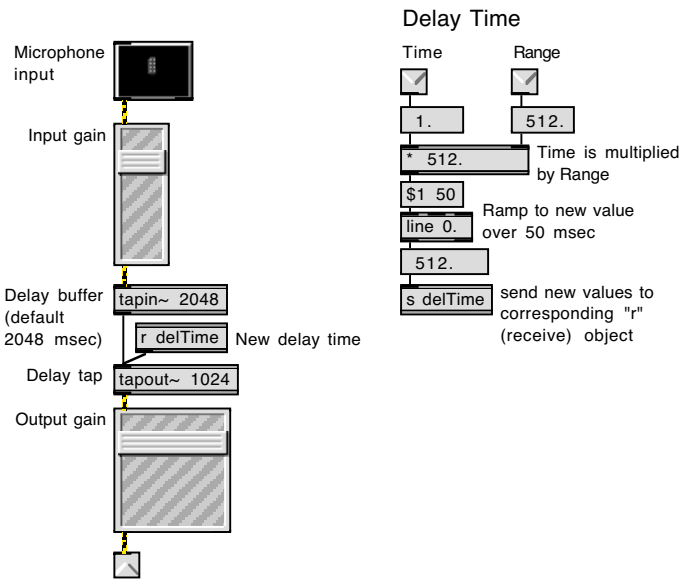

Figure 9.4. Delay time control module

9.2.3 Feedback

Feedback is implemented by routing the delay line output back to its own input. Feedback gain is dynamically attenuated according to signals received from the Feedback control module. Feedback values 0, 2, 4, and 6 from the score are translated to values 0, .25, .5, and .75, as indicated by the analysis presented in chapter 5. The complete implementation of the Delay Feedback system is shown in figure 9.5.

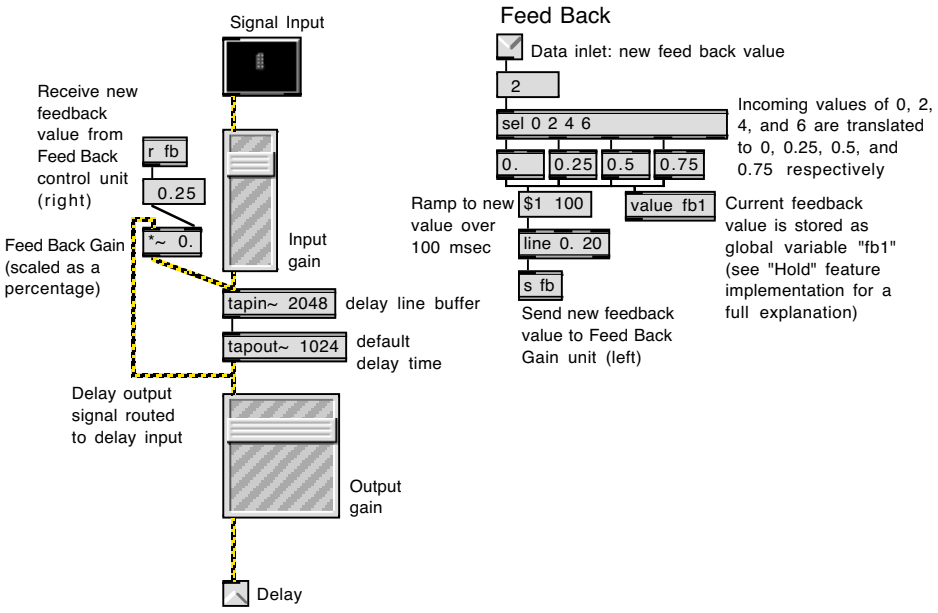

Figure 9.5. Delay Feedback control module

9.2.4 Modulation

Delay time modulation is implemented in software by applying a low frequency oscillator (LFO) to the delay time. Following the analysis presented in chapter 5, the LFO frequency is set to 0.1 Hz by default, and incoming Modulation Depth values of 0, 1, or 2 are translated to 0, 7, and 21 milliseconds, respectively. The output of the LFO (a stream of values oscillating between -1 and 1 at the prevailing audio sampling rate, 44.1 kHz) is used to scale the Modulation Depth values. When the product of the Modulation depth and the LFO output is added to the current Delay time value, the result is a delay time that varies from the original value by plus or minus the Modulation Depth value at a rate set by the LFO frequency. In other words, a Modulation Depth setting of 1, with a Delay Time of 512 milliseconds, causes a delay time that cycles from 505 to 519

milliseconds and back again over a period of about ten seconds. The actual implementation of this concept is shown in figure 9.6.

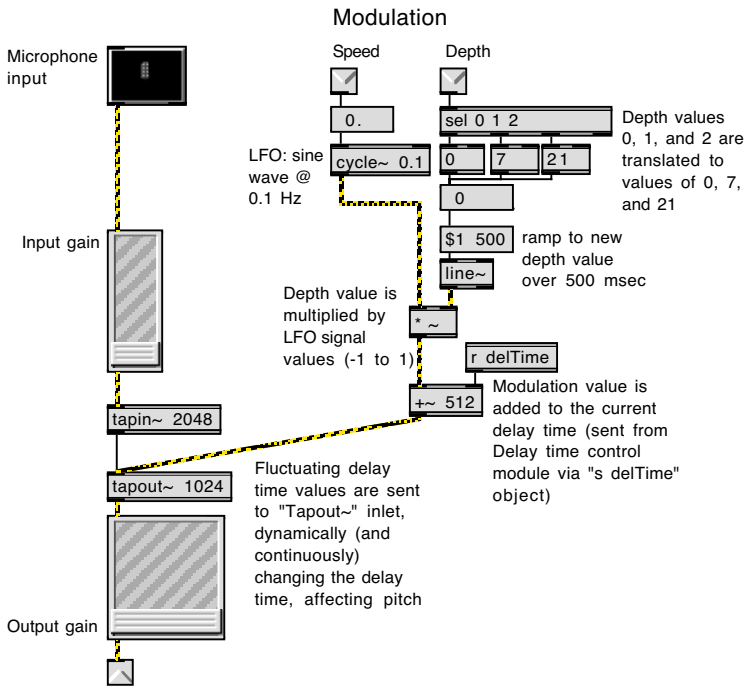

Figure 9.6. Modulation Control Module

9.2.5 Hold

The Delay Hold is implemented by simply engaging the Bypass (described below) while simultaneously setting delay feedback to 100%. When the Delay Hold is released, Bypass is disengaged, and Feedback is returned to its previous level. The effect is that while the Delay Hold is engaged, sound currently recorded in the delay buffer will continue to re-circulate indefinitely, but no new sounds from the microphone will be added to the delay line. Figure 9.7 shows the Delay Hold both on and off.

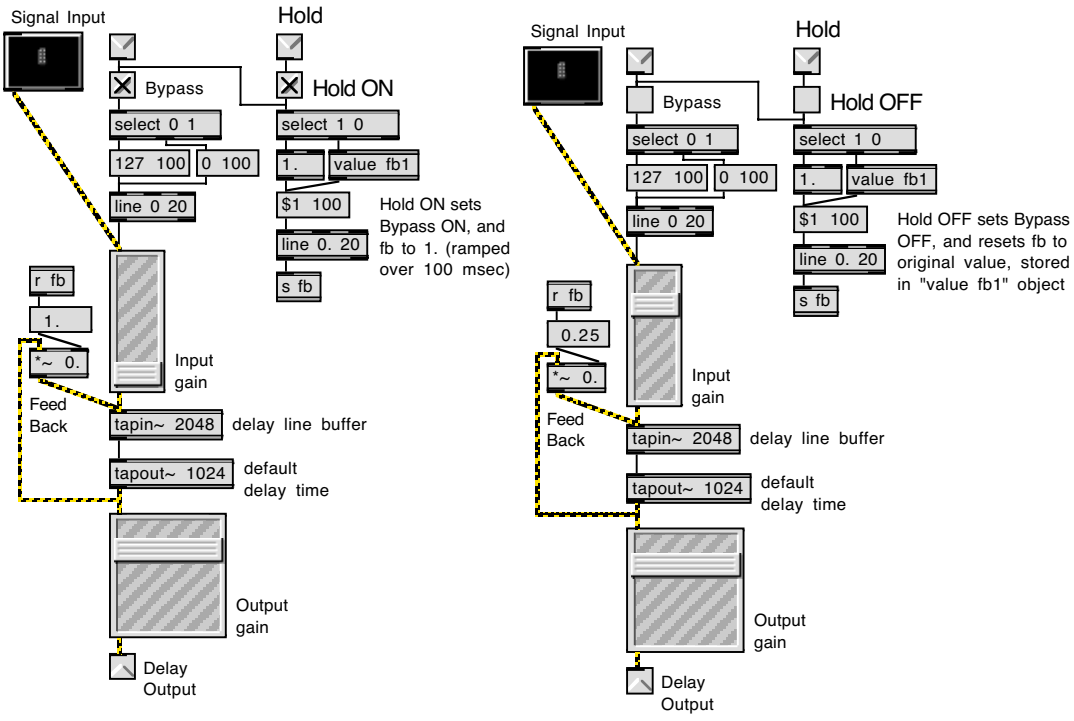

Figure 9.7. Software implementation of Delay Hold

9.2.6 Volume

Output volume of the Digital Delay system is controlled directly from the MIDI foot pedal without further software processing. Volume control must be placed in the signal chain after delay processing in order to accomplish the dynamic effects notated in the score. For example, the passage shown in figure 9.8 requires sound to be circulating in the delay system before the volume pedal is raised.

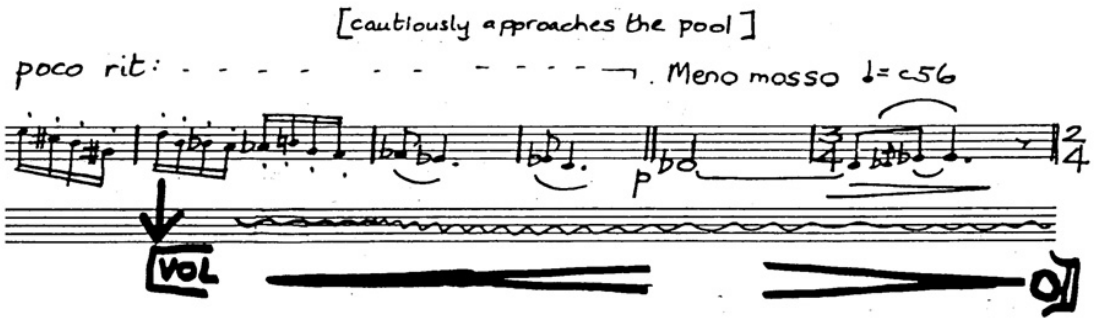

Figure 9.8. Score example: application of the volume pedal

I have implemented the volume control as shown in figure 9.9.

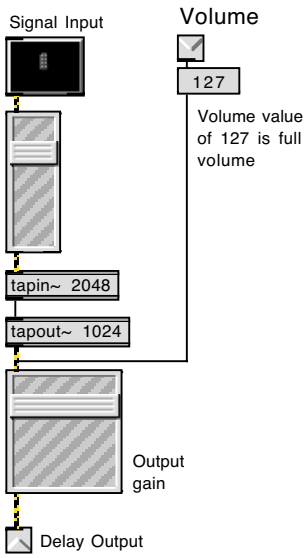

Figure 9.9. Software implementation of the Volume pedal

9.2.7 Bypass

Bypass is implemented by cutting the signal input to the digital delay system at the preprocessing stage. When Bypass is engaged, input is ramped from its current level to zero, over a period of 100 milliseconds (in order to avoid audible clicks caused by the abrupt cutoff of a digital signal). When bypass is disengaged, signal input to the delay is restored to its previous level.

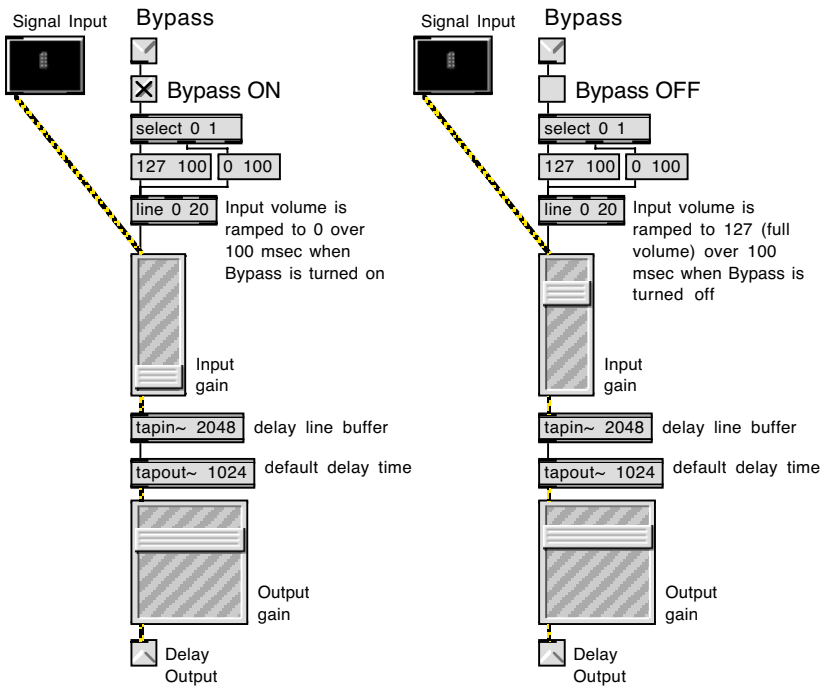

Figure 9.10. Bypass Control Module

9.3 CONTROL SYSTEM

The control system is implemented as a system of MIDI inputs to the software. Bypass, Hold, and Volume are controlled directly by foot switches or foot pedal. Changes to Delay Time, Feedback, and Modulation are controlled by software processing, in response to a single footswitch trigger.

9.3.1 MIDI Input Processing

MIDI input from the foot pedals is processed by a fairly simple software module. Non-zero values transmitted by controllers 64 and 65 turn on the Bypass and Hold modules, respectively (a zero value turns the module off). Volume messages from the pedal are simply passed through. Controller 60 messages (any value) trigger advancement to the next score event (as described in section 9.3.3).

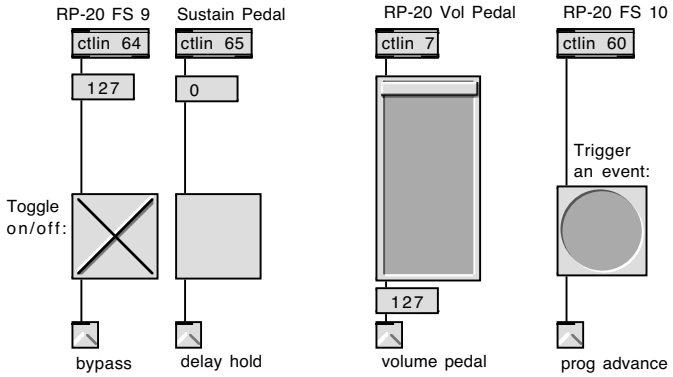

Figure 9.11. MIDI input controls

9.3.2 Score Event Processing

The “prog. advance” trigger shown in figure 9.11 (above) is passed through to the Score-Event processing module, shown in figure 9.12. Program-advance trigger signals are sent to the Max object *coll* which contains a list of Delay Time, Feedback, and Modulation parameters for a sequence of score events shown in detail previously in Table 5.2 (chapter 5, above). Each time a MIDI controller 60 message is received (in response to pressing the RP-20 footswitch 10), the *coll* object sends out the array of parameter values for the next event in the list.

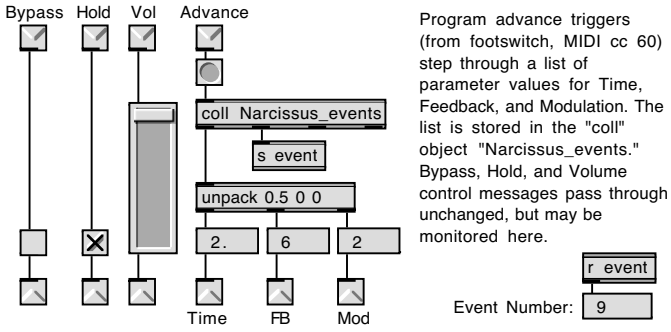

Figure 9.12. Score Event processing module

Figure 9.12 (above) shows the delay system parameters, “Time,” “Feedback,” and “Mod,” set for event #9, as shown in the score example, figure 9.13, below. Delay Time is set to “512 X 2” (1024 milliseconds), Feedback to 6, and Modulation to 2. Hold is engaged, and Volume is up.

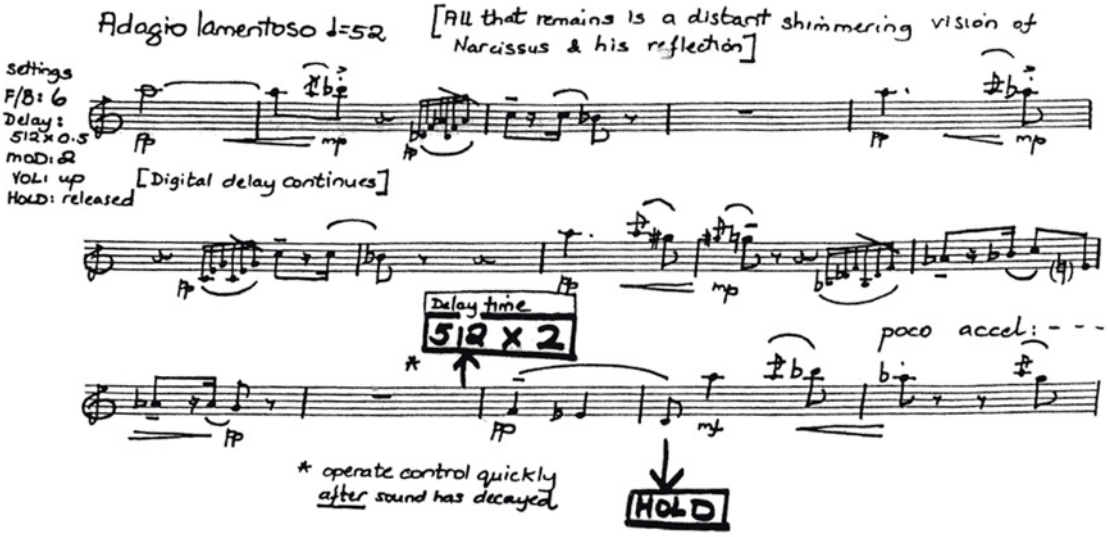

Figure 9.13. Score example: event 9, with hold engaged

The *coll* object stores its data in an external text file ("Narcissus_events"), which is read into memory when the program loads. The contents of this file, shown in table 9.1, correspond to the sequence of parameter changes found in Table 5.2 (chapter 5, above).

Table 9.1. Contents of *coll* file "Narcissus_events"

Event	Time	FB	Mod
1,	0.5	0	0;
2,	0.5	2	0;
3,	1.0	4	0;
4,	1.0	6	0;
5,	0.5	4	0;
6,	0.5	6	0;
7,	0.5	6	1;
8,	0.5	6	2;
9,	2.0	6	2;
10,	2.0	6	1;
11,	2.0	6	0;

9.3.3 Linkage of Control and Processing Modules

Each of the program components described so far, the Digital Delay system, the Control Interface, and the Score Event processor, are actually subprograms within a larger software application. These components are all linked together in a top-level program that routes data between the individual parts. Figure 9.14 shows the data connections between the three main components of the performance software.

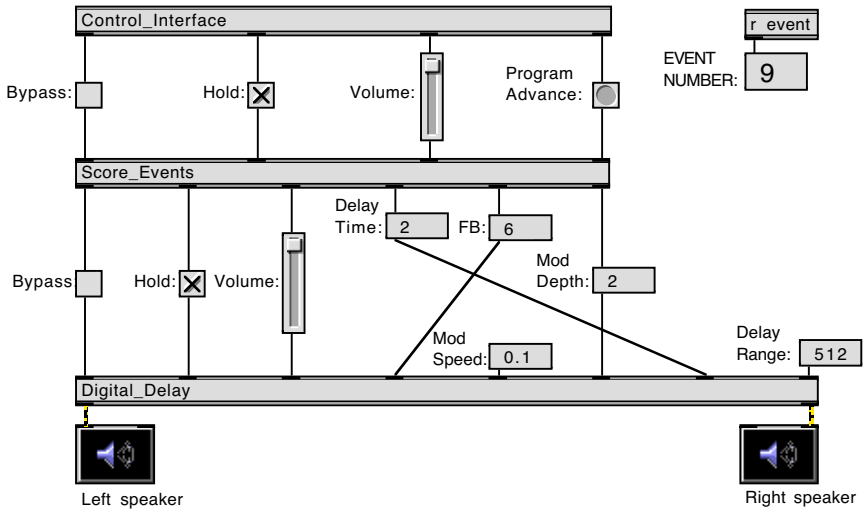

Figure 9.14. Data routing between control and signal processing subprograms

9.4 SUMMARY

The Max/MSP implementation of the digital delay system for Thea Musgrave's *Narcissus* is fairly simple. Each component of the delay system described in the analysis in chapter 5 is recreated in a manner that reflects the functions of the original instrument as much as possible while at the same time allowing a great deal of flexibility. The software version presented here compensates for the lack of physical controls found on

the original instrument (for Delay Time, Feedback, and Modulation) by placing these elements under the control of a single footswitch that advances through a list of program changes.

This software could be used by other performers running Macintosh or Windows computers at the present time. Further enhancements to the user interface, including provisions for event monitoring, reassigning MIDI controllers, and a menu for choosing rehearsal start-points, would make the experience more "user-friendly." These enhancements will be incorporated soon, but the present version shown in this chapter is fully functional and ready for use on stage.