ANALYSIS, RECONSTRUCTION, AND PERFORMANCE OF INTERACTIVE ELECTROACOUSTIC WORKS FOR CLARINET AND OBSOLETE TECHNOLOGY:
SELECTED WORKS BY MUSGRAVE, PENNYCOOK, KRAMER, AND LIPPE

by

David Brooke Wetzel

Copyright © David Brooke Wetzel 2004

A Document Submitted to the Faculty of the SCHOOL OF MUSIC AND DANCE
In Partial Fulfillment of the Requirements For the Degree of DOCTOR OF MUSICAL ARTS WITH A MAJOR IN MUSIC
In the Graduate College THE UNIVERSITY OF ARIZONA

2004
THE UNIVERSITY OF ARIZONA ©
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have read the
document prepared by David Brocks Wetzel
entitled ANALYSIS, RECONSTRUCTION, AND PERFORMANCE OF INTERACTIVE
ELECTROACOUSTIC WORKS FOR CLARINET AND OBSOLETE TECHNOLOGY:
SELECTED WORKS BY MUSGRAVE, PENNYCOOK, KRAMER, AND LIPPE

and recommend that it be accepted as fulfilling the requirements for the Degree
of DOCTOR OF MUSICAL ARTS

Professor Jerry Kirkbride
Dr. William Dietz
Dr. Kelland Thomas

Final approval and acceptance of this document is contingent upon the candidate's
submission of the final copy of the document to the Graduate College.

I hereby certify that I have read this document prepared under my direction and
recommend that it be accepted as fulfilling the requirement.

Professor Jerry Kirkbride  8/11/04
STATEMENT BY AUTHOR

This document has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this document are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: David B. Wetzel
To Allan Brooke Wetzel, Ph. D.

(1933 – 2003)
ACKNOWLEDGMENTS

This project would not have been possible without the generosity and kind assistance of a number of individuals. First, I would like to thank my teacher, Jerry Kirkbride, not only for supporting this project, but also for asking the difficult questions that helped me to clarify the need for doing this work. I would also like to thank Dr. Janet Sturman for many enlightening and inspiring discussions concerning the nature of musical repertoire and the issues involved in the preservation and diffusion of technology-driven musical works.

I owe an enormous debt of gratitude for the kind help I have received from the four composers whose works are analyzed in this document: Thea Musgrave, Bruce Pennycook, Jonathan Kramer, and Cort Lippe. Through our correspondence I have received many perceptive comments and kind feedback, as well as critical help in understanding the technical details of the interactive electroacoustic systems used in each work. I am especially grateful to Jonathan Kramer, not only for his tireless help and quick responses in straightening out the details of my analysis of Renascence, but also for inspiring me to pursue interactive electroacoustic performance studies more than twelve years ago. I am deeply saddened to hear of his passing in June of this year. He will be sorely missed.

A number of noted performers of interactive electroacoustic music were also instrumental in supporting my research. My deepest thanks go to flutist Wendy Rolfe for allowing me to analyze first-hand the Vesta Koza DIG-411 used by Musgrave to compose Narcissus. Also helpful in understanding the requirements of Musgrave’s work were the comments and perspectives of clarinetist F. Gerrard Errante and my former advisor and computer music teacher, Dr. McGregor Boyle.

I would like to thank Kim Freyermuth and Central Arizona College for technical support and the loan of the E-Mu Proteus 2000 synthesizer for my lecture-recital. I am also grateful for the logistical, technical, and moral support I have received from Lyneen Elmore and the University of Arizona School of Music staff.

Todd J. Niquette has given most generously of his time and expertise; acting as technical assistant for all three academic recitals I have given that feature interactive electroacoustic music over a twelve-year span. I could not ask for a more faithful friend.

Finally, and most importantly, I could not have done any of this work without the love and support of my wife, Meredith J. Soyster. She has helped me keep my focus, forced me to clarify my ideas, and given constant encouragement to see this project through to completion. Words cannot fully express my gratitude.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>13</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>16</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>19</td>
</tr>
<tr>
<td>CHAPTER 1. INTENT AND SCOPE OF THIS STUDY</td>
<td>20</td>
</tr>
<tr>
<td>CHAPTER 2. OVERVIEW OF INTERACTIVE ELECTROACOUSTIC MUSIC AND PERFORMANCE PRACTICE: TOOLS, TECHNIQUES AND TERMINOLOGY</td>
<td>23</td>
</tr>
<tr>
<td>2.1. WHAT IS INTERACTIVE ELECTROACOUSTIC MUSIC?</td>
<td>23</td>
</tr>
<tr>
<td>2.2. HISTORICAL FOUNDATIONS OF INTERACTIVE ELECTROACOUSTIC MUSIC</td>
<td>24</td>
</tr>
<tr>
<td>2.3. GENERAL PURPOSE ELECTRONICS AS MUSICAL INSTRUMENTS</td>
<td>26</td>
</tr>
<tr>
<td>2.3.1. Microphone Feedback Processing</td>
<td>26</td>
</tr>
<tr>
<td>2.3.2. Tape Recorders</td>
<td>27</td>
</tr>
<tr>
<td>2.3.3. Tone Generators, Ring Modulators, Filters, and Other Equipment</td>
<td>27</td>
</tr>
<tr>
<td>2.4. SYNTHESIZERS AND AUDIO SIGNAL PROCESSING INSTRUMENTS</td>
<td>28</td>
</tr>
<tr>
<td>2.4.1. Synthesizers</td>
<td>29</td>
</tr>
<tr>
<td>2.4.2. Samplers</td>
<td>31</td>
</tr>
<tr>
<td>2.4.3. Effects Processors</td>
<td>32</td>
</tr>
<tr>
<td>2.5. THE MIDI STANDARD</td>
<td>33</td>
</tr>
<tr>
<td>2.5.1. MIDI Message Types</td>
<td>34</td>
</tr>
<tr>
<td>2.5.2. Extensions to MIDI</td>
<td>35</td>
</tr>
<tr>
<td>2.6. EXPERIMENTAL INTERACTIVE SOFTWARE SYSTEMS</td>
<td>36</td>
</tr>
<tr>
<td>2.6.1. Cypher</td>
<td>36</td>
</tr>
<tr>
<td>2.6.2. MIDI Live</td>
<td>37</td>
</tr>
<tr>
<td>2.6.3. 4x</td>
<td>38</td>
</tr>
<tr>
<td>2.7. WIDELY-USED GENERAL-PURPOSE INTERACTIVE SOFTWARE SYSTEMS</td>
<td>38</td>
</tr>
<tr>
<td>2.7.1. Max/MSP</td>
<td>39</td>
</tr>
<tr>
<td>2.7.2. Pd</td>
<td>43</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS -- Continued

2.7.3. SuperCollider ................................................. 44
2.7.4. Kyma ............................................................ 44
2.8. SUMMARY .......................................................... 45

CHAPTER 3. INTERACTIVE ELECTROACOUSTIC MUSIC AND THE
PROBLEM OF TECHNOLOGICAL OBSOLESCENCE .................. 47

CHAPTER 4. FOUR INTERACTIVE ELECTROACOUSTIC WORKS FOR
CLARINET AND OBSOLETE TECHNOLOGY: AN OVERVIEW ........ 52

CHAPTER 5. ANALYSIS OF TECHNOLOGY COMPONENTS IN THEA
MUSGRAVE'S NARCISSUS (1987) FOR B-FLAT CLARINET AND
DIGITAL DELAY ..................................................... 56
5.1. HISTORICAL BACKGROUND ...................................... 56
5.2. MUSICAL ROLE OF TECHNOLOGY ................................ 58
5.3. ANALYSIS OF TECHNOLOGY COMPONENTS ....................... 60
5.3.1. Sound Reinforcement ......................................... 60
5.3.2. Delay system .................................................. 61
  Delay time ..................................................... 62
  Delay Feedback .................................................. 63
  Modulation ..................................................... 66
  Hold .......................................................... 69
  Volume ....................................................... 70
  Bypass ......................................................... 70
  Digital Delay System Summary .................................... 71
5.3.3. Control interface ............................................. 73
5.4. SUMMARY ........................................................ 75

CHAPTER 6. ANALYSIS OF TECHNOLOGY COMPONENTS IN BRUCE
PENNYCOOK'S PRAESCIO IV (1990) FOR CLARINET AND
INTERACTIVE MIDI SYSTEM ......................................... 76
6.1. HISTORICAL BACKGROUND ...................................... 76
6.2. MUSICAL ROLE OF TECHNOLOGY ................................ 78
6.3. ANALYSIS OF TECHNOLOGY COMPONENTS ....................... 79
### TABLE OF CONTENTS – Continued

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.1. Sound Reinforcement</td>
<td>80</td>
</tr>
<tr>
<td>6.3.2. Control interface</td>
<td>80</td>
</tr>
<tr>
<td>Event Trigger</td>
<td>81</td>
</tr>
<tr>
<td>Pitch Tracking</td>
<td>82</td>
</tr>
<tr>
<td>Sustain Pedal</td>
<td>83</td>
</tr>
<tr>
<td>Volume Controller</td>
<td>83</td>
</tr>
<tr>
<td>6.3.3. Synthesizer</td>
<td>84</td>
</tr>
<tr>
<td>6.3.4. Prepared Data</td>
<td>85</td>
</tr>
<tr>
<td>Event List</td>
<td>85</td>
</tr>
<tr>
<td>MIDI Sequences</td>
<td>86</td>
</tr>
<tr>
<td>6.3.5. Event Processing</td>
<td>88</td>
</tr>
<tr>
<td>Input Processing</td>
<td>89</td>
</tr>
<tr>
<td>Play Event Processing</td>
<td>89</td>
</tr>
<tr>
<td>THRU Event Processing</td>
<td>91</td>
</tr>
<tr>
<td>6.4. SUMMARY</td>
<td>93</td>
</tr>
</tbody>
</table>

### CHAPTER 7. ANALYSIS OF TECHNOLOGY COMPONENTS IN JONATHAN KRAMER’S *RENAASCENCE* (1974) FOR CLARINET, TAPE, AND TAPE DELAY

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1. HISTORICAL BACKGROUND</td>
<td>95</td>
</tr>
<tr>
<td>7.2. MUSICAL ROLE OF TECHNOLOGY</td>
<td>96</td>
</tr>
<tr>
<td>7.3. ANALYSIS OF TECHNOLOGY COMPONENTS</td>
<td>100</td>
</tr>
<tr>
<td>7.3.1. Sound Reinforcement</td>
<td>102</td>
</tr>
<tr>
<td>7.3.2. Delay System</td>
<td>103</td>
</tr>
<tr>
<td>Delay Time</td>
<td>104</td>
</tr>
<tr>
<td>Delay Feedback</td>
<td>105</td>
</tr>
<tr>
<td>7.3.3. Control System</td>
<td>105</td>
</tr>
<tr>
<td>Click Track</td>
<td>106</td>
</tr>
<tr>
<td>Signal Matrix</td>
<td>107</td>
</tr>
<tr>
<td>Score Events</td>
<td>108</td>
</tr>
<tr>
<td>7.3.4. Pre-recorded Sounds</td>
<td>110</td>
</tr>
<tr>
<td>Recorded Excerpts</td>
<td>112</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS – Continued

Drone ................................................................. 114
Loops ................................................................. 115
7.4. SUMMARY ....................................................... 117

CHAPTER 8. ANALYSIS OF TECHNOLOGY COMPONENTS IN CORT LIPPE’S MUSIC FOR CLARINET AND ISPW (1992) ................. 118
8.1. HISTORICAL BACKGROUND .................................. 118
8.2. MUSICAL ROLE OF TECHNOLOGY ......................... 120
8.3. ANALYSIS OF TECHNOLOGY COMPONENTS ............... 121
  8.3.1. Sound System and Necessary Hardware .................. 122
  8.3.2. Sound sources ............................................. 123
    Microphone Input ............................................... 124
    Pre-recorded Samples ......................................... 124
  8.3.3. Control Sources ........................................... 127
    Event List ....................................................... 128
    Pitch Tracking .................................................. 130
    Envelope Following ............................................ 130
    Automated Processes .......................................... 130
    Graphical User Interface (GUI) ............................... 131
    Automated Score Following ................................... 131
  8.3.4. Synthesis and Signal Processing ......................... 132
    Sampler ......................................................... 133
    Granular Sampling ............................................. 134
    Harmonizer ...................................................... 137
    Reverb .......................................................... 139
    Noise Modulation .............................................. 139
    Frequency shifter .............................................. 140
    Flange .......................................................... 142
    Frequency/Amplitude Modulation ............................. 143
    Signal Routing ................................................. 144
    Spatializer ...................................................... 145
<table>
<thead>
<tr>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4. SUMMARY</td>
<td>146</td>
</tr>
<tr>
<td>CHAPTER 9. PERFORMANCE REALIZATION OF THEA MUSGRAVE’S NARCISSUS</td>
<td>148</td>
</tr>
<tr>
<td>9.1. EQUIPMENT AND STAGE SETUP</td>
<td>149</td>
</tr>
<tr>
<td>9.1.1. Input</td>
<td>150</td>
</tr>
<tr>
<td>Microphones</td>
<td>150</td>
</tr>
<tr>
<td>Computer Sound Input</td>
<td>150</td>
</tr>
<tr>
<td>MIDI Footswitch Controller</td>
<td>151</td>
</tr>
<tr>
<td>9.1.2. Output</td>
<td>151</td>
</tr>
<tr>
<td>9.2. DIGITAL DELAY SYSTEM SOFTWARE</td>
<td>151</td>
</tr>
<tr>
<td>9.2.1. Multi-tap Delay Line</td>
<td>153</td>
</tr>
<tr>
<td>9.2.2. Delay Time</td>
<td>153</td>
</tr>
<tr>
<td>9.2.3. Feedback</td>
<td>154</td>
</tr>
<tr>
<td>9.2.4. Modulation</td>
<td>155</td>
</tr>
<tr>
<td>9.2.5. Hold</td>
<td>156</td>
</tr>
<tr>
<td>9.2.6. Volume</td>
<td>157</td>
</tr>
<tr>
<td>9.2.7. Bypass</td>
<td>159</td>
</tr>
<tr>
<td>9.3. CONTROL SYSTEM</td>
<td>160</td>
</tr>
<tr>
<td>9.3.1. MIDI Input Processing</td>
<td>160</td>
</tr>
<tr>
<td>9.3.2. Score Event Processing</td>
<td>161</td>
</tr>
<tr>
<td>9.3.3. Linkage of Control and Processing Modules</td>
<td>163</td>
</tr>
<tr>
<td>9.4. SUMMARY</td>
<td>163</td>
</tr>
<tr>
<td>CHAPTER 10. PERFORMANCE REALIZATION OF BRUCE PENNYCOOK’S PRAESCIO IV</td>
<td>165</td>
</tr>
<tr>
<td>10.1. EQUIPMENT AND STAGE SETUP</td>
<td>165</td>
</tr>
<tr>
<td>10.1.1. Input Devices</td>
<td>166</td>
</tr>
<tr>
<td>Foot Controls</td>
<td>166</td>
</tr>
<tr>
<td>Pitch Tracker</td>
<td>167</td>
</tr>
<tr>
<td>10.1.2. Output Devices</td>
<td>168</td>
</tr>
<tr>
<td>10.1.3. Overview of Stage Setup</td>
<td>169</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS – Continued

10.2. INTERACTIVE MIDI SYSTEM SOFTWARE ........................................ 170
  10.2.1. Prepared Data ................................................................. 171
    Event Lists ................................................................. 171
    Standard MIDI Files (SMFs) ........................................... 174
  10.2.2. Event Processing ......................................................... 174
    Event List Control ....................................................... 175
    Play Events .............................................................. 176
    THRU Events ............................................................ 176
    MIDI Output ............................................................ 177
10.3. SYNTHESIZER ................................................................. 178
10.4. SUMMARY ................................................................. 179

CHAPTER 11. PROJECT SUMMARY .................................................. 181

CHAPTER 12. FUTURE DIRECTIONS FOR THIS RESEARCH .................. 185
  12.1. REAL-WORLD TESTING ..................................................... 185
  12.2. LIMITATIONS OF THIS MODEL .......................................... 186
  12.3. NEXT STEPS .............................................................. 189

APPENDIX A. SELECTED LIST OF WORKS FOR CLARINET AND
INTERACTIVE ELECTRONICS .................................................... 190

APPENDIX B. EVENT LIST FOR BRUCE PENNYCOOK’S PRAESCIO IV. . 193

APPENDIX C. BLOCK DIAGRAMS FOR SIGNAL PROCESSING
MODULES AND ADDITIONAL ALGORITHMIC PROCESSING USED
IN CORT LIPPE’S MUSIC FOR CLARINET AND ISPW ............ 196

APPENDIX D. KEY TO SYSTEM VARIABLES USED IN CORT LIPPE’S
MUSIC FOR CLARINET AND ISPW .......................................... 219

REFERENCES ........................................................................... 223
LIST OF FIGURES

CHAPTER 2. OVERVIEW OF INTERACTIVE ELECTROACOUSTIC MUSIC AND PERFORMANCE PRACTICE: COMMON MATERIALS AND TERMINOLOGY

Figure 2.1. Simple mathematics implemented with Max objects ........................................... 42
Figure 2.2. A simple signal processing function implemented with Max/MSP objects ........................................ 42

CHAPTER 5. ANALYSIS OF TECHNOLOGY COMPONENTS IN THEA MUSGRAVE’S NARCISSUS (1987), FOR B-FLAT CLARINET AND DIGITAL DELAY

Figure 5.1. Audio system setup ...................................................................................... 61
Figure 5.2. Score example: delay time .............................................................................. 62
Figure 5.3. Score example: delay feedback ........................................................................ 63
Figure 5.4. Score example: delay time modulation .............................................................. 67
Figure 5.5. Score example: delay hold ............................................................................... 70
Figure 5.6. Score example: delay volume ........................................................................... 70
Figure 5.7. Score example: delay bypass .......................................................................... 71
Figure 5.8. The complete digital delay system .................................................................... 72
Figure 5.9. Vesta Koza DIG-411 front panel ..................................................................... 73

CHAPTER 6. ANALYSIS OF TECHNOLOGY COMPONENTS IN BRUCE PENNYCOOK’S PRAESCI0 IV (1990) FOR CLARINET AND INTERACTIVE MIDI SYSTEM

Figure 6.1. Score example: event 2 triggered by footswitch control ....................................... 81
Figure 6.2. Score example: sequence 3a as shown in the score ........................................... 88
Figure 6.3. Score example: notation of events 35 – 37 ......................................................... 91
Figure 6.4. Score example: parallel tracking of the clarinet by the synthesizer (event 12) ........................................................................ 92
Figure 6.5. Score example: chordal harmonization of the clarinet pitch ............................... 93
Figure 6.6. Diagram of the Praescio IV interactive MIDI system .......................................... 94

CHAPTER 7. ANALYSIS OF TECHNOLOGY COMPONENTS IN JONATHAN KRAMER’S RENASCENCE (1974) FOR CLARINET, TAPE, AND TAPE DELAY

Figure 7.1. Score example: initial material recorded into the delay line ............................... 98
LIST OF FIGURES – Continued

Figure 7.2. Score example: a new fragment layered with the first as it returns 34 measures later .................................................. 98
Figure 7.3. Score example: continuous eighth-note texture built out of layered fragments ......................................................... 98
Figure 7.4. Score example: layered chords and glissandi built up over multiple delay cycles ..................................................... 99
Figure 7.5. Tape delay system for the 1974 version of Renascence .......... 104
Figure 7.6. Matrix mixer for the 1974 version of Renascence .............. 107
Figure 7.7. Score example: complex changes to matrix settings .......... 109
Figure 7.8. Score example: matrix used as a simple delay bypass .......... 110
Figure 7.9. Excerpts used to construct the prerecorded tape ............... 113
Figure 7.10. Drone for the prerecorded tape .................................. 114
Figure 7.11. Score example: measure 920, loops end in phase, cueing the clarinet entrance ...................................................... 116

CHAPTER 8. ANALYSIS OF TECHNOLOGY COMPONENTS IN CORT LIPPE’S MUSIC FOR CLARINET AND ISPW (1992)

Figure 8.1. Minimal sound system and control hardware .................. 123
Figure 8.2. Score example: sample 1 (section I, events 5 – 7) ............... 124
Figure 8.3. Score example: sample 2 (section I, events 8 – 10) .......... 125
Figure 8.4. Score example: sample 3 (section I, events 11 – 13) .......... 125
Figure 8.5. Score example: sample 4 (section I, events 14 – 16) .......... 125
Figure 8.6. Score example: sample 5 (section II, event 8) ................. 126
Figure 8.7. Score example: sample 6 (section I, events 3 – 4) .......... 126
Figure 8.8. Score example: sample 7 (section I, events 9 – 10) .......... 126
Figure 8.9. Score example: sample 8 (section V, events 3 – 7) ........ 127
Figure 8.10. Event list excerpt: section III, event 14 ....................... 128
Figure 8.11. Algorithmic control of granular sampling ...................... 137
Figure 8.12. Algorithmic control of harmonizer and frequency shifter .... 141
Figure 8.13. Score example: algorithmic control of frequency shifter and harmonizer ......................................................... 142
CHAPTER 9. PERFORMANCE REALIZATION OF MUSGRAVE'S
NARCISSUS

Figure 9.1. Stage setup for a new realization of Narcissus .................. 149
Figure 9.2. Software implementation of the digital delay system ............ 152
Figure 9.3. Multi-tap delay line ........................................ 153
Figure 9.4. Delay time control module ................................... 154
Figure 9.5. Delay feedback control module ............................... 155
Figure 9.6. Modulation control module ................................... 156
Figure 9.7. Delay hold control module ................................... 157
Figure 9.8. Score example: application of the volume pedal ............... 158
Figure 9.9. Volume control module ....................................... 158
Figure 9.10. Bypass control module ...................................... 159
Figure 9.11. MIDI input controls ......................................... 160
Figure 9.12. Score event processing module ................................ 161
Figure 9.13. Score example: event 9, with hold engaged .................... 162
Figure 9.14. Data routing between control and signal processing subprograms.. 163

CHAPTER 10. PERFORMANCE REALIZATION OF PENNYCOOK’S
PRAESCIIO IV

Figure 10.1. DigiTech RP-20 effects processor and MIDI foot controller .. 167
Figure 10.2. Software implementation of the pitch tracker ................... 168
Figure 10.3. Complete stage setup for Praescio IV ......................... 170
Figure 10.4. Interactive MIDI system software user interface .............. 171
Figure 10.5. Event List control module .................................... 175
Figure 10.6. Play Event control module ................................... 176
Figure 10.7. THRU Event control module .................................. 177
Figure 10.8. MIDI output module ......................................... 178

APPENDIX C. BLOCK DIAGRAMS FOR SIGNAL PROCESSING MODULES
AND ADDITIONAL ALGORITHMIC PROCESSING USED IN CORT
LIPPE'S MUSIC FOR CLARINET AND ISPW

Figure C.1. Basic symbols used in the block diagrams of Cort Lippe's signal-
processing software instruments ........................................... 196
LIST OF FIGURES – Continued

Figure C.2. A simple Max/MSP instrument for additive synthesis and its corresponding block diagram ................................. 197

Figure C.3. A Max/MSP instrument for amplitude modulation (AM synthesis) and its corresponding block diagram .................. 197

Figure C.4. A phase-driven wavetable oscillator in Max/MSP and its corresponding block diagram ........................................... 198

Figure C.5. Samplers 1 and 2: playback is controlled by formulating MIDI-style note packets of variable parameters for pitch, velocity, duration, envelope, and glissando .................................................. 199

Figure C.6. Sampler Voice Module: each of the 16 sampler voice modules can read from any of the buffered sample files. Input variables control pitch, amplitude, file onset, envelope attack and decay rates, and glissando (range and duration) ................................................................. 200

Figure C.7. “Trevor-back.” Backwards granular sampling initiated by event list variable \texttt{b-start} ............................................................... 201

Figure C.8. “Trevor-back.” Backwards granular sampling initiated by event list variable \texttt{b-start} ............................................................... 201

Figure C.9. “PLAY_RAND.” Granular sampling controlled by random processes ........................................................................ 202

Figure C.10. Sampler control algorithm 1: Section I, events 5 – 6 and 9 ................................................................. 203

Figure C.11. Sampler control algorithm 2: section I, events 5 – 10 ............................................................................................ 203

Figure C.12. Sampler control algorithm 3: “Trevor” in section I, events 12 – 16 (also shown in figure 8.11) ................................ 204

Figure C.13. Sampler control algorithm 4: Section II, events 11 and 26 .............................................................................. 204

Figure C.14. Sampler control algorithm 5: Section III, event 23 .............................................................................................. 205

Figure C.15. Sampler control algorithm 6: Section III, event 26 .............................................................................................. 205

Figure C.16. Sampler control algorithm 7: send ramping pitch values to PLAY_RAND in Section IV, event 3 ......................... 206

Figure C.17. Sampler control algorithm 8: play Sampler1 in response to pitch tracker values in Section V, events 3 – 6 ................. 206

Figure C.18. Two delay lines with delay time increased at a constant linear rate. Delays are synchronized by osc1, but delay time increments are out of phase by 180 degrees. Amplitude window (“wind” oscillators) controls envelope for delay output, cross-fading between the two delay lines 207

Figure C.19. Frequency shifter: all harmonic components are shifted individually up or down by a fixed frequency interval ........ 208
LIST OF FIGURES – Continued

| Figure C.20. | A typical flange effect created by a short delay with LFO modulation | 209 |
| Figure C.21. | Combination of FM and AM synthesis based on real-time analysis of clarinet pitch and amplitude | 210 |
| Figure C.22. | Harmonizer/Frequency Shifter control: Section I, event 18 and Section II, event 1 | 211 |
| Figure C.23. | Reverb: the input signal is subjected to a series of delays with variable feedback | 212 |
| Figure C.24. | Noise Modulation: random amplitude envelopes generated by multiple noise wavetable LFOs | 213 |
| Figure C.25. | Spatializer input crossbar: inputs from all DSP modules are mixed individually for left and right channel output | 214 |
| Figure C.26. | Spatializer output: signal from the input crossbar is scaled and sent to the Digital to Analog Converters (DAC) for amplification via loudspeakers | 215 |
| Figure C.27. | Spatializer control algorithm: section I, events 12-16. Hard-left or hard-right placement of the computer-generated sound is chosen randomly each time a note between 50 and 62 is played and is detected by the pitch tracker | 216 |
| Figure C.28. | Spatializer control algorithm: random left-right placement of DSP modules’ output in section II, event 19 through two seconds after the onset of event 20 | 216 |
| Figure C.29. | Algorithmic control of spatializer using event list variable spaton | 217 |
| Figure C.30. | Amplitude tables used by spaton algorithm | 218 |
### LIST OF TABLES

**CHAPTER 2. OVERVIEW OF INTERACTIVE ELECTROACOUSTIC MUSIC PERFORMANCE PRACTICE: TOOLS, TECHNIQUES, AND TERMINOLOGY**

Table 2.1. Basic structure and interpretation of common MIDI messages ........................... 35

**CHAPTER 5. ANALYSIS OF INTERACTIVE TECHNOLOGY IN THEA MUSGRAVE’S NARCISSUS (1987), FOR CLARINET IN B-FLAT AND DIGITAL DELAY**

Table 5.1. Delay effect parameters: notated and actual values ............................ 72
Table 5.2. Pre-programmable effects changes ...................................................... 74

**CHAPTER 6. ANALYSIS OF TECHNOLOGY COMPONENTS IN BRUCE PENNYCOOK’S PRAESCIO IV (1990) FOR CLARINET AND INTERACTIVE MIDI SYSTEM**

Table 6.1. Suggested sound set for Praescio IV .............................................. 85
Table 6.2. Praescio IV event list sample .......................................................... 86
Table 6.3. Sequence 3a in MIDI event list format .............................................. 87
Table 6.4. Event list excerpt: events 35-38 ....................................................... 90

**CHAPTER 7. ANALYSIS OF TECHNOLOGY COMPONENTS IN JONATHAN KRAMER’S RENASCENCE (1974) FOR CLARINET, TAPE, AND TAPE DELAY**

Table 7.1. Transposition of long tones ............................................................. 114

**CHAPTER 8. ANALYSIS OF TECHNOLOGY COMPONENTS IN CORT LIPPE’S MUSIC FOR CLARINET AND ISPW (1992)**

Table 8.1. Variable parameters for sampler playback ........................................ 134

**CHAPTER 9. PERFORMANCE REALIZATION OF THEA MUSGRAVE’S NARCISSUS**

Table 9.1. Contents of coll file “Narcissus_events” ........................................... 162

**CHAPTER 10. PERFORMANCE REALIZATION OF BRUCE PENNYCOOK’S PRAESCIO IV**

Table 10.1. Excerpt from coll file 1: event control ........................................... 172
Table 10.2. Excerpt from coll file 2: play events .............................................. 173
LIST OF TABLES – Continued

Table 10.3. Excerpt from *coll* file 3: THRU events ......................... 174
Table 10.1. Proteus 2000 sound set for *Praescio IV* ......................... 179
ABSTRACT

Performers interested in presenting interactive electroacoustic works face serious obstacles when the required equipment or technology becomes obsolete or unavailable. Transcription to updated technology provides at best a temporary solution. Detailed and device-independent documentation of interactive electronic systems used in older works can guide new performance realizations using available equipment. Such documentation should itself be formatted in a way that does not depend on specific electronic devices for interpretation or retrieval. Therefore, this paper proposes a model for the documentation and preservation of interactive electroacoustic music systems in which all synthesis and audio signal processing algorithms, control functions, and human-machine interactions are described in machine-neutral terms, using a combination of text, mathematics, and schematic diagrams.

As an example of such documentation, the technical requirements for four works are analyzed and described: Thea Musgrave’s *Narcissus* (1987), for clarinet in B-flat and digital delay, Bruce Pennycook’s *Praescio IV* (1990) for clarinet and interactive MIDI system, Jonathan Kramer’s *Renascence* (1974) for clarinet, tape, and tape delay system, and Cort Lippe’s *Music for Clarinet and ISPW* (1992). New performance realizations of two of these works, Musgrave’s *Narcissus* and Pennycook’s *Praescio IV* are described and presented as part of the accompanying lecture-recital.
Interactive electroacoustic works, i.e. compositions combining live instrumental performance with flexible, performer-controlled electronic sound generation and processing systems, are often out of reach for performers because of their elaborate technological requirements. This barrier to performance is made more acute when specialized electronic hardware or software is required to perform a given piece, especially when the specified hardware or software is no longer available due to technological obsolescence. As Joel Chadabe has recently pointed out:

Traditional instrumental music can be preserved through notation, first because traditional compositions are defined by elements which can be notated, and, second, because traditional instruments are played in standard ways. Since electronic instruments are not played in standard ways, and further, since rapid changes in technology lead to a steady turnover of electronic instruments, notation can not serve as a way of preserving performances of electronic sounds. ... [E]lectronic performance can be preserved by describing the sounds themselves so that they can be performed on any appropriate instrument, by using current technology, or by updating the composition itself, and ... such approaches can be artistically viable if the performer understands the composer's intentions.¹

This study is an attempt to implement Chadabe's prescription for preserving electroacoustic works by analyzing and describing the sounds and functions of the electronics from a performance-practice perspective. Since standard musical notation is clearly inadequate for describing non-standardized musical uses of electronic technology, my goal is to thoroughly document the electronic sound generation and processing algorithms, control functions, and required performer-computer interactions that define

the electroacoustic components of four separate works. My technical analysis is intended as a blueprint for future performance implementations of the required interactive electroacoustic systems using whatever technology is available to the performer.

For each work considered in this project, I will provide a brief historical overview of the piece in terms of the circumstances of its commission and premiere performance, the original technology employed, and issues of diffusion and continued performance as related to availability of technological resources for performance.

Secondly, I will briefly describe the musical role played by electroacoustic technology in each work. This musical overview will concentrate on the composer’s motivation for combining live clarinet with interactive electroacoustic resources and the performance relationship between the live player and the electronic technology. This study is primarily concerned with the performance practice of interactive electroacoustic music rather than with the compositional techniques used in its creation. Therefore I will not attempt a general musical analysis, and I will focus instead on the practical issues performers face when attempting concert realizations of these works in the absence of the original technology used by the composer.

Thirdly, I will present a thorough analysis of the interactive electroacoustic music systems employed in each of the four works. Each system’s internal components and functions are described in general terms independent of any particular device or computer program. My intention is to present all the necessary synthesis, processing, and control algorithms, and to guide the interested performer or system engineer in realizing these works for performance in the absence of the composer’s original equipment.
Although each work is significantly different from the others in its use of electronics, there are some important commonalities. I have made every effort to standardize my analysis format in order to facilitate comparison between these works. I am hopeful that continuation of this type of research on a broader scale might some day lead to a general definition for a "common practice" in interactive electroacoustic composition and performance. Preliminary findings from the limited study presented here are encouraging in this regard.

Finally, I will present a brief overview and documentation of the software and equipment used in concert realizations of the two works presented at my lecture-recital. This will include a description of the electronic hardware and setup configuration used on stage as well as an explanation of the main components of new software implementations of the interactive systems required for performance of Musgrave's *Narcissus* and Pennycook's *Praescio IV*. 
CHAPTER 2
OVERVIEW OF INTERACTIVE ELECTROACOUSTIC MUSIC PERFORMANCE PRACTICE: TOOLS, TECHNIQUES, AND TERMINOLOGY

The following is an overview of the tools and techniques of interactive electroacoustic music. This is not intended as a comprehensive investigation of the history of electroacoustic music or of current computer music performance practice. Rather, I hope to define some terms and key concepts in order to provide general background and context for the analysis and reconstruction of interactive electroacoustic works as presented elsewhere in this document.

2.1 WHAT IS INTERACTIVE ELECTROACOUSTIC MUSIC?

Robert Rowe defines interactive computer music systems as "those whose behavior changes in response to musical input. Such responsiveness allows these systems to participate in live performances, of both notated and improvised music." Therefore, interactive electroacoustic music is distinguished from electroacoustic music rendered on tape or other fixed media because it may be modified or controlled in real time by the actions of a performer or by other variable elements within the performance environment. The term "real time" is used throughout this paper to describe changes to electronic processes or synthesis algorithms that take effect immediately, altering the sound as it is heard, rather than requiring complex rendering or compilation before auditioning the results.

---

An interactive electroacoustic music system may be controlled by a technician, or it may take its cues and control signals directly from other instruments. An interactive system may be assembled from general-purpose electronics, or it may be based on the capabilities of a specialized computer music workstation. Some interactive music systems consist of a complex “mix-and-match” array of audio software and hardware, based on the preferences and circumstances of the composers and performers involved.

To date there is no single standard by which composers and performers approach the concept, much less the details, of interactivity in electroacoustic musical performance. Rather, performance practice in this area has evolved over several decades, and continues to evolve through a process of inventing solutions as needed on a case-by-case basis. Much of the focus in the computer music literature has been devoted to compositional processes and the development of new technologies. It is possible that performance issues will gain prominence in the computer music debate if enough energy is devoted to the development of standardized systems for realizing and performing the existing interactive electroacoustic literature. This project is intended as a step in that direction.

2.2 HISTORICAL FOUNDATIONS OF INTERACTIVE ELECTROACOUSTIC MUSIC

Before 1950, the history of electroacoustic music mainly involves the introduction of new electronic instruments. The “Telharmonium,” patented in 1897 and first produced commercially in 1906, was an additive synthesis engine designed for live musical
performance distributed over telephone lines to an audience of subscribers.\(^3\) Other notable electronic instruments of the first half of the twentieth century include the Theremin (1920s), Ondes Martenot (1928), Trautonium (1930), and Hammond Organ (1935). Such self-contained instruments can be played alone or included in an ensemble. They function much like any other instrument in the orchestra, albeit as members of a new organological family known as “electrophones.”

Beginning around 1948, composers and engineers working at studios in Paris and Cologne, and later in the United States, created electronic music by means of magnetic tape manipulation (cutting, splicing, overdubbing, etc). By the late 1950s and early 1960s, a new electronic music performance tradition emerged, based on the techniques and equipment of the tape music studio. Live electronic music followed two main (though not mutually exclusive) paths, both of which continue to the present. The first is the combination of traditional acoustic instruments with electronic sounds on tape. The second incorporates electronic equipment directly into live performance (“performed electronics”) in order to generate and manipulate electronic sound in “real time” (as opposed to being rendered on tape through painstaking studio processes). Often, such techniques are used to supplement, modify, and extend the sound of live instruments. Well-known early works of the first type include Mario Davidovsky’s series of *Synchronisms* for various instruments and tape, as well as works by Jacob Druckman and Milton Babbitt. Early efforts in the field of performed electronics include the work of

Gordon Mumma and the ONCE group, John Cage and David Tudor at Mills College, and Karlheinz Stockhausen in such works as *Solo* and *Mikrophonie I*.

### 2.3 GENERAL PURPOSE ELECTRONICS AS MUSICAL INSTRUMENTS

Many of the early works of live electroacoustic music, especially from the 1960s, used general-purpose audio equipment in creative musical ways. Such equipment was often obtained from radio stations or electronic tape music studios, and included such items as tape recorders, microphones, tone generators, filters, and other sound modulating devices.

#### 2.3.1 Microphone Feedback Processing

Microphones have been used most commonly in performance for the simple amplification of acoustic instruments and vocals or as an input device for tape recorders. However, a number of composers, beginning in the early 1960s, began to find creative musical uses for microphones as a live performance instrument in its own right. Stockhausen's *Mikrophonie I* (1964) uses microphone feedback and a bank of filters to transform the sound of a tam-tam. All of the electronic equipment is controlled by a group of six performers who manipulate microphone placements and filter/amplifier settings. Throughout the 1960s, contact microphones used with loudspeakers to generate feedback were a regular part of the improvisational performances of live electronic groups such as Musica Electronica Viva, AMM, Gentle Fire, Naked Software, and John

---

Cage's group at Stony Point, NY.\(^5\) Percussionist Max Neuhaus' 1964-65 realizations of works by Cage, Earle Brown, and Sylvano Bussotti demonstrated creative use of feedback as an expressive timbral element, using contact microphones and loudspeakers to manipulate various percussion instruments.\(^6\)

2.3.2 Tape Recorders

The earliest known work to use a tape recorder as a live performance instrument is Mauricio Kagel's *Transición II* (1958-9), for piano, percussion, and two tape recorders. In this work, recorded segments of the performance are looped and played back before the conclusion of the piece.\(^7\) Stockhausen's classic work *Solo* (1966) is written for any melody instrument (traditionally flute or trombone) and tape delay system. *Solo* features a long tape delay with six movable playback heads placed at different intervals along the tape path, controlled in performance by four technical assistants.\(^8\) Jonathan Kramer's *Renascence* (1974), discussed in detail in chapter 7 of this paper, provides another example of a live tape delay as the basis of an interactive performance system.

2.3.3 Tone Generators, Ring Modulators, Filters, and Other Equipment

Ring modulators (a device that applies spectral transformations by combining two separate inputs), sine wave generators, and audio filters were favored by electronic music

\(^7\) Ibid., 153.
composers working at the Cologne studios in the 1960s. Stockhausen's *Mixtur* (1964/67) features four ring modulators controlled by sine tone generators. The electronic equipment in this case modifies the sound of instrumental groups in a chamber orchestra through a process of additive synthesis, augmenting and complicating the spectrum of traditional instrumental sounds. This is in contrast to Stockhausen's earlier use of subtractive synthesis processes in *Mikrophonie I*, in which the harmonically rich sound of a tam-tam is selectively focused and reduced using band-pass filters. Other elaborate uses of ring modulators appear in concert works by Roger Reynolds (*Traces*, 1969) and Larry Austin (*Accidents*, 1967, for prepared piano and ring modulator). Stockhausen's *Mantra* (1970) combines two pianos with shortwave radios, ring modulators, and multi-speaker sound projection. Alvin Lucier’s *North American Time Capsule* (1967), commissioned by Sylvania Electronic Systems, applied vocoder effects to a vocal chorus as a demonstration of this new audio signal processing technique designed for telephony.

2.4 SYNTHESIZERS AND AUDIO SIGNAL PROCESSING INSTRUMENTS

Instruments that generated sound electronically were first introduced as early as 1906 and enjoyed great popularity in the 1920s and 30s. The invention of the RCA Mark II, installed at the Columbia-Princeton Electronic Music Center in 1959, allowed a handful of composers to develop works based on the capabilities of a fully controllable

---

music synthesizer.\textsuperscript{12} Widespread use of synthesizers among composers, especially as a part of the emerging live electroacoustic performance tradition, came with the invention of the transistor and the introduction of modular voltage-controlled synthesis in the early 1960s.

Since that time, synthesizers and sound processing equipment ("effects processors") have become ubiquitous in nearly all spheres of musical activity. Typically, a synthesizer is a self-contained device for the electronic generation of sound. It may have a keyboard interface or it may be simply a box, or "sound module," controlled by other devices within a network. Increasingly, "virtual" synthesizers and effects processors have become available. Use of software-based instruments neatly avoids the need for external hardware or associated interfaces and cables.

2.4.1 Synthesizers

A synthesizer is broadly defined as a musical instrument that generates sound by entirely electronic means. Early analog synthesizers generated individual harmonic components of a sound using banks of individually controlled oscillators, a technique known as "additive synthesis." A corollary technique is "subtractive synthesis" in which individual harmonic components or spectral regions are selectively filtered out of a complex timbre. The introduction of the modular synthesizer in the mid-1960s, in which oscillators and control voltage sources could be combined in any order to create complex

\textsuperscript{12} Manning, \textit{Electronic and Computer Music}, 83.
user-definable timbres, set the tone (so to speak) for nearly all subsequent development of electronic music.

Direct digital synthesis, in which a stream of audio data is generated according to a particular synthesis algorithm, is the basis for most synthesizers manufactured since the 1980s. New techniques for direct digital synthesis have flourished in this time period. Common digital synthesis techniques include emulations of analog additive and subtractive techniques as well as frequency modulation (FM synthesis), amplitude modulation (AM), wave-table and wave-shaping synthesis, physical modeling, spectral analysis/resynthesis techniques, and granular synthesis to name only a few. Many excellent resources on the topic of digital synthesis techniques and their application to musical composition exist. For detailed discussions of these and other synthesis techniques, I refer the reader to books on the subject by Eduardo Reck Miranda\textsuperscript{13} or Max Matthews and John R. Pierce.\textsuperscript{14}

Early uses of modular synthesizers in concert works began in the 1960s. The first reported such instance was for John Eaton's *Songs for RPB* (soprano, piano and synket) in 1965.\textsuperscript{15} Throughout the 1960s and 70s, synthesizers became a mainstay of popular and progressive rock groups, and the commercial development of synthesizers for performance applications was therefore closely tied to this market.


\textsuperscript{15} Manning, *Electronic and Computer Music*, 163.
2.4.2 Samplers

A sampler is a type of synthesizer that uses prerecorded sounds ("samples") as the basis of tone production. Typically, sound samples are assigned to specific notes or ranges of notes, and may be transposed according to the tempered scale. In addition to transposition, samples may be layered or processed (including looping, amplitude scaling/enveloping, filtering, etc.) to achieve sonic complexity beyond mere imitation of the original sound source.

Digital samplers have been available since 1979, with the introduction of the Fairlight CMI, followed in 1981 by the E-Mu Emulator series.\(^\text{16}\) Over the last twenty years, samplers have become a standard part of the electroacoustic palette, and the once sharp distinction between sampler and synthesizer has become somewhat blurred. Many contemporary sound modules offer a combination of sampling and direct digital synthesis.

One advantage of digital samplers is the portability of sounds from one device to another using standard digital file formats and transfer protocols. C. Matthew Burtner’s 1996 interactive multimedia work *Taruyamaarutet (Twisted Faces in Wood)*, uses a set of sampled wood percussion sounds prepared by the composer from his own field recordings. For the premiere performance (in which I participated), the instrument used was an Akai S1000. For a subsequent performance in 1999, I transferred Burtner’s sound set and key map to a Kurzweil K2000 sampling synthesizer. As long as the set of individual source sound files and their associated key assignments are available, this

work can be recreated essentially unchanged using whatever sampling system is available.

2.4.3 Effects Processors

Signal processing equipment designed specifically for musical applications are commonly used to transform sounds arriving at their inputs. The effects may be subtle, as with the application of mild reverberation, or they may be radical, as in the use of intervallic pitch shifting and distortion. Effects processors are commonly available as stand-alone devices that can be incorporated into a studio network. They may also be integrated into the architecture of a synthesizer or implemented in software running on a general-purpose computer.

Effects processors tend to draw from a standard repertoire of audio effects. The most common of these are time-based effects, such as reverberation, delay, and flanging, or amplitude effects, such as ring modulation, dynamic expansion, or compression. Most effects processors can combine individual effects in series to achieve complex results.

The use of effects processing in concert works has its roots in the application of filters and ring modulators to acoustic instruments by Stockhausen and others in the 1960s (discussed in section 2.3.3, above). With the advent of inexpensive and flexible digital effects processors in the 1980s, composers began to seriously explore the musical possibilities for live transformation of acoustic sounds offered by these devices. Examples from the literature for clarinet and electronic effects include Thea Musgrave’s

2.5 THE MIDI STANDARD

MIDI (Musical Instrument Digital Interface) was created between 1981 and 1983 as a collaborative effort between commercial manufacturers of synthesizers and musical equipment. The MIDI 1.0 Specification, completed in 1983, described an open standard for the intercommunication of electronic musical instruments, regardless of manufacturer or internal synthesis architecture. Most synthesizers introduced after 1984 have been compatible with this specification, and MIDI has remained essentially unchanged as a communication standard for over 20 years.

The basic concept of MIDI is the transmission of discrete data messages from one device to another over a standard cable. MIDI does not describe actual sounds or timbres, but instead sends basic instructions to which a sound generator may respond. These instructions are high-level abstractions of musical gestures, such as the striking of a key or the movement of a control slider or pedal. MIDI has become an essential link in almost all stage or studio electronic music environments, since it is well suited to centralized control of large networks of devices. Because MIDI messages are constructed digitally, they can be stored and manipulated in a variety of ways.

---

17 Pellman, An Introduction to the Creation of Electroacoustic Music, 132.
2.5.1 MIDI Message Types

MIDI messages are represented as 8-bit binary data packets (or “bytes”), and fall into two main categories: status bytes and data bytes. A status byte defines a MIDI message as one of eight available types, as well as its transmission channel (out of 16 possible). A data byte following a status byte defines its parameter values. The affected parameter depends on the last status byte received.

MIDI was designed primarily for use with keyboard synthesizers, and this fact is reflected in the structure of the protocol. The most common type of MIDI message is the Note On message. A Note On status byte is followed by two data bytes, defining a note number (corresponding to a key on a synthesizer), and a velocity value (a measurement of the attack velocity of the depressed key). The effect of a Note On message is typically (and not surprisingly) to initiate a tone with the specified pitch at an attack amplitude corresponding to the given velocity value. Other possible MIDI message types include Note Off, Program Change (used to change synthesizer presets), Aftertouch (a control message tied to physical pressure on the keys), Polyphonic Aftertouch (separate data streams generated for pressure on individual keys), Pitch Bend (usually controlled by hand using a pitch wheel or joystick), Control Change (definable for use with external sliders, foot pedals, or other control devices), and System Exclusive messages (allowing for detailed remote control of synthesizer settings using device-specific control codes). The structure of typical MIDI messages is shown as a sequence in table 2.1.
Table 2.1. Basic structure and interpretation of common MIDI messages

<table>
<thead>
<tr>
<th>Status Byte</th>
<th>Data Byte 1</th>
<th>Data Byte 2</th>
<th>Interpreted as ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>10011000 (152)</td>
<td>00111100 (60)</td>
<td>01000000 (64)</td>
<td>Note On, channel 8, Middle C, \textit{mezzo forte}</td>
</tr>
<tr>
<td>10111000 (184)</td>
<td>01000000 (64)</td>
<td>01111111 (127)</td>
<td>Control Change on channel 8, Sustain Pedal, On</td>
</tr>
<tr>
<td>11001000 (200)</td>
<td>01111111 (127)</td>
<td>--</td>
<td>Change the synthesizer voice on channel 8 to preset 127</td>
</tr>
<tr>
<td>10111000 (184)</td>
<td>01000000 (64)</td>
<td>00000000 (0)</td>
<td>Control Change on channel 8, Sustain Pedal, Off</td>
</tr>
<tr>
<td>10011000 (152)</td>
<td>00111100 (60)</td>
<td>00000000 (0)</td>
<td>Note On, channel 8, Middle C, \textit{silent} (effectively a &quot;Note Off&quot; message)</td>
</tr>
</tbody>
</table>

2.5.2 Extensions to MIDI

Extensions to the MIDI standard include the Standard MIDI File (SMF) format, which defines a common strategy for storing recorded MIDI messages and associated timing data for playback, and the General MIDI (GM) standard, which defines a set of standard voice types assigned to specific program numbers. These last two extensions have led to the proliferation of commercially prepared MIDI files, often available via Internet, and the use of MIDI to control soundtracks for multimedia applications and video games. MIDI, including its extensions, has proved extremely useful to composers of concert music, despite its limitations. Bruce Pennycook’s creative use of MIDI controllers, sequences, and synthesizers (discussed in the context of his work \textit{Praescio IV}, in chapters 6 and 10, below), provides evidence of the flexibility and usefulness of MIDI as a control system for complex musical situations.
2.6 EXPERIMENTAL INTERACTIVE SOFTWARE SYSTEMS

In the late 1980s and early 1990s, composers and researchers working independently at various institutions and studios developed new interactive computer music systems that took advantage of MIDI software processing or used powerful new systems for computer-controlled digital signal processing. The common approach among many of these systems involved a general-purpose personal computer as a control system for an external array of sound generating and processing equipment. The advantages were that the control software could be relatively simple and portable from one machine to another, while the external audio hardware could execute complex synthesis and signal processing routines that were far beyond the capabilities of the personal computers of the day.

In many cases, such systems were developed by individual composers to accommodate the specific demands of their own works. Three such systems provide a representative example of the pioneering work carried out in this direction.

2.6.1 Cypher

Cypher is an interactive computer music system created by composer Robert Rowe, and is divided into two parts: a “listener” and a “player.” The listener analyzes an incoming stream of MIDI data, and the player generates musical material in response. Rowe avoids using stored sequences in his compositions with Cypher. Rather, the program is designed to respond to the input that it “hears” according to various
algorithmic processes. Therefore, Cypher is well suited for improvisational
performance in which the computer assumes the role of accompanying performer, rather
than a mere extension of a soloist or ensemble.

2.6.2 MIDI-Live

A team of researchers led by composer Bruce Pennycook at McGill University in
the late 1980s and early 90s created an interactive computer music system dubbed
"MIDI-Live." This system was designed specifically around the demands of
Pennycook's PRAESCIO series of compositions, which were motivated primarily by a
desire to add flexible performer control to pre-composed electronic sounds.

The MIDI-Live software system, running on a standard IBM PC, evolved with
each work in the PRAESCIO series. The main features of the system were harmonization
(or "colorization") of a stream of incoming MIDI notes, playback of stored MIDI
sequence data, MIDI continuous controller processing, and event triggering according to
stored lists of parameter values and system actions. Subsequent works in the series, as
well as a few of the older ones (including Praescio IV, discussed in chapters 6 and 10),
were later implemented using Max software on the Macintosh computer.

---

18 Rowe, Interactive Music Systems, 39.
19 Bruce Pennycook, “Machine Songs II: The PRAESCIO Series – Composition-
2.6.3 4X

The 4X machine was a powerful signal processing and synthesis workstation developed in the mid-1980s at IRCAM (Institut de Recherche et Coordination Acoustique/Musique), Paris. Its chief strength was its ability to apply complex audio signal processing effects to live sounds arriving at microphone inputs, and to respond to real-time control input from a computer keyboard or MIDI device. Composers working with the 4X included Pierre Boulez (*Repons* for clarinet), Philippe Manoury, and Robert Rowe. The expense of the 4X system kept it out of reach of many composers, and the proliferation of sophisticated MIDI devices and audio signal processing add-on boards for inexpensive personal computers led to its obsolescence by the late 1980s. However, the 4X was an important precursor to the IRCAM Signal Processing Workstation (ISPW), discussed in detail in chapter 8, below.

2.7 WIDELY-USED GENERAL-PURPOSE INTERACTIVE SOFTWARE SYSTEMS

The practice of live electroacoustic music over the last ten years has been primarily shaped by the proliferation of computer-based interactive music programming systems. These systems are flexible and open-ended, allowing composers and performers to design the specific tools they need for any given musical task. Most such systems essentially provide a “canvas” and a “palette” of tools (which may be extended if necessary). Within this framework, virtual machines for audio and music processing may

---

be constructed. Such virtual instruments may be entirely self-contained in software (with
signal processing functions taken on by the host computer’s main processor), but may
also incorporate and manage a network of external devices.

2.7.1 Max/MSP

Max/MSP is the programming vehicle of choice for a large number of active
composers and performers in the field of interactive electroacoustic music today. The
materials supplied by Bruce Pennycook and Cort Lippe for the analyses presented in
chapters 6 and 8 (respectively) came in the form of performance software applications
developed in Max/MSP. I have worked with Max/MSP extensively for the last eight
years, and it is therefore the tool I have chosen for my own realizations of Thea
Musgrave’s Narcissus and Pennycook’s Praescio IV. Examples from my own Max/MSP
performance software are given in chapters 9 and 10. Because of its flexibility as a
programming tool, and its stability in performance, I am in the process of developing my
entire repertoire of interactive electroacoustic works for clarinet as a series of Max/MSP
software applications that can run on the same machine. In fact, one major advantage of
performance software developed in Max/MSP is that it can be compiled as a “stand-
one” application that will run on any computer (so long as it meets minimum
requirements) without needing a full Max/MSP license or installation. In other words, it
is therefore possible to create performance applications in Max/MSP that could be
downloaded from the Internet and used by performers with little or no software
programming experience.
Max/MSP is an object oriented graphical programming environment for music and multimedia, distributed commercially by Cycling 74, founded by Max/MSP developer and co-creator David Zicarelli. Max/MSP is essentially two programs functioning together in a single environment. The first part of the program, Max (named for computer music pioneer Max Matthews), was developed by Miller S. Puckette at IRCAM and prepared for commercial release by Zicarelli for Opcode Systems in 1990. Max consists of a flexible system for user-definable control-data processing and MIDI communication. MSP ("Max Signal Processing") is an audio processing system superimposed on the Max environment that extends its capabilities into the realm of real-time direct digital synthesis and signal processing. Max/MSP was developed for the Apple Macintosh personal computer, but was recently released (February 2004) in a version for Microsoft Windows computers as well. Max/MSP enables composers and performers with limited traditional programming skills to create complex virtual instruments for audio signal processing and real-time interaction. All sound processing and control functions can take place within an integrated software environment running on a single computer, without needing additional hardware or intermediary layers of software.

Max/MSP is considered an "object oriented" programming environment. Robert Rowe describes it as follows:

Object orientation is a programming discipline that isolates computation in objects, self-contained processing units that communicate through passing messages. Receiving a message will invoke some method within an object. Methods are constituent processing elements, which are related to each other, and

---

isolated from other methods, by virtue of their encapsulation in a surrounding object.\textsuperscript{23}

Object orientation is implemented in Max/MSP graphically. Each object is represented by a rectangular box that can be placed anywhere on the screen. The object box typically features one or more “inlets” and one or more “outlets.” Multiple instances of the same object will function completely independently from one another. Messages are passed from one object to another by virtual “patch chords” that connect the message outlet of one object to the message inlet(s) of one or more other objects. Each object has a well-defined set of functions designed to manipulate certain data types and messages arriving at each inlet. A network of Max/MSP objects interconnected by patch chords is known as a “patcher.” With this system, virtually any algorithm can be implemented using standard objects. Max/MSP is designed to operate in real time. Input signals from the computer keyboard or external devices (including MIDI or audio signal sources) may trigger execution of an algorithm, producing an instant response. For the rare cases in which standard objects are insufficient to the task at hand, additional “external” objects may be written in the C programming language and added to the available palette.

Because of its graphic nature, programs created in Max/MSP can be relatively easy to follow, even for those who are not expert programmers. A simple Max patcher, shown in figure 2.1, below, demonstrates two alternate strategies for solving a problem. As with any programming language or system, solutions to a particular problem are often a matter of style, and multiple strategies can achieve the same result.

\textsuperscript{23} Rowe, \textit{Interactive Music Systems}, 26.
Two realizations of the same computational problem

**Summation of numbers between 1 and n (1 + 2 + 3 ... + n)**

**Strategy 1 - Count from 1 - n, adding each value to the previous one.**

- **initialize**: 
  - "bang" = trigger: M M(start at 1)
  - "counter" object:
    - "bang" message causes increment

- **loop while condition (count < n) is true; stop when count > n**

**Strategy 2 - Use a simple formula: sum = n(n+1)/2 to achieve the same result**

- \[ n = 10 \]
- \[ \text{add:} \quad | + 1 | n + 1 \]
- \[ \text{multiply:} \quad n(n + 1) \]
- \[ \text{divide:} \quad / 2 \]
- \[ \text{sum} = 55 \]

Figure 2.1. Simple mathematics implemented with Max objects

MSP objects add a signal-processing layer to the Max environment. Patch cords representing audio signal connections are represented by a striped line, and signal objects are named with a ' ~ ' character at the end. Figure 2.2 shows a basic signal network for applying amplitude modulation to a sound arriving at the computer’s audio input:

Figure 2.2. A simple signal processing function implemented with MSP objects
Relatively simple Max/MSP patchers like the ones shown in figures 2.1 and 2.2 (above) may be interconnected to create extremely complex processing networks for system control, audio signal processing, and real-time direct digital synthesis. Because of its graphical interface and real-time operation, Max/MSP has enormous potential not only for performers and composers, but also for applications in other fields such as music pedagogy and psychoacoustics.

2.7.2 Pd

Pd (Pure Data) is a freely available program that is closely related to Max/MSP. As described on the Pure Data users’ web site:

PD (aka Pure Data) is a real-time graphical programming environment for audio, video, and graphical processing. It is the third major branch of the family of patcher programming languages known as Max (Max/FTS, ISPW Max, Max/MSP, jMax, etc.) originally developed by Miller Puckette and company at IRCAM. The core of Pd is written and maintained by Miller Puckette and includes the work of many developers, making the whole package very much a community effort.

Pd was created to explore ideas of how to further refine the Max paradigm with the core ideas of allowing data to be treated in a more open-ended way and opening it up to applications outside of audio and MIDI, such as graphics and video.24

Pd follows essentially the same concept and structure as Max/MSP, and the two programs even share many objects in common. However, Pd is designed to run primarily on UNIX-based machines (Linux, IRIX, BSD, etc), though versions also exist for Windows and Mac OS X computers. Pd’s main advantages come from its open-source

distribution, allowing any interested programmer access to the source code for purposes of modification or extension.

2.7.3 SuperCollider

SuperCollider is a real-time object oriented sound synthesis programming language created by James McCartney and introduced in 1996. It was designed for the Apple Macintosh computing platform and is largely based on the programming language Smalltalk. A version has been created for Linux computers and some work has been done towards a Windows version as well.\(^\text{25}\)

SuperCollider synthesis instruments and compositional algorithms are defined in blocks of text code, using a specialized syntax. In addition, SuperCollider offers facilities for creating graphic user interfaces and control panels. It can communicate with external devices via MIDI and can process incoming audio in real time, making it a suitable tool for live performance applications.\(^\text{26}\)

2.7.4 Kyma

Kyma is a graphical programming environment for sound design, composition, and performance. It runs on either Windows or Macintosh computers, but depends on an external audio accelerator unit, the "Capybara," to handle digital signal processing


functions. In this way it is very similar to the IRCAM Signal Processing Workstation, which linked a version of Max running on a NeXT computer with a separate multiprocessor audio accelerator.\textsuperscript{27}

The host computer running Kyma is linked to the Capabara unit via FireWire (400 Mbps IEEE 1396) cable. The Capybara may have between 4 and 32 separate processors devoted entirely to audio signal processing. Like Max/MSP and Pd, Kyma allows the user to combine individual modules, perform arithmetic operations on control values, and build self-contained modules that play by themselves. Timeline features are also included, allowing for the creation of complex sequences.\textsuperscript{28}

Kyma was first developed in 1986 and has been continuously updated to the present. The original external sound accelerator used with the first version of Kyma was known as "Platypus," followed by the first Capybara in 1990.\textsuperscript{29} Kyma also includes an extensive library of sounds and sound design prototypes, offering perhaps an easier learning curve than programs such as Max/MSP, Pd, and SuperCollider.

2.8 \textbf{SUMMARY}

In just over forty years, interactive electroacoustic music has developed as a distinct discipline of computer music, with a wide range of tools and techniques and an


\textsuperscript{29} Ibid.
emerging performance practice. Changes in the field have been rapid and far-reaching, often moving faster than the ability of the musical community to keep pace with the tasks of developing and maintaining a viable literature.

Early experiments with sound generating and modifying equipment led to many new compositional forms and performance techniques. Commercial development of portable synthesizers and effects processors brought real-time electronics within reach of a wider pool of composers and performers. The MIDI standard made it possible to link large arrays of specialized equipment together in a network, functioning as a single, flexible interactive system. Most recently, inexpensive but powerful personal computer systems can now incorporate all the functions of a complex interactive system within a single, user-definable software environment.
CHAPTER 3
INTERACTIVE ELECTROACOUSTIC MUSIC AND THE PROBLEM OF TECHNOLOGICAL OBSOLESCENCE

In 1991, Karlheinz Stockhausen, speaking in the context of a recent performance of his 1964 work *Mikrophonie I* and the difficulties associated with digitally reconstructing the specially designed filters used in the original performance, seems to make a case for the preservation of electroacoustic works only in their original state:

> It is extremely important to comprehend works, which were born to a particular historical moment, for their uniqueness. It just won't do to be continually discarding everything and making something different, but rather we should be preserving things and adding new ones. Anyway, it is my experience of music that every instrument, every item of equipment, every technique can produce something unique, which can be achieved in no other way. Since that is the case, then we can speak of an original technique, and thus deal with an original instrument. If it is imitable, then it is also not worth much.\(^\text{30}\)

Stockhausen's position poses a problem for the performer who would like to play an existing electroacoustic work, but is unable to obtain the original equipment used by the composer. If a technique truly cannot be achieved any other way than the original, then there is not much point in attempting a new realization or interpretation, and the work must therefore exist only at its moment of creation. Of course Stockhausen is speaking here of the unique sound qualities and tactile response of specific instruments used in one of his own works, and one can certainly sympathize with the desire to maintain the special qualities of an instrument that so compelled the composer in the first

place. On the other hand, some balance is necessary if the composer also desires a work to be open to continued interpretation by a wider pool of performers.

Technological innovation is a two-edged sword when it comes to musical works that incorporate interactive electronics. On the one hand, the passage of time has given us a body of new musical works. On the other hand, it has led to the obsolescence or disappearance of many of the instruments required to play them. Global communication via Internet has vastly increased awareness of existing works among interested performers and has also provided easy access to them. However, the extremely rapid development of computer technology has also greatly accelerated the turnover of electronic equipment and software used in musical applications. This has led to the rapid abandonment of devices and systems that were once considered "cutting-edge," many of which even served as the basis for certain key compositional processes. Simultaneously, the rapid development of technology has given rise to extremely powerful general-purpose computing systems that can accomplish many of the same things that previously required special-purpose proprietary equipment or software. This last development now makes it possible to incorporate the functions of older interactive electroacoustic works into more standardized and portable systems. The final result of all this technological advancement is, I believe, a net gain for the prospects of a functional electroacoustic repertoire. A number of researchers have already begun efforts in this direction, attempting to identify important works of interactive electroacoustic music literature, and adapting them to current technological resources.
Joel Chadabe describes new realizations of John Cage's *Bird Cage* (1972), David Tudor's *Rainforest* (1958), and his own *Solo* (1978) using various synthesis and signal processing software systems including Kyma and SuperCollider.\(^1\) Benny Sluchin, working at IRCAM in Paris, has created computer-based realizations of Stockhausen’s 1966 *Solo for Melody Instrument with Feedback*, first using Max on the NeXT computer (in 1992, in collaboration with Cort Lippe), and later in Max/MSP on a Macintosh computer (in 1998 with Carl Harrison-Faia).\(^2\) Christopher Burns, working at Stanford University, developed new performance realizations of classic electroacoustic works by Alvin Lucier (*I am Sitting in a Room*, 1969) and Stockhausen (*Mikrophonie I*) using both Pd (Linux) and Max/MSP (Macintosh) software.\(^3\) Clarinetist Bruce Bullock has documented his experiences in creating a performance implementation of Thea Musgrave’s *Narcissus* using alternate digital delay hardware.\(^4\)

Perhaps the most interesting work to date in this new field of preserving and re-implementing older electroacoustic works, and most relevant to this study, is that of Miller Puckette and his team at the University of California at San Diego. Puckette’s approach is to create “reference realizations” of interactive works using the Pd software

---


environment running on Linux (and other) computers. Pd offers a high degree of flexibility and sound quality, but it usefulness in terms of preserving and distributing realizations of older electroacoustic works among researchers comes from its relative transparency, due to the nature of open-source computing systems such as Linux and Pd. Puckette describes four works realized in this manner: Stockhausen’s 1970 *Mantra* (two pianos and ring modulators), Boulez’ 1985 *Dialogue de l’Ombre Double* (two clarinets and special processing), Philippe Manoury’s 1988 *Pluton* (solo piano and live electronics), and Kaija Saariaho’s 1991 *Noanoa* (flute and live electronics).

Puckette is the creator of the Pd software environment, and is also one of the inventors of object-oriented interactive music programming. His efforts in this area are therefore bound to carry a great deal of influence in the computer music community. His aim is to establish a performable repertoire of interactive electroacoustic works, which is also the ultimate goal of my own work presented in this document. Puckette’s project bodes well (especially in the short term) for the emerging field of interactive electroacoustic performance and for the preservation of electroacoustic repertoire.

---


36 The Open Source Initiative (OSI) is a non-profit organization that maintains the Open Source Definition for software development. Open Source software is characterized by freely available executable programs along with source code. The Open source definition specifies licensing standards designed to allow modifications to source code by other programmers, and to promote the shared development of software while avoiding the proprietary model followed by most commercial software developers. The Open Source Definition is available from the OSI web site: http://www.opensource.org

However, Linux and Pd are not universally accepted performance systems among performers of electroacoustic music, and their adoption by the majority of performers is far from inevitable. In the long run, I do not believe it is sufficient to transcribe these works from one specific system to another, even if the newer system is more generalized and accessible. Transcriptions are ultimately limited in their longevity because new implementations must always conform to the specific limitations of the chosen instrument. This is as true of open-source systems such as Linux and Pd as it is of any proprietary hardware or software. At some point, Pd will become outmoded, and these works will once again have to be adapted to newer technologies. If the only reference source is itself a transcription, a greater potential exists for a departure from the composer’s intentions in subsequent updates.

Therefore, I have adopted the approach of describing interactive electronic systems in a prose format, with supporting material in the form of mathematical equations, pseudo-code outlines of specialized algorithms, and generalized block diagrams of complex synthesis and signal processing instruments. I leave it to individual performers and technicians to choose the equipment and programming environments that best suit their preferences and budgets. As much as possible, I have attempted to base my analysis on the original systems, or at least on the composers’ own updated versions of the required electronics. My goal is to describe the interactive electroacoustic systems in sufficient detail to guide new realizations that are as faithful to the composers’ intentions as possible and to create a resource that could be used repeatedly to create accurate realizations of these works, regardless of the state of computer music technology.
CHAPTER 4
FOUR INTERACTIVE ELECTROACOUSTIC WORKS FOR CLARINET AND OBSOLETE TECHNOLOGY: AN OVERVIEW

The four works I am analyzing for this study present examples of works that make musical and idiomatic use of interactive electronic technologies that were once considered "cutting edge" but are now, within twelve to thirty years of their creation, relics of technological history. I would hasten to point out the obsolescence of the technology is not the same as obsolescence of the music itself. I have undertaken this project in an effort to separate the musical concepts employed by four very different composers from the ephemeral technologies with which their works were first implemented. Each composition considered in this study presents a different use of electronic technology for sound manipulation and generation and for human-machine interaction. The following is a brief summary of the technology employed in each work and the problems involved in its performance realization.

Thea Musgrave's *Narcissus* is notated very specifically for a now-obsolete digital delay system, a Vesta Koza DIG-411. This fact causes ambiguity when the directions in the score are applied to other equipment. Several authors have discussed the use of technology in Musgrave's *Narcissus* and the practical issues involved in performing it. Diane Boyd presents a thorough analysis of the dramatic aspects of *Narcissus*, and the impact of the Narcissus story on the compositional process and use of digital delay.38 Patricia Spencer, a co-commissioner of the work, has described the musical integration of

technology and the effect of technology on Musgrave’s compositional process. Bruce Bullock and Ron Burns describe an actual performance realization of the clarinet version of *Narcissus* using their own digital delay processor (essentially a transcription for the DigiTech DSP-256 XL, now obsolete). While these articles provide helpful insights, none provides a systematic and thorough explanation of the digital delay effects or their implementation.

Bruce Pennycook’s *Praescio IV* is part of a series of compositions written between 1989 and 1993 at McGill University, using Pennycook’s custom-designed “MIDI-Live” interactive computer music system. Since that time, the MIDI-Live software has fallen out of use and any new performance will require updated software. To date, no formal analysis of the processes or musical functions executed by the software has been published. Pennycook has provided invaluable assistance in understanding the interactive system and has also provided a completely new and updated version of the central parts of the interactive system software for analysis and performance. However, critical systems for live controller input (including pitch tracking) and sound output via synthesizer are still left to the performer.

Jonathan Kramer’s *Renascence* (1974) originally required an extremely complex tape delay system for live processing of the clarinet. In 1977, the composer made a

---

42 Bruce Pennycook, email to the author, April 5, 2003.
simplified version for clarinet and pre-recorded tape in order to facilitate performances in the absence of the required equipment. This revision was later updated in 1985 with a better quality tape. The composer has maintained that the original 1974 version is preferred, although he admits that performance with the tape delay system was never completely satisfactory in actual practice due to the overwhelming difficulties in keeping the tape delay system synchronized and stable during performance. At Columbia University in 1998, Kramer supervised a digital recreation of the tape delay system for a single performance by clarinetist Jean Kopperud. In my correspondence with the composer on this issue, he has indicated that the accuracy achieved in the digital version far outweighs the value of any interesting sonic side-effects of the original analog version and that digital recreations of the delay system are the preferred strategy for any future performances. The original 1974 score includes copious technical notes describing the setup and control of the analog tape delay system. Although these notes make it very clear what the electronics are supposed to achieve, some translation is necessary for reconstruction with a digital system. Furthermore, an accompanying pre-recorded tape is required for performance. Kramer created the original tape, based on recorded clarinet material (provided by Phillip Rehfeldt), in 1974 using analog equipment. Although the original 1974 pre-recorded tape is still available, my analysis includes a complete explanation of the composer’s method for its realization. This is intended as a guide to performers who wish to recreate this portion of the electroacoustic system using digital technology and their own clarinet playing as sound source material.

Cort Lippe’s *Music for Clarinet and ISPW* was written for a system that is no longer in use or manufactured (NeXT-based ISPW processor and Max/FTS software). No generalized analysis of the work exists so far.\(^4\) Published articles that mention this piece and the ISPW system for which it was developed such as those by Lippe and Miller Puckette are concerned primarily with the technical details of the signal processing software (Max and FTS) and not with the actual musical processes, events, and interactions contained in the score.\(^5\) My analysis of this work is an attempt to systematically describe all of the synthesis and signal processing algorithms and interactive control systems in general terms that could be translated to alternate computing platforms. The composer has provided me with a current implementation of the interactive system software translated to the Max/MSP software environment. Lippe has also provided invaluable assistance in understanding and explaining the extremely complex inner workings of the performance software and the synthesis and signal processing algorithms it is designed to implement. Due to the extreme complexity of several of Lippe’s signal processing modules, general descriptions are given in the text and detailed block diagrams of each signal-processing instrument are included in appendix C.

\(^4\) Cort Lippe, email to the author, March 27, 2003.

CHAPTER 5
ANALYSIS OF INTERACTIVE TECHNOLOGY IN THEA MUSGRAVE’S
NARCISSUS (1987) FOR CLARINET IN B-FLAT AND DIGITAL DELAY

My interest in Thea Musgrave’s *Narcissus* began after hearing clarinetist F. Gerrard Errante’s recording in 1991. McGregor Boyle, my major advisor at the Peabody computer music studios, was the technical director for one of the first performances of *Narcissus*, and had had first-hand experience with the original equipment as well as direct supervision from the composer in the concert production. Advice from both Boyle and Errante was invaluable in preparing a performance of *Narcissus* using my own digital delay equipment in April of 2000. The experience of implementing Musgrave’s directions using new equipment led me to wonder if the process could be formalized for the benefit of other players. A number of performers interested in the piece have expressed their enthusiasm for such an effort as well. The result is the following analysis of Musgrave’s digital delay system.

5.1 HISTORICAL BACKGROUND

*Narcissus* was written in 1987 for flute and digital delay, as one of four works resulting from a National Endowment for the Arts Consortium Commissioning Grant. The commissioning flutists were Wendy Rolfe, Harvey Sollberger, Patricia Spencer, and

---

Robert Willoughby. Clarinetist F. Gerrard Errante, a noted specialist in music for clarinet and electronics, served as technical consultant to Musgrave during the composition of this work. In return, the composer made a transcription of *Narcissus* for clarinet. Novello Music now publishes both flute and clarinet versions of this work.

*Narcissus* requires a digital delay system controlled from the stage by the performer. Though relatively simple, the electronics are notated for the particular model of digital delay equipment used by the composer, a Vesta Koza DIG-411. This instrument is no longer manufactured. Even so, *Narcissus* is performed often, using alternate equipment. Musgrave’s technical notes included in the score refer to several other delay devices used by performers who were involved in the creation or early performances of this work. However, the specificity of the notation for the DIG-411 poses a few problems in reconstructing the digital delay system for those who do not have the benefit of advice from either the composer or the handful of performers who had experience with the original implementation.

The score includes three pages of technical notes, compiled by Musgrave, Patricia Spencer, Karen Bennet, and F. Gerard Errante, explaining the digital delay system. The notes include stage diagrams, a sketch of the front and back panels of the DIG-411, and a list of warnings and recommendations for performance setup and preparation. These notes are very helpful since they provide some insights into the composer’s intentions, as well as recommendations for handling certain common performance problems. However,

---

several key digital delay effects are not explained in general terms, but are instead expressed only as DIG-411 knob positions. In order to reconstruct the digital delay system using updated or alternate technologies, the exact nature of these effects and their parameters must be explicitly understood. This information is not contained in the technical notes accompanying the score, nor is it to be found in any of the available literature on this work to date.

The original Vesta Koza DIG-411 Digital Delay used by Musgrave to create this piece has been in the possession of flutist and Narcissus co-commissioner Wendy Rolfe. Dr. Rolfe has very kindly lent me the DIG-411 for this research, and I have compared the functions of the original instrument to my own software-based reconstruction. With this comparison, I have analyzed and defined the digital delay system as a set of signal processing algorithms and related parameters, rather than as proprietary settings specific to one now-obsolete device.

5.2 MUSICAL ROLE OF TECHNOLOGY

Narcissus is programmatic, musically illustrating the Greek myth of Narcissus, who was so drawn to his own reflection in the water that he drowned trying to grasp it. Musgrave provides the following program text, fragments of which appear throughout the score:

Narcissus wanders through the forest, observing, enjoying ... unselfconscious but self-absorbed.

He sees a pool of water and then as he approaches notices his reflection in the water. He is intrigued and then jumps back in fright. Once more he approaches ... it is still there.
Narcissus steps away from the pool to consider this phenomenon. Several times he approaches, the figure is always there watching him.

In the shimmering sunlight Narcissus seems to see this glorious and attractive being moving in the rippling water. He is dazzled and slowly holds out his arms. To his amazement the figure responds.

In awe and wonder Narcissus approaches closer and closer. With a sudden change of mood Narcissus dances happily and playfully ... the figure echoing him. But then Narcissus begins to question anxiously the lack of any independent response ... is he being mocked? He gets more and more agitated and finally in a fury he rushes headlong into the water to grapple with the figure. The waves surge up and Narcissus is drowned. There is a distant shimmering vision of Narcissus and his reflection. Then in the setting sun the vision disappears, the forest is empty and the pool lies undisturbed.\(^\text{49}\)

\textit{Narcissus} contains a great variety of moods and textures, from the unaccompanied opening melody to a playful scherzo, and angular and agitated semi-improvised sections. The digital delay is more than an effect in this piece. Programmatically, it assumes the role of Narcissus’ reflection, and at the end, the water itself. Musically, the delay extends the solo instrument in time, creating a sonic reflection that very nearly takes on a life of its own. Repeating delay effects are used to create harmonies out of a single melodic line, and continuous delay-time modulation (defined in section 5.3.2, below) is used to bend the pitch of the delayed sound.

Duration is listed in the score as 14 minutes, but this is variable considering the numerous indications for rubato, open-ended pauses designed to allow electronic effects to fade before moving on, and the variability of the semi-improvised segments. All electronic sounds are generated in real time directly from the sound of the live clarinet.

using the basic features of a digital delay, and there are no pre-recorded elements. Therefore, the electronics should be viewed as a flexible extension of the solo clarinet rather than as an accompaniment.

5.3 ANALYSIS OF TECHNOLOGY COMPONENTS

The electronics required for Narcissus can be broken down into three basic parts: a sound reinforcement system (microphone, amplifier, and loudspeakers), the digital delay system (creating echo effects with several variable parameters), and a control interface (foot switches, pedals, and other devices used to change delay settings during performance). The following is an explanation of the required effects, the technical methods for implementing them, and the musical contexts in which they appear.

5.3.1 Sound Reinforcement

The sound reinforcement requirements for *Narcissus* are straightforward and will generally pose no special problems in adapting to available equipment. Musgrave suggests a contact microphone to provide audio input from the clarinet to the digital delay system.\(^{50}\) Other microphone systems can work equally well for this piece provided they offer enough isolation from the loudspeakers to prevent signal feedback or other extraneous noise.

The setup diagram included in the score specifies that the signal output from the digital delay should be routed to a loudspeaker placed stage-right, while the amplified but

\(^{50}\)Musgrave, *Narcissus*, technical notes preceding the score.
otherwise unaffected ("dry") clarinet signal should be routed to a second speaker placed stage-left. This relatively simple arrangement is intended to force an aural separation between the clarinet and its digital "reflection."

![Audio system setup diagram](image)

Figure 5.1. Audio system setup

5.3.2 Digital Delay System

A digital delay is a standard effect found on most commercially available signal processing equipment. In its basic form it creates a simple echo—the input signal is played back after a specified time interval has elapsed. This time interval is usually expressed in milliseconds, but may range up to several seconds. Various manipulations of the delay can produce interesting sonic results, some of which are called for in Narcissus. Six digital delay parameters must be controlled during performance of this work: delay interval (time), feedback, time modulation, hold, volume, and bypass.

---

51 Musgrave, Narcissus technical notes preceding the score.
Delay Time. Musgrave indicates three distinct delay times to be applied at various points. The DIG-411 delay time was set using two separate knobs. The first determined the base delay time (labeled “range”), with available settings of 2, 8, 32, 128, and 512 milliseconds. The second selected a multiplier of the base delay time, with values ranging continuously from 0.5, to 2 (labeled “time”). Therefore, the base delay time could be set anywhere from one half to double the base delay time. Narcissus calls for a base delay time set to 512 milliseconds throughout. Multiplier values of 0.5, 1.0, and 2.0 are used, resulting in 3 separate delay times: 256, 512, and 1024 milliseconds.

In Figure 5.2 the delay time is set to 1024 milliseconds (512 x 2). With the sound of the clarinet echoing approximately one second later, the first note, F, is repeated over the E-flat, which is repeated over the D. In this passage, feedback (see below) is set to 6 (the highest setting in the piece), and therefore, the F is also heard repeating again (but softer) over the D.

![Figure 5.2. Score example: delay time](image)

Delay feedback. Delay feedback creates a repeating echo by routing a portion of the delay output back to its own input. Normally, the signal being fed back is at a lower volume than the output, and the repeating echoes fade away gradually, with a duration
that depends on the amount of feedback. Musgrave uses this technique very effectively to portray the environment and character of Narcissus:

![Score example: delay feedback](image)

**Figure 5.3. Score example: delay feedback**

In the passage shown in Figure 5.3, delay feedback creates a fairly thick texture. The “shimmering sunlight” and “rippling water,” referred to by the program text, are expressed in the accelerating and ascending arpeggios, which continue to repeat while fading away (the “Hold” feature is explained below).

Delay feedback is indicated in Narcissus according to settings specific to the DIG-411: values are indicated from 0 to 6 (on a scale from −10 to 10). Musgrave describes feedback settings as controlling the “number of repeats” in the delay. This should not imply that these repeats are discrete repetitions of a musical phrase, or that a feedback setting of 6 will yield six repeats. In fact, a standard digital delay with variable feedback will route an *attenuated* portion of the delayed signal (usually scaled from 0-

---

52 Settings between 0 and −10 are not audibly different in any way from settings between 0 and 10. It is not completely clear, in the absence of a DIG-411 user manual, exactly what the purpose of negative feedback values might be. Since negative values are not used in Narcissus, I will defer a definitive explanation of this curious feature of the DIG-411 to future research.

53 Musgrave, Narcissus score, 1.
100 %) back to the delay inputs. The result is that sound repeating in the delay gradually diminishes in volume as less and less of it is fed back each time. A feedback setting of 0 will simply produce a single echo of the input signal, with no portion of the echo redirected to the delay input. The DIG-411 operates on this principle as well, so alternate implementations of the delay system should be constructed using scaling factors equivalent to the DIG-411 settings used by the composer.

The most logical assumption is that the settings from 0 to 10 would correspond to feedback gain from 0 to 100% of the original signal, with each number on the dial representing a 10% increment. Therefore, the maximum setting in the score, feedback at 6, would be interpreted as 60%. However, the DIG-411 behaves quite differently. Settings above 6 actually produce some very undesirable effects: rather than diminishing and fading away, the repeated echoes become louder and begin to distort, eventually overloading the system. Apparently, settings above 6 create feedback levels effectively greater than 100%, so that even the quietest sounds introduced into the system quickly build into an overwhelming noise. It would seem that by limiting the delay settings in Narcissus to 6 and below, Musgrave was simply working within the idiomatic boundaries of the equipment on hand, rather than choosing settings according to any arbitrary rules—a clear demonstration of Musgrave’s “hands on” approach to composition with technology noted by Patricia Spencer in her description of this work and its inspiration.\(^4\)

The practical question remains: using an alternate system, what levels of delay feedback would most closely match the DIG-411 settings indicated in the score? To

\(^4\) Spencer, “The Musical Shape of Technology,” 47.
answer this question, I measured the output of the DIG-411 using a test application created with Max/MSP software. With delay time set to 512 milliseconds (512 x 1), a synthesized test signal was sent to the DIG-411 input, and the delay output was recorded into an AIFF sound file for each of the three feedback settings (2, 4, and 6). A feedback setting of 6 yielded approximately 18 seconds of diminishing repetitions (or 36 repeats). With feedback at 4, repetitions lasted for 5 seconds (10 repeats). Feedback at 2 created 2.5 seconds of echo (5 repeats).

The same signal was then put through a software-based delay with variable feedback (also in Max/MSP), in order to simulate the results of the DIG-411 test. The DIG-411 feedback setting of 2 corresponded to a software delay feedback setting of 25%. DIG-411 feedback of 4 was equivalent to 50%, and the DIG-411 setting of 6 corresponded most closely to 75% feedback in the software simulation.

It should be noted that on the DIG-411, feedback is not limited to discrete settings. The feedback control knob is a potentiometer that allows for settings anywhere along the range between the minimum (0) and maximum (10) values. Considering the fact that the performer is required to quickly change these settings manually while handling a flute or clarinet as well, it is likely that some variation would occur in performance from the notated values. In actual practice with this machine, feedback settings of 2, 4, and 6 would be rough targets. Furthermore, differences in microphones

55 A test tone (sine wave at 440 Hz) was used to first calibrate the DIG-411 input so that the incoming signal would be as close to 0dB as possible (using the front panel “Headroom” LED as a guide). The same synthesized tone (440 Hz sine wave with an immediate attack and linearly descending decay envelope (1-0) over 500 msec) was sent to the DIG-411 input for each of the three feedback settings. The “bell-like” sound of this test signal enabled easy counting of repetitions.
(and their placement), individual playing style, and concert hall acoustics will produce slightly different results in actual practice. My analysis of the DIG-411 feedback settings, as 25, 50, and 75 percent of the delay output signal, should be used only as an approximate guide for recreation of the digital delay. Some variation or adjustment may be necessary to account for real-world performance situations.

Modulation. The modulation effect is not clearly defined in the score or in any other published article to date concerning this work. According to clarinetist F. Gerard Errante (who assisted the composer in designing the electronic effects), “the modulation effect is meant to be a gradual, ‘undulating’ pitch transformation, like a slow, wide vibrato.”

The modulation effect is applied at the end of the piece, as the character of Narcissus drowns in the reflecting pool (“The waves surge up, the figure is shattered & Narcissus is drowned”). In Figure 5.4, the modulation effect makes its debut, portraying Narcissus’ watery demise. The short delay time in this passage (512 x 0.5, or 256 milliseconds) creates a very close mirror of the solo clarinet, warped slightly by the oscillating pitch of the delayed signal. Delay feedback, set to 6, adds thickness to the texture.

---

Pitch fluctuation of this sort can be achieved by continuously varying (modulating) the delay time by a small amount, similar to a typical "flange" effect. As the delay time shifts, audio samples in the delay buffer are played back at a shifting rate of speed. As playback speed increases, the pitch rises. As playback speed decreases, the pitch falls. This phenomenon is similar to the "Doppler" effect on pitch from a moving sound source, and it should be noted that this pitch shifting effect created by a continuously variable delay only occurs while the delay time is still changing. Once the delay time is set, the pitch stabilizes at its original pitch level. Furthermore, the amount of pitch shifting is directly related to the amount of offset from the original delay time and the speed of modulation. Larger offsets at faster speeds create more radical pitch shifting effects.

The score indicates that "[m]odulation speed remains at 0 throughout, modulation depth ranges from 0 – 10 (0 – 3 [sic] used)."\(^{57}\) According to the front panel of the DIG-411, modulation speed is actually scaled from .1 Hz to 10 Hz, with a continuous range of settings available between the minimum and maximum settings. Therefore the

\(^{57}\) Musgrave, *Narcissus*. Although the technical notes indicate modulation values from 0-3, the score contains only modulation settings from 0 to 2.
modulation speed setting of zero indicated in the score should be implemented as 0.1 Hz when using other equipment. The score indicates depth values of 0, 1, and 2 on a scale of 0-10. An analysis of the DIG-411 output, with an input signal of a sine tone at 440 Hz, shows that a modulation setting of 0 causes no pitch deviation, a setting of 1 causes a pitch fluctuation between 438 and 442 Hz (± 5 cents), and a modulation setting of 2 produces a pitch variation from 432 to 248 Hz (± 30 cents). Using a modulated delay constructed in Max/MSP software, I was able to produce the same pitch variations as the DIG-411 and record the amount of delay time modulation required to produce such results in terms of variation from the base delay time in milliseconds. In other words, to produce the ± 2 Hz pitch variation recorded from a modulation setting of 1, the delay time should be varied by approximately 7 milliseconds in either direction, i.e. delay time continuously fluctuates between 1017 and 1031 milliseconds at a rate of one cycle per 10 seconds (0.1 Hz). For a mod setting of 2 (pitch fluctuation between 432 and 448 Hz), a delay time modulation of 21 milliseconds is required, i.e. delay time fluctuates gradually between 1003 and 1045 milliseconds.

One caveat: on the Vesta Koza DIG-411, the knob control for modulation depth allows for continuous adjustment between values, rather than discrete settings. Therefore, the exact values for pitch variation may be slightly different from one performance to the next. The values I have come up with should be used as a guideline only, and not as absolute values. However, I believe that this analysis is precise enough to serve as a guide for future implementations of the modulation effect.
**Hold.** The hold function allows the performer to capture a short duration of sound in the delay line that loops continually. While the hold is engaged, no new sounds are added to the delay, so the hold loop becomes a background to whatever the performer plays at that time.

The DIG-411 had a particularly smooth hold feature, with no audible clicks or other artifacts creeping in to the sound when the feature is engaged or released. Musgrave mentions this requirement as an absolute necessity for any performance implementation of the delay system.\(^{58}\)

The hold effect is used quite effectively in Narcissus, especially at measure 247 (*Giocoso – Dopio movimento*, “Narcissus then responds playfully, happily …”). Figure 5.5 shows delayed arpeggios synchronizing to form brief ostinato chords that accompany the continuing melody in the clarinet. The four notes before the hold is engaged (B-G#-G-F) align as a chord, since the delay time coincides with the eighth note pulse, and they continue to repeat as background accompaniment for the next two and a half measures until the hold is released.

![Figure 5.5. Score example: delay hold](image)

\(^{58}\) Musgrave, *Narcissus*, technical notes preceding the score.
**Volume.** The volume of the delay output is to be controlled by foot pedal. Several places in the score call for the delay signal to gradually fade in or out to silence. The original setup used a simple analog volume pedal between the delay output and the loudspeaker. This technique is used programmatically in *Narcissus* to illustrate the main character approaching or retreating from the reflecting pool where he sees his own image:

![Score example: delay volume](image)

*Figure 5.6. Score example: delay volume*

**Bypass.** The bypass function is used in *Narcissus* to turn the entire delay system on or off. This feature is used at the opening of the piece so that the unaccompanied introductory section (“Narcissus wanders through the forest …”) is unaffected by the digital delay. Once the bypass is disengaged (by foot switch control), the delay system is active.

![Score example: delay bypass](image)

*Figure 5.7. Score example: delay bypass*
**Digital Delay System Summary.** The digital delay system for *Narcissus* consists of: 1) an audio input source (a microphone); 2) an echo effect with variable delay time, feedback, time/pitch modulation, and functions for delay hold, bypass, and volume control; and 3) audio output via amplifier and loudspeakers. The delay effects can be easily described in terms of standard audio signal processing algorithms and the DIG-411 settings given in the score can be translated into specific effect parameters. Table 5.1 (below) summarizes the required effects and their variable parameters as notated in the score and as actual values to be used as a guide for reconstruction with other equipment.
Table 5.1. Delay effect parameters: notated and actual values

<table>
<thead>
<tr>
<th>Effect</th>
<th>Notated</th>
<th>Actual Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Delay Time</em> (time interval between input signal and its echo)</td>
<td>512 x 0.5</td>
<td>256 milliseconds</td>
</tr>
<tr>
<td></td>
<td>512 x 1</td>
<td>512 milliseconds</td>
</tr>
<tr>
<td></td>
<td>512 x 2</td>
<td>1024 milliseconds</td>
</tr>
<tr>
<td><em>Feedback</em> (amount of delay output signal routed back to its input; duration of repeating delay)</td>
<td>0</td>
<td>No feedback (1 repeat)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>75%</td>
</tr>
<tr>
<td><em>Modulation Depth</em> (slow cyclical pitch shifting; Speed = 0.1 Hz)</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Delay time +/- 7 msec</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Delay time +/- 21 msec</td>
</tr>
<tr>
<td><em>Hold</em> (delay loops continuously in background allowing “ostinato” effect)</td>
<td>On/Off</td>
<td>Feedback = 100%, Input Off</td>
</tr>
<tr>
<td><em>Bypass</em> (disengages the entire delay system by cutting input)</td>
<td>On/Off</td>
<td>Input Off</td>
</tr>
<tr>
<td><em>Volume</em> (output level from the delay system)</td>
<td>&lt; &gt;</td>
<td>Continuous control from 0 to full volume</td>
</tr>
</tbody>
</table>

Figure 5.8 shows the complete digital delay system and the signal flow between the various components:

![Figure 5.8. The complete digital delay system](image-url)
5.3.3 Control Interface

Because variable parameters must be changed during performance, an interface of some sort is necessary to control the delay system from the stage. Musgrave’s directions in the score are so specific to the DIG-411 that any new realization of the work will require some departure from the notation. The DIG-411 has back panel inputs for a hold pedal, a bypass pedal, and a volume pedal. On the front panel are knob controls for “Input” (0 – 10), “Feed Back” (-10 – 10), Modulation “Speed” (0.1 – 10 Hz) and “Depth” (0 – 10), Delay “Range” (2, 16, 64, 128, 512) and “Time” (0.5 – 2.0), and “Delay” (0 – 10) and “Dry” (0 – 10) Output levels. Additional on/off switches control “Hi-Cut” (filter), “Bypass,” and “Hold.” The Vesta Koza DIG-411 front panel is shown in figure 5.9.

Figure 5.9. Vesta Koza DIG-411 front panel

In practice, the performer would be required to manipulate knobs 2 (feedback), 4 (modulation, depth), and 6 (time) by hand. Bypass, hold, and volume are controlled by two foot-switches and a connected to inputs on the back panel of the DIG-411 and a volume pedal placed between the DIG-411 output and the amplifier.

Alternative delay systems (whether hardware- or software-based) may not feature the same physically accessible controls for the aforementioned variable parameters. Aware of this fact at the time the clarinet version was published, Musgrave mentions the
possibility of using a third footswitch to advance through a sequence of pre-set delay settings. Such an arrangement allows the performer to easily change delay system parameters while remaining focused on the music. The points in the score that require changes to these three parameters are as shown in Table 5.2.

<table>
<thead>
<tr>
<th>Score Event</th>
<th>Delay time</th>
<th>F/B</th>
<th>Mod.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. m. 1 - “Narcissus wanders through the forest, observing ...”</td>
<td>512 x 0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2. m. 78 - “Is ‘It’ Still there?”</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3. m. 89 - “Narcissus steps back from the pool ...”</td>
<td>512 x 1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4. m. 172 - “In the shimmering sunlight ...”</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5. m. 247 - “Narcissus then responds playfully, happily ...”</td>
<td>512 x 0.5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6. m. 316 - “Narcissus anxiously questions...”</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7. m. 370 - “The waves surge up ... Narcissus is drowned”</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8. m. 387 - “All that remains is a distant shimmering vision ...”</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9. m. 398</td>
<td>512 x 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. m. 426</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11. m. 428 - “The vision disappears ... the forest is empty ...”</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Therefore, eleven pre-programmed delay system changes are required, two of which require changes to two parameters simultaneously (events 3 and 5). Many strategies currently exist for cycling through a series of effects changes, either by footswitch control or by other means. More than likely, new strategies will emerge in the near future for controlling parameter changes within an interactive computer music system, to which this list of control events should be easily adaptable.

---

59 Musgrave, Narcissus, technical notes preceding the score (“Requirements”).
5.4 SUMMARY

The digital delay system required for *Narcissus* could be easily reconstructed using a wide range of equipment or software-based audio processing environments. I have based my analysis of the digital delay system, and its use in the score, on a close examination of the original Vesta Koza DIG-411 system used by Musgrave. Therefore I hope that it leads to future performance realizations that are faithful to the intentions of the composer. By translating DIG-411 settings into specific parameter values for standard signal processing algorithms, I hope to provide enough information to serve as a reliable guide for anyone attempting a recreation of the digital delay system for *Narcissus* using alternate equipment or technology. My own software-based realization of the digital delay system is discussed in detail in chapter 9.
In 1991, while researching the emerging field of real-time interactive computer music systems, I ran across Bruce Pennycook's description of his *Praescio* series of compositions in the *Computer Music Journal*. The concept of flexible, performer-controlled electronics, as outlined by Pennycook, was revelatory to me, and was a formative factor in my decision to pursue advanced study in computer music. At the Peabody Computer Music Studios, my digital music programming instructor and mentor was Ichiro Fujinaga, a former member of Pennycook's team at McGill University that had developed the interactive software system for the *Praescio* series. The performance aesthetic of this system, including expressive control of electronics, and flexibility determined by the musical sensibilities of a live performer rather than a fixed tape, were guiding principles in my computer music training at Peabody. *Praescio IV* is a work that was groundbreaking in its day, but is not playable in its original form due to technological obsolescence. It is an excellent counterpoint, musically and technologically, to the other works considered in this study.

6.1 HISTORICAL BACKGROUND

Bruce Pennycook's *Praescio IV* is part of a series of compositions written between 1989 and 1993 at McGill University using the custom-designed "MIDI-Live"
interactive computer music system. Praescio IV was written in 1990 for clarinetist Jean-Guy Boisvert and premiered at the 1990 Clarinet Fest International in Quebec City. Boisvert performed Praescio IV subsequently at the 1991 International Computer Music Conference at McGill University. In 1995, the composer prepared an updated version of the software for Boisvert's Canadian concert tour. Boisvert has recorded Praescio IV on a compact disc titled Zodiak/Zodiaque.

The MIDI-Live interactive system for Praescio IV consisted of software that captured live input from the performer via MIDI controllers and pitch-tracking hardware and produced electronic sound via MIDI-controlled synthesizers. The pitch tracking was originally accomplished using the IVL 4000 Pitchrider pitch-to-MIDI interface. Additional control of the system was accomplished using a custom-built harness that added MIDI control keys to the clarinet. A sound module (Emu Proteus I) generated the synthesized accompaniment in response to MIDI output from the system software, based on pre-recorded MIDI sequences and on direct control by the pitch tracking system (effectively turning the clarinet into a MIDI controller).

Pennycook's 1995 update of the Praescio IV software was developed in Max for the Apple Macintosh computer (then distributed by Opcode Systems). However, this version no longer runs on the current generation of Macintosh computers and another update is now necessary. The IVL 4000 Pitch Rider is also no longer available. Therefore the pitch-tracking functions required for Praescio IV must be accomplished by

---

other means. The Proteus sound module also may prove difficult to obtain for many
performers and alternate means of tone generation may be necessary for performance of
this piece. In fact, the composer recommends using a more up-to-date sound set using
synthesizers chosen by the performer.

As of spring 2004, Pennycook has created a new update of the performance
software using Max/MSP. Programmer Dale Stammen has updated an essential Max
object (playSMF) used in the Praescio IV software to run on the current generation of
Macintosh computers. I have now added a pitch-tracking component to this software that
emulates the features of the IVL 4000 Pitchrider used in the original incarnation of this
piece. While this current version was developed to run on late-model Apple Power­
Macintosh computers (G3 – G5), the recent release of Max/MSP for Microsoft Windows
XP means that the current software is theoretically cross-platform compatible.

Now that there is a functional version of the interactive system software,
performances of Praescio IV are possible at least for the present and very near future. As
demonstrated by the history of technological change over the relatively short lifetime of
this work, it is likely that new versions of the performance software will have to be
constructed again within a few years.

6.2  MUSICAL ROLE OF TECHNOLOGY

The interactive system used in Praescio IV functions as a separate instrument that
accompanies and contrasts with the solo clarinet. All electronic sounds in Praescio IV
are generated by a synthesizer, rather than through transformations of the clarinet sound.
The distinction between the acoustic and electronic instruments is quite clear. Pennycook describes the relationship of the clarinet to the synthesized sound in his program notes:

The work explores the various relationships among the sonic resources—clarinet alone, synthetic sounds, and “colorized” clarinet where pitches from the clarinet are tracked by the system and enhanced with synthetic sounds. The interplay of these three resources is used to articulate the formal organization of the work.\(^{62}\)

The *Praescio* series was conceived as an attempt to achieve meaningful interaction between live performers and electronic musical instruments and processes. Pennycook’s aim was primarily “the removal of prerecorded tape from electroacoustic performance” by creating flexible computer-controlled MIDI systems.\(^{63}\)

The interactive system produces sound both in direct response to actions of the performer and according to pre-programmed automated processes. Pacing is completely under the control of the performer who is therefore free to interpret the score in a flexible manner, rather than having to play to a fixed timeline.

A single player can perform *Praescio IV* without additional technical assistance. Although the interactive system plays an accompanying role, *Praescio IV* is essentially a solo work.

6.3 ANALYSIS OF TECHNOLOGY COMPONENTS

The technology requirements for *Praescio IV* are extensive, although current trends in computer music technology make it increasingly possible to reduce the overall


size of the performance setup by incorporating more functions into the system software. However the system is implemented, certain elements must be present, whether in physical or virtual form. The complete system consists of a sound input and output hardware, loudspeakers, microphones, input devices, a synthesizer, score-related data files, and interactive MIDI processing software.

6.3.1 Sound Reinforcement

The sound system requirements for *Praescio IV* are fairly standard. Two-channel playback from the synthesizer is recommended, though the composer gives no special instructions for loudspeaker placement and no sound spatialization effects are employed. The clarinet may be amplified slightly to balance with the electronics, but this is not an absolute requirement. A microphone is required to provide input to the interactive system from the clarinet for purposes of pitch tracking. For this, a contact microphone is recommended in order to isolate the clarinet sound as much as possible. Separate standard microphones might be used for amplification of the clarinet.

6.3.2 Control Interface

The interactive system for *Praescio IV* relies on live input signals from the performer to control playback of pre-recorded sequences, to initiate changes to various system parameters, and for direct expressive control over volume and sustain parameters of the synthesizer output. Four separate devices are used to provide these signals to the system: an event trigger, a pitch tracker, a sustain pedal, and a volume controller.
**Event Trigger.** A simple trigger signal is used to advance sequentially through a list of system events. This technique is not unlike using a remote control to advance through a slide show. Since the performer is responsible for triggering event changes, the device used should be convenient to operate from the stage while playing. Events to be initiated by the trigger interface are notated in the score by the \[T\] symbol, as shown in figure 6.1, below:

![Fig6.1](image)

**Figure 6.1.** Score example: event 2 triggered by footswitch control

The 1990 version of *Praescio IV* used a custom-designed harness with extra keys mounted to the clarinet, designed by Pennycook and Eric Johnstone at McGill University. Only one harness was ever built, so this is not a reasonable option for most clarinetists. While the clarinet-mounted key switches may have been ergonomically more convenient for the performer, in terms of system interaction they were no different

---

64 Pennycook, “Machine Songs II,” 24
from a simple MIDI footswitch. Adapting to available devices would in no way affect the functionality of the system.

**Pitch Tracking.** Pitch data derived from real-time analysis of the live clarinet signal is used for two purposes in *Praescio IV*. Pitch information is used to advance system events in the same way as the trigger, described above. For these events, the system waits for a particular pitch to be played before initiating the next event. Event 3 (shown in figure 6.1 above) is cued after event 2 is triggered by the foot pedal, but is not executed until the clarinetist plays the low E.

At certain event numbers the system follows the clarinet pitch and matches it with synthesized notes and parallel harmonies. The composer refers to this as "colorization" of the clarinet sound through synthesizer doubling. Attack volume (MIDI velocity) of the synthesized sound is derived from an analysis of the clarinet’s attack amplitude for each note detected. These events are termed "THRU" events by the composer because MIDI data from the pitch tracker is passed directly "thru" to the synthesizer.⁶⁵

Therefore, *Praescio IV* requires a pitch-tracking system that reliably tracks the clarinet and accurately identifies, in real time, the notes being played. The pitch-tracker must also measure attack amplitude and assign a numerical value. In MIDI terms, this would be a note velocity value between 0 and 127. The system should provide a way for

---

⁶⁵ The use of the word “THRU” by Pennycook is a reference to the MIDI specification, which defines IN, OUT, and THRU ports for MIDI devices. THRU ports pass data received at the MIDI IN without alteration or processing. The “THRU” events in *Praescio IV* involve processing and alteration of the pitch tracker data, and therefore Pennycook’s use of the term “Thru” is inspired by, but not exactly in compliance with, the MIDI definition of THRU.
calibrating velocity values to the minimum and maximum attack dynamics produced by
the individual clarinetist using his or her own microphone and audio input system, based
on a pre-concert sound check.

_Sustain Pedal._ A footswitch or other device is required for turning on and off the sustain
state of the synthesizer. Similar to the sustain pedal of a piano, MIDI sustain holds any
notes that are initiated by Note-On messages to the synthesizer until sustain is released.
The sustain pedal is indicated in the score with the symbol $\text{[s]}$. Like the event trigger
described above, the 1990 version of _Praescio IV_ used the custom harness, with an added
fourth-finger key acting as a sustain pedal. Pennycook now recommends a simple MIDI
footswitch.

_Volume Controller._ Several points in the score require direct dynamic control of the
synthesizer output volume. All versions of the _Praescio IV_ software to date have
required a standard MIDI pedal to send volume messages to the synthesizer. Whether or
not a MIDI pedal is used, the volume controller should smoothly control volume fades
between minimum and maximum amplitudes for only the synthesizer channel to which it
is assigned.
6.3.3 Synthesizer

*Praescio IV* requires a multi-timbral polyphonic synthesizer capable of playing back pre-programmed MIDI sequences (a list of synthesizer control data messages with associated playback timings) on at least 16 separate channels simultaneously. The aforementioned capabilities are currently standard for most professional-grade synthesizers compatible with the MIDI specification. Alternatively, sound synthesis could be integrated into the interactive system software (using software-based synthesizers or samplers), removing the need for an external sound module and streamlining the equipment needed on stage.

The composer leaves synthesizer voicing up to the performer so long as basic sound types are used for the appropriate channels (sustained, percussive, etc.). Table 6.1 (below) gives Pennycook's general suggestions for voicing each channel of the synthesizer. This guideline is based on the Proteus I sound set listed in a sub-patcher of the current Max/MSP software. Pennycook recommends a richer sound set than what was possible with the original Proteus I, and suggested Boisvert's recording as an audible guide.\(^{66}\) In any case, synthesizer voice programming is the performer's prerogative in this piece.

\(^{66}\) Bruce Pennycook, email to the author, March 17, 2004.
Table 6.1. Suggested sound set for *Praescio IV*

<table>
<thead>
<tr>
<th>Channel</th>
<th>Voice type</th>
<th>Pennycook’s general description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pad</td>
<td>“Spacey” FM-type sustained sound</td>
</tr>
<tr>
<td>2</td>
<td>Piano</td>
<td>Concert grand piano sampled in stereo</td>
</tr>
<tr>
<td>3</td>
<td>Pad</td>
<td>“Hard-edged” pad</td>
</tr>
<tr>
<td>4</td>
<td>Pad</td>
<td>“Stringy” pad, big</td>
</tr>
<tr>
<td>5</td>
<td>Bell</td>
<td>“Tingly,” bell-like with sustain</td>
</tr>
<tr>
<td>6</td>
<td>Pad</td>
<td>“Hard-edged” pad, like #3, but more edge</td>
</tr>
<tr>
<td>7</td>
<td>Piano</td>
<td>Concert grand piano sampled in stereo</td>
</tr>
<tr>
<td>8</td>
<td>Brass</td>
<td>“Cheesy” synthesized trombones</td>
</tr>
<tr>
<td>9</td>
<td>String</td>
<td>Big flanged strings</td>
</tr>
<tr>
<td>10</td>
<td>Piano/pad</td>
<td>Piano with synthesizer sustain tails (long decay)</td>
</tr>
<tr>
<td>11</td>
<td>Percussion</td>
<td>Vibes</td>
</tr>
<tr>
<td>12</td>
<td>Pad</td>
<td>“Airy,” “spacey” pad – diffuse, with slow attack and decay</td>
</tr>
<tr>
<td>13</td>
<td>Brass</td>
<td>French horn like with light flange</td>
</tr>
<tr>
<td>14</td>
<td>String</td>
<td>Big flanged strings (as #9 above)</td>
</tr>
<tr>
<td>15</td>
<td>Guitar</td>
<td>A &quot;stratocaster&quot; guitar sound</td>
</tr>
<tr>
<td>16</td>
<td>Pad</td>
<td>FM-type sustain used for the clarinet THRU sustains</td>
</tr>
</tbody>
</table>

6.3.4 Prepared Data

The interactive system for *Praescio IV* operates according to a set of data files that are specific to the piece, analogous to a traditional score. An “event list” stores a series of cues for system actions and parameter changes tied to specific transition points in the score. In addition, a collection of pre-recorded MIDI sequences (stored as Standard MIDI Files) drive synthesizer playback according to parameters stored in the event list.

*Event List.* For each of the seventy-seven event points in the score, the event list specifies a trigger condition (either a signal from the footswitch or a specific note played by the clarinetist) and a series of parameters for either controlling SMF playback (“Play”
events) or for defining colorization/harmonization settings ("THRU" events). An excerpt from the event list is given in figure 6.2, below. The complete event list for *Praescio IV* can be found in appendix B (with kind permission from the composer).

Table 6.2. *Praescio IV* event list sample

<table>
<thead>
<tr>
<th>Ev.</th>
<th>Trigger</th>
<th>Play Event Parameters:</th>
<th>THRU Event Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>T</td>
<td>P</td>
<td>Chan</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3a</td>
<td>-2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3b</td>
<td>-2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3c</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>6</td>
<td>5a</td>
<td>-4</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>5b</td>
<td>-2</td>
<td>-</td>
</tr>
</tbody>
</table>

*MIDI Sequences.* Like the event list, a series of short, pre-recorded sequences of notes and synthesizer control data serve as electronic "score excerpts" for the synthesizer. Unlike the event list, the sequences are extremely detailed, specifying precisely which notes to play, and how and when to play them.

The prerecorded sequences are stored as multi-track standard MIDI files (SMF, type 1 files). Individual sequences for each event are stored as separate tracks within the file. Counting each track as a separate sequence (which is indeed how they are used), there are a total of 73 short prerecorded MIDI sequences (although a few tracks are in fact duplicates of others).

---

67 A brief overview of MIDI is given in chapter 2.
The first MIDI sequence used in *Praescio IV*, sequence 3a, is extremely short, consisting of a series of five two-note chords. The sequence data contained in this file is shown in table 6.3.

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Duration</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>111000</td>
<td>Note 52 (E2), velocity 126</td>
<td>7160</td>
<td>214160</td>
</tr>
<tr>
<td></td>
<td>Note 40 (E1), velocity 126</td>
<td>7160</td>
<td>214160</td>
</tr>
<tr>
<td>213320</td>
<td>Note 50 (D2), velocity 126</td>
<td>9080</td>
<td>414400</td>
</tr>
<tr>
<td></td>
<td>Note 42 (F#1), velocity 126</td>
<td>9080</td>
<td>414400</td>
</tr>
<tr>
<td>414000</td>
<td>Note 46 (Bb1), velocity 126</td>
<td>101477</td>
<td>712477</td>
</tr>
<tr>
<td></td>
<td>Note 27 (Eb0), velocity 126</td>
<td>101477</td>
<td>712477</td>
</tr>
<tr>
<td>711477</td>
<td>Note 45 (A1), velocity 126</td>
<td>81340</td>
<td>912337</td>
</tr>
<tr>
<td></td>
<td>Note 38 (D1), velocity 126</td>
<td>81340</td>
<td>912337</td>
</tr>
<tr>
<td>911437</td>
<td>Note 40 (E1), velocity 126</td>
<td>141320</td>
<td>1214277</td>
</tr>
<tr>
<td></td>
<td>Note 28 (E0), velocity 126</td>
<td>141320</td>
<td>1214277</td>
</tr>
</tbody>
</table>

This same sequence of notes, rendered in standard musical notation, is shown as it appears in the score in figure 6.2.

---

68 Time in this MIDI sequence is given in a “measure | beat | tick” format used by most commercial MIDI sequencing programs. In this case, there are 480 ticks per quarter note beat, and four beats to a measure. Timings are therefore relative to the tempo, defined in the file header, but subject to variation on playback. Corresponding Note Off messages are given as “End” times, and durations are calculated as “beat | tick.”
Because MIDI is an internationally accepted standard protocol for synthesizer control now in use for more than 20 years, storage of sequence data in this format is reasonably secure, even if future realizations of Praescio IV are not MIDI-based. Problems with digital file storage and retrieval can be easily avoided by formatting (and possibly printing) data files in standard plain-text format, similar to the format given in Figure 4. Due to the sheer quantity of data, and the composer’s copyright, the complete set of MIDI sequences required for Praescio IV will not be included here in its entirety.

6.3.5 Event Processing

Score events in Praescio IV are processed in three distinct stages: input processing, “play” event processing, and “thru” event processing. Input processing controls the advancement through the event list sequence in response to MIDI input received from the foot switch trigger and the pitch tracker. The play event processor
controls the playback of SMF sequences for each event. The thru event processor controls the synthesizer colorization of the clarinet (passing tracked pitches as MIDI note messages “thru” to the synthesizer).

Input Processing. For each event in the list, a trigger condition must be met before the event parameters are sent to the play or thru event processing stages. Trigger conditions are either a signal from the footswitch (i.e., MIDI controller 64, value 127), or a specific note number received from the pitch tracker. The input processor must look ahead to the upcoming event in the list and set the appropriate trigger condition. When the appropriate trigger signal is received, the list of parameters stored for that event are sent out to the play and thru event processors.

Play Event Processing. The Play event processor interprets a list of playback parameters for each event in order to execute playback of MIDI data stored in the SMF sequences. The event list specifies a track name, a MIDI playback channel (1-16), transposition level (+/- n semitones), harmonization (+/- n semitones), velocity scaling (% of stored values), and tempo (% of stored tempo). Several sequences may be played back simultaneously, and the same sequence may be played in multiple simultaneous instances at various transposition levels, dynamic levels, and tempi.

Several sequences are used in more than one event, although playback parameters may be changed. For example, events 35 – 38 all use the same sequence data (“tr38,” or track 38—a rapid single-note sextuplet ostinato, with a gradual decrescendo). Values for
transposition (trans), velocity (vel), and tempo (tempo) control sequence playback, as shown in table 6.4.

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>play tr38 chan 8</td>
</tr>
<tr>
<td>36</td>
<td>play tr38 chan 8 trans 8 vel 0.6</td>
</tr>
<tr>
<td>37</td>
<td>play tr38 chan 8 trans -3 vel 0.8 tempo 1.1</td>
</tr>
<tr>
<td>38</td>
<td>play tr38 chan 8 trans -2 vel 0.5 tempo 0.9</td>
</tr>
</tbody>
</table>

Event 35 plays back the MIDI sequence “tr38” on synthesizer channel 8 with no alterations. Event 36 plays the same sequence a minor sixth higher (+8 semitones) with all velocity values scaled by 60% (x 0.6). Event 37 plays the sequence a minor third below the original (-3 semitones), with velocities scaled to 80% and tempo slowed by scaling durations to 110%. Event 38 transposes the same sequence two semitones below the original, cuts velocities by half (0.5), and shortens durations to 90% of their original values.

This synthesizer output is only roughly notated in the score in order to give a simplified version of the electronic accompaniment as a convenience to the performer. Although standard musical notation is not as precise as the MIDI sequence data itself, the rendering of this passage in the score provides an easily readable reference of the synthesizer activity, as shown in figure 6.3.
Play events are formatted in Pennycook’s most recent performance software for the custom-built Max external object playSMF. PlaySMF was created in 1993 by Pennycook, Dale Stammen, and Basil Hilborn for Praescio VI (for flute and interactive system) and was used in the 1995 version of Praescio IV performed on tour by Boisvert.\textsuperscript{69} The current software uses an updated and recompiled version of playSMF (as a Max4/MSP2 external object for Macintosh PPC computers) completed by Dale Stammen in February 2004.

\textit{THRU Event Processing}. Score Events that apply synthetic “colorization” (doubling or harmonization) to the clarinet are designated as “thru events” because they pass the clarinet pitch data “thru” to the synthesizer on MIDI channel 16, playing the synthesizer as if by keyboard control. Because note data sent to the synthesizer from the pitch tracker is mediated by the interactive software and controlled by parameters in the event

\begin{footnote}
\end{footnote}
list, pitch data can be altered or manipulated before it reaches the output. The thru event processor manages harmonizations by matching each incoming note with up to four pitches at variable transposition intervals, expressed in semitones (or more accurately, as MIDI note number offsets). The harmonizations are based on a list of variable settings controlled by the event list. Figure 6.4 shows the synthesizer simply doubling the clarinet at event 12 (sequence 11 continues playing):

Figure 6.4. Score example: parallel tracking of the clarinet by the synthesizer (event 12)

Event 29 is more complex. The clarinet G3 is harmonized by notes at intervals of 0 (G3), 1 (Ab3), -7 (C3), and -18 (C#2):
SUMMARY

On close examination, the interactive system used in Praescio IV can be broken into two main parts: a general-purpose MIDI data processing and synthesizer control system, and a set of data files (MIDI files and specially formatted event lists) that are specific to the piece and act as an electronic “score.” The entire system is controlled by real-time inputs from the performer, in the form of footswitch signals and pitch data collected by a pitch-tracking module. The main role of the system software is to interpret incoming control signals, parse commands from the event list, playback recorded MIDI sequences on cue, manage the colorization/harmonization of the live clarinet, and route MIDI messages to the appropriate synthesizer channels. A diagram of the complete processing system is shown in figure 6.6.
Input devices:

- Foot switch
- Pitch tracker
- Sustain pedal
- Volume pedal

“Score” files:

- SMF Sequences
- Event List

Event number:

Input Processor

MIDI cc 64:

MIDI note/vel:

MIDI cc 65:

MIDI cc 7:

“Score” files:

SMF Sequences

MIDI sequence data:

Play event parameters:

Thru event parameters:

Output device:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

|------------------| MIDI channels |------------------|

Synthesizer

Figure 6.6. Diagram of the Praescio IV interactive MIDI system

Implementation of Praescio IV for newer technology or for competing computer platforms could be a fairly straightforward process, so long as the technology used is capable of the input and MIDI file processing outlined above. However, any realization of this work will require a complete set of Pennycook’s SMF sequences in addition to the clarinetist’s score. These items are currently distributed in electronic format with the performance software upon request to the composer.
In 1991, I received a copy of Jonathan Kramer’s 1977 manuscript score for
Renascence from the collection of retiring University of Wisconsin clarinet professor
Glen Bowen, via my teacher, Dan C. Sparks. I eventually included a performance of this
work in its original version, along with a detailed discussion of the technical challenges
involved in its concert production, as part of my undergraduate project for honors in
independent study at Lawrence University. I presented two more performances of this
work in its original version in Baltimore in 1999, using “period” instruments found in the
storage rooms of the Peabody Conservatory. In November of 2003, I approached the
composer, inquiring about the possibility and desirability of a digital reconstruction of
Renascence. Kramer’s enthusiastic response led to the inclusion of Renascence in this
study.

7.1 HISTORICAL BACKGROUND

Renascence was commissioned by the University of Redlands and dedicated to
Phillip Rehfeldt and Barney Childs. The first performance was given in March of 1975
by clarinetist Keith Wilson with composer Jonathan Kramer operating the electronics.70
In 1977, due to the difficulty of accurately performing Renascence with a live tape delay
system, Kramer released a revised version for clarinet and tape only. The 1977 version

70 Jonathan D. Kramer, email to the author, January 5, 2004
combined the prerecorded tape from the original version with a prerecorded, ideal version of the tape delay part. A second revised version was released in 1985, which is essentially identical to the 1977 version, but with a higher quality tape part. In 1997, Bradford Garton, director of the Computer Music Center at Columbia University, created a digital implementation of *Renascence* that received its only performance to date in 1998 by clarinetist Jean Kopperud. Garton’s digital delay version is not generally available. The score and prerecorded tape are currently available on hire from G. Schirmer, Inc.

7.2 MUSICAL ROLE OF TECHNOLOGY IN *RENASCENCE*

*Renascence* is an eleven minute work and is musically based on the main principle of audio delay lines: a recorded signal is delayed a specified amount of time before playback begins. Kramer uses this principle to create a counterpoint between the live clarinet and its previous self. Delay feedback processing, in which the output of the delay is also routed back to its own inputs to create multiple repetitions, allows the music to build into a larger and larger texture. Each cycle of the delay line adds another voice to a growing thicket of sound.

*Renascence*, although it is for solo clarinet, was conceived as an ensemble piece with parts for the clarinetist, a technician, and a page-turner. The technician, performing from a mixing/control console, accompanies the solo clarinetist. The technician’s role is 1) to maintain the audio balance between the live clarinet, the delay system, and the prerecorded sound; 2) to execute changes in audio routing within the delay system during performance; and 3) to maintain balance of the delay system and its feedback loop during
performance. The page-turner is on hand to act as technical assistant as well as to turn pages.

Because the musical texture is built up from multiple repetitions of the clarinet line, pitch materials are confined to a six-note mode: E-F#-G-B-C#-D. A constant pulse of 200 beats per minute (in 2/4 meter) governs all musical activity and tempo must be adhered to strictly throughout. The rhythms are often syncopated, with wide interval leaps and displaced accents, giving the piece a bright, swing feel (the direction “quasi jazz ...” is used several times).

*Renascence* calls for two extended clarinet techniques. Multiphonics, made up of pitches from the six-note mode, anticipate chords created later on by the delay system. In the final section, long glissandi are layered through successive re-recordings in the delay. The longest glissando covers an octave and a fourth over a period of nine measures.

The delay system creates 34-measure cycles of repetitions, in which any material recorded into it repeats 34 measures later and is added to any new material recorded at that time. Therefore, large textures can be built in 34-measure increments over longer periods. In the opening section, after a short introduction, a sparse and jagged line is fed into the delay system (figure 7.1). The rests are gradually filled in during subsequent passes of the delay, in what the composer describes as a “hocket-like fashion” (figure 7.2). This gradually builds to a constant eighth-note texture, heard for the first time in its entirety at measures 341 (figure 7.3). In the final section, the same principle is used to layer successive long tones into block chords and to combine multiple glissandi at
different rates and pitches into complex blocks of sound, building to a climax at measure 875 (figure 7.4).

Figure 7.1. Score example: initial material recorded to the delay line

Figure 7.2. Score example: a new fragment layered with the first as it returns 34 measures later

Figure 7.3. Score example: continuous eighth-note texture built out of layered fragments
Despite the complexities of the electronics, *Renascence* is an appealing work for the performer and audience alike. The technical obstacles to a successful performance are formidable, but many of these obstacles are due to the limitations of the original technology. Newer implementations using streamlined and accurate digital delay and control systems solve many of the problems that plagued early realizations of this work and come closer to the intentions of the composer. Hopefully, new realizations of this work can more easily live up to the *Cincinnati Enquirer’s* impression of Phillip Rehfelt’s recording:

The main point about Kramer’s *Renascence* is that it is fun. There is only one clarinet on stage ... but by the time this piece starts cooking there are more than a dozen tape-recorded clarinet lines in the works, all swinging like Benny Goodman and doodling in other jazz-like rhythms on the same six notes.\(^71\)

In the 1974 version, Kramer outlines a role for two technical assistants: one at a mixing console to control the delay system and monitor sound levels, and a second to

turn pages for the clarinetist, and at one point remove the clarinetist's headphones (monitoring the necessary click-track). New realizations of this work may reduce the need for technical assistance somewhat, by automating certain control functions, but Renascence was conceived as an ensemble piece, and the role of the technician at the mixing console is critical to the musical success of the piece in performance.

7.3 ANALYSIS OF TECHNOLOGY COMPONENTS

The 1974 version of *Renascence* used a tape delay system for live processing of the clarinet sound. However, the tape delay system as it is specified in the score presents serious difficulties in accurately realizing the composer's intentions. This fact led the composer to create a subsequent version for clarinet and tape that leaves the interactive element out altogether.

However, the live delay processing was central to Kramer's aesthetic of the piece. An addendum to the technical notes in the 1977 version for clarinet and tape included the following notice.

The original version is to be preferred in cases where the proper equipment is available. When the new version is used, the following explanation should appear in the concert program or program notes: "*Renascence* was originally scored for clarinet, tape delay system, and prerecorded tape. In this performance a version combining the tape delay system and the prerecorded tape onto one tape is used in order to simplify otherwise complex electronics. The sound of the piece is theoretically identical in both versions."\(^{72}\)

---

\(^{72}\) Jonathan D. Kramer, *Renascence, for B-flat Clarinet, Tape Delay System, and Prerecorded Tape* (Jonathan D. Kramer, 1977), errata sheet regarding the 1977 version, included with the 1974 version of the score.
In my correspondence with the Kramer, he clearly indicated his preference for the live version: “I may have written that ‘The sound of the piece is theoretically identical in both versions,’ but this is not strictly true, since the ideal is to have the same clarinetist’s sound heard both live and on the tape.” This statement underscores the need for new implementations of Renascence using updated interactive technology.

Renascence could be easily adapted to digital technology. The composer provides extensive technical notes with the score, describing the tape delay system in conceptual terms rather than as settings for a specific machine. However, certain aspects of the electronics are described in terms that are specific to analog equipment. Some translation is required in order to adapt this work to digital or other technology. A digital implementation of Renascence would in fact represent a considerable simplification of the technological requirements of the piece, greatly reducing the logistical difficulties encountered in performances of the original 1974 version.

According to the composer, the basic theoretical concept of Renascence—the recurrence of previously played music combining with the present performance—is more important than the sonic side effects (analog distortion and machine noise) and visual spectacle (rotating reels and magnetic tape traveling across the stage between tape decks) of the original analog tape delay system. Therefore, a delay system that more

---

74 Ibid., January 4, 2004
accurately and effectively accomplishes the composer's musical goals is preferable to one that utilizes original or "period" instruments.

The basic elements of the system that must be reconstructed include 1) a sound reinforcement system including microphones, loudspeakers, and a matrix mixer with three inputs and three outputs; 2) a delay system with a long delay time and controllable feedback; and 3) a control system that manages signal gain between components, provides a click track to the performer (via head-phone), and makes discrete changes to the matrix mixer connections according to events notated in the score. In addition, a prerecorded tape constructed from clarinet sounds (based on an excerpt from the score) plays in the background. This tape may be reconstructed according to the method outlined in section 7.3.4, below.

7.3.1 Sound Reinforcement

The sound reinforcement requirements for *Renascence* include a microphone for pickup of the live clarinet into the delay system and two-channel loudspeaker playback for the delay output and the prerecorded tape. The delay system output is routed to the left speaker and the prerecorded tape is played back to the right speaker throughout.

The sound mixing system is used to control routing between microphone input and delay inputs and outputs, in addition to controlling loudspeaker levels for the audience. The mixer specified in the score is a three-input/three-output matrix mixer, with any of the three inputs instantly switchable to any or all of the three outputs.
Most standard mixing consoles used in live applications do not easily meet the requirements of Kramer's score. However, in a digital reconstruction of the delay system, the control and routing functions of the matrix mixer could be implemented in software, requiring only a standard mixer to handle microphone input and loudspeaker output. The control functions required of the mixer in the original version will be discussed separately (below).

7.3.2 Delay System

The original tape delay system required two open-reel tape decks. Sound was recorded on deck one and played back on deck two, after the tape had physically traversed the distance between the two tape machines. Feedback was controlled by routing the playback output of deck 2 back to the record input of deck 1, with recording level gain controlled by an analog mixing console. Figure 7.5 shows a diagram of the complete system.
Any reconstruction of the tape delay by other means will have to match two important features of the original tape-delay system: delay time and delay feedback functions.

*Delay Time.* The original tape delay system specified a total delay time of 34 measures, set by the physical distance between tape machines and tape speed. A digital delay would have to be set for the appropriate time interval, assuming enough system memory is available for the required delay buffer. With a meter of 2/4 throughout, and a tempo marking of 200 beats per minute, the delay should be set to 20.4 seconds, (or 20,400 milliseconds). The delay system should have two separate channels set to the same delay interval.
Delay feedback. Delay feedback must be routed and controlled separately for each of the two channels in the following manner: delay outputs 1 and 2 should each be assignable to either (or both) delay input 1 or 2; each channel must be turned on and off quickly without producing audible “pops” or “clicks;” fine adjustments also must be made to the amplitude levels for each channel to maintain the feedback signal as it circulates in the delay line and is combined with new layers from the incoming microphone signal. A technical assistant should control the feedback system should manually, as in the original version.

7.3.3 Control System

Control of the delay system must be coordinated very closely with the live performer and with the overall timeline of the piece. It is absolutely critical that the changes to signal routing within the delay system are executed precisely at the points indicated in the score.\textsuperscript{75} It is also absolutely critical that the performer stay synchronized with the delay system throughout the performance. Two elements of the electronic system have to do with synchronization and control: the click track and the signal matrix. The overall sequence of changes to the signal matrix, or to use terminology employed in more recent works for interactive computer music systems, “score events,” could be

\textsuperscript{75} In his May 28, 2004 email to the author, Kramer indicated that in practice (based on a 1997 digital implementation of Renascence with Bradford Garton at Columbia) changes to the signal routing should be placed slightly before the points indicated (“a tenth of a beat or so”), in order to avoid cutting off the attack portion of the clarinet sound.
thought of as a third element of the control system, especially if this sequence of score events is placed under automated software control.

*Click Track.* In order to accurately synchronize with the delayed sound, which makes its first entrance at measure 103, a click-track is necessary. As indicated in the score, this click track (quarter/half = 200/100 b.p.m.) is sent to the performer's headphone only, and should be unnecessary after measure 477. Setting up the click track in the original version was a very complex affair:

[Thirty-four] measures of click track can be counted and marked on the tape. Note that the click track is recorded over and over through the Tape Delay system; this must be done, even if a slight irregularity of beats occurs each 34 measures (to use a totally prerecorded click track would result in a small lack of synchronization building gradually into a major problem).\(^7^6\)

The difference between the original version and any new implementation using a digital delay is the absence of the "irregularity" produced by a tape delay system due to the slightly inaccurate physical placement of open-reel tape machines. Therefore, in a digital realization of the delay system, the click track no longer needs to be fed through the repeating delay on channel 2 as directed in the score. A simple metronome, completely external to the delay system, should be sufficient, although it is still critical that the clarinetist stays with the click-track in order to synchronize with his or her "reflection" 34 measures later.

---

\(^7^6\) Kramer, *Renascence* score, 4.
Signal Matrix. The technician’s line in the score features a 3 x 3 grid representing a
matrix mixer that can route any or all of three inputs to any or all of three outputs. Each
connection represented by the grid must be able to switch on or off instantly, with no
audible clicks or pops entering the sound system.

Figure 7.6. Matrix mixer for the 1974 version of Renascence

In the technical notes that preface the score, Kramer explains the matrix mixer
notation of the technician’s line:

An arrow up (↑) in one of the boxes of the grid means that the connection
between the indicated input and output channels should be opened; and arrow
down (↓) indicates that the connection should be closed; and empty box indicates
no change; a crescendo sign (<) in a box indicates that the output level should be
increased extremely gradually until the next grid.77

77 Kramer, Renascence score, 3
In a digital implementation of the system, outputs 1 and 2 (to channel 1 and 2 delay inputs) do not need to be physical outputs at all, but may instead be virtual connections to a software-based delay. In such a system, the only physical mixing connections would be input from the microphone and output to loudspeaker 1.

Regardless of the implementation, the grid notation used in the score is quite clear as to which logical connections to make at each point indicated. Whether these functions are operated manually in performance by a technician or automated according to the timeline, the notations found in the 1974 score are sufficient for guiding any realization regardless of the equipment used.

Score Events. The technician's part in the score contains directions for controlling the matrix mixer, notated using the box diagram described above. Many of these changes to the system must be performed very quickly and accurately in regards to timing, and some require changes to several connections at once, as shown in figure 7.7.
However, most of the matrix changes in the score are quite simple. This is especially true in the first section, in which the matrix mainly serves a bypass control for the delay input. Figure 7.8 shows the upper left arrow controlling microphone input to channel 1 of the delay system. Clarinet material enclosed in a box is not recorded into the delay.
Figure 7.8. Matrix used as a simple delay bypass

Because *Renascence* must be performed according to a strict timeline (synchronized by a click track), it would be fairly simple to automate the entire process of changing matrix settings according to events notated in the score. This would leave the technician free to monitor system levels and maintain gain equilibrium in the delay feedback system.

7.3.4 Prerecorded sounds.

The 1974 version of *Renascence* required a prerecorded tape in addition to the tape delay system. This tape played in the background, beginning at measure 35 and continuing through to the end. The performer does not need to coordinate in any way with the tape until measure 920, at which point the sound on the prerecorded tape provides an obvious cue.
The prerecorded tape is constructed from recorded clarinet sounds based on a short excerpt from the score. In the 1977 and 1985 clarinet-plus-tape versions of *Renascence*, the original prerecorded tape was mixed with the sound from the delay system. The result was a single tape to be used for performance in the event that the tape delay system could not be used.

In any new reconstruction of the delay system, the separate prerecorded tape will again be needed. The original tape was constructed nearly thirty years ago, using analog equipment and audio excerpts recorded by Phillip Rehfeldt. This tape was constructed according to a fairly simple process, but its overall sound qualities reflect the original equipment on which it was produced as well as the unique characteristics of Rehfeldt’s clarinet playing.

While the original tape still exists, it may be necessary or desirable to reconstruct the prerecorded tape. The advantages of doing so include matching the clarinet sound on the tape to that of the performer and matching the sonic qualities of the tape to the sound of the digital delay system (i.e., elimination of analog distortion).

Through our recent correspondence, Kramer has outlined the method for reconstructing the prerecorded tape, which I will relate here. The entire tape is based on six excerpts recorded by the clarinetist. These excerpts are then used to create a drone and a series of loops. These elements are then layered before being processed with a time-varying filter.

---

78 Jonathan D. Kramer, email to the author, March 17, 2004
Recorded Excerpts. First, the clarinetist must record six short excerpts, shown in Figure 7.9. The first two excerpts are long tones used to create a drone that is heard from the beginning to the end of the tape. The last four excerpts are slight variations on the music from measure 920 – 930.
Figure 7.9. Excerpts used to construct the prerecorded tape
Drone. The drone is constructed using long-tone excerpts 1 and 2 only. Through octave transposition and looping of the recorded long tones, the drone consists of four notes, sustained indefinitely. Excerpt 1 (C-sharp) is included in the drone transposed down one octave, and excerpt 2 (F-sharp) is included in the drone at its original pitch and again one and then two octaves lower.

![Drone](image)

Figure 7.10. Drone for the prerecorded tape

Transpositions are obtained by altering playback speed. In the original version, tapes were played back at half speed to achieve transposition of the tone down one octave. In the case of the lowest note in the drone, this process was repeated a second time to achieve transposition two octaves below the note recorded. This same effect can be achieved digitally by altering playback rates or by re-sampling.

<table>
<thead>
<tr>
<th>Note</th>
<th>Excerpt</th>
<th>Transposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>f#</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>c#</td>
<td>1</td>
<td>One octave below</td>
</tr>
<tr>
<td>F#</td>
<td>2</td>
<td>One octave below</td>
</tr>
<tr>
<td>F#1</td>
<td>2</td>
<td>Two octaves below</td>
</tr>
</tbody>
</table>
The drone loops should be constructed so that their loop points are masked and the tone is as steady as possible. The duration of the drone should span from measure 35 to at least measure 1037, approximately 10 minutes.

**Loops.** A second layer of the prerecorded tape is constructed using excerpts 3 – 6. In the original version Kramer looped the four excerpts, placing the loop points at the onset of the first note and preserving the last eighth-note rest. He then played all four loops backwards simultaneously, synchronizing their start points to end of the second eighth of the excerpt. Because the loops are different lengths, the loops gradually move out of phase with one another. While recording the output of the four loops, Kramer applied a band reject filter tuned to the middle of the audible frequency range (around 8 kHz), gradually widening its bandwidth from zero (no filtering) to the point where all sounds from the loops disappear. This process was timed to last from measure 919 to measure 35, where the tape enters in the score. The resulting tape was then reversed, so that the recorded loops now played forward, emerging gradually from the filter, then coming into phase at the end, with all four voices in perfect unison. This tape was then superimposed over the drone with the end point of the loops placed exactly at measure 920, an important cue point for the clarinetist, as shown in Figure 7.11.
This process could be recreated using digital audio editing tools and automated filter processes. The steps taken would be essentially the same, although the required editing would be much easier to achieve and would enable greater precision. A digital reconstruction would differ from the original mainly in the absence of analog tape distortion. This is the same effect a digital reconstruction would have on the live delay system. Therefore, a reconstruction of the prerecorded tape would more closely match the sound of the live system.
7.4 SUMMARY

Renascence is an example of a work that is composed according to general principles of live electroacoustic interaction but realized using available equipment. The equipment available at the time of composition was insufficient to consistently or conveniently realize the intentions of the composer. Thirty years later, sufficient tools are commonplace and new realizations of this work could be easily constructed using any number of strategies and systems that would achieve the degree of accuracy and system control needed for a successful performance.

A reconstruction of Renascence using current digital technology has the advantages of precise synchronization, higher audio quality, and convenience for the performer. For some musicians, there may be a certain romance lost by abandoning the visual and sonic aspects of the original analog system. However, the composer considered the visual spectacle and the effect of analog distortion to be secondary to the accurate synchronization of the live clarinet with periodic reflections of itself created by the delay system. As Kramer explained to me concerning a digital version produced at Columbia University in 1997:

It is true that actually seeing a tape delay system in operation has been interesting for audiences, but I feel this aspect is secondary to an authentic sonic experience.

[The] digital version is now my preferred version, even if the sound of re-recorded clarinets is not the same as in the original analog tape-delay version.  

Therefore a digital reconstruction is more likely to achieve the musical goals of this work than any reconstruction using “period” instruments.

---

Jonathan D. Kramer, email to the author, January 4, 2004
In 1996 I presented a master’s degree recital in computer music at the Peabody Conservatory in Baltimore, Maryland. Cort Lippe’s *Music for Clarinet and ISPW* was a central work on that program, which consisted of four interactive electroacoustic works for clarinet. Lippe provided the ISPW hardware and software, and he set up and monitored the system during the performance. My recital commentary, submitted in partial fulfillment of requirements for the master of music degree in computer music performance and concert production, described in detail the process of preparing for a performance using the original equipment. This commentary did not, however, attempt to describe the complex inner workings of the interactive system, which is the purpose of this chapter.

8.1 HISTORICAL BACKGROUND

Cort Lippe composed *Music for Clarinet and ISPW* at the Institut de Recherche et Coordination Acoustique/Musique (IRCAM) in Paris and at the Center for Computer Music and Computer Technology, Kunitachi College of Music in Tokyo, Japan, where it was commissioned. Clarinetist Esther Lamnek gave the first performance in March

---

1992 and has since recorded it for the Centaur/CDCM label.\textsuperscript{82}

The original version, for IRCAM Signal Processing Workstation (ISPW), required a NeXT computer (now obsolete) and special sound processing hardware (the ISPW sound card). A version of Miller Puckette’s Max software for the NeXT operating system controlled the synthesis and signal processing functions of the ISPW.\textsuperscript{83} At the time of its composition, the live audio signal processing featured in *Music for Clarinet and ISPW* was beyond the capacity of common desktop (Macintosh or Windows) computer systems. No commercially available effects processor offered the flexibility or computing power required for a realization of this piece.

In 1999 Lippe created a new implementation of the interactive electronics using Max/MSP for Macintosh G3 (or later) computers. In this version all real-time signal processing previously performed by the ISPW hardware is now handled by the main processor of a relatively inexpensive desktop or laptop computer. As of spring 2004 Max/MSP has been released for the Microsoft Windows computing platform. Therefore, Lippe’s latest software version is at least theoretically cross-platform compatible with either Windows or Macintosh computers.

However, cross-platform compatibility of the software is not the same as universal access. Users of competing real-time audio processing systems such as SuperCollider or Kyma are still left out. Furthermore, the future prospects for this work


still depend on regular updates of the performance software. While the original version was limited to performers with access to an ISPW/NeXT system (and often the composer’s direct participation), the latest version is now limited to users of Max/MSP. While this undoubtedly improves the prospects for diffusion of this work among performers, it does not guarantee its existence beyond the current realization. The following technical analysis of the performance system is intended to guide new implementations of Music for Clarinet and ISPW using alternative technology.

8.2 MUSICAL ROLE OF TECHNOLOGY IN LIPPE’S MUSIC FOR CLARINET AND ISPW

Lippe describes the relationship between the live clarinet and the interactive computer music system:

All the sounds used in the electronic part come from the composed clarinet part, and are transformed by the computer in real time during the piece. Thus, the musical and sound material for the instrumental and electronic parts are one and the same. The instrument/machine relationship is neither a dialog nor a duo. Musically, the computer part is not separate from the clarinet part, but serves rather to “amplify” the clarinet in a multitude of dimensions and directions.\(^{84}\)

*Music for Clarinet and ISPW* is an eighteen-minute *tour de force* for computer-enhanced clarinet. The sounds created by the computer system are strikingly unlike the sound of the acoustic clarinet, though they are generated from it. These effects are achieved by digital signal processing (DSP) routines that alter the source sound in complex ways, often bending the normal rules of acoustics by directly modifying the

\(^{84}\) Cort Lippe, *Music for Clarinet and ISPW*, program notes accompanying the score, 1992.
sound's harmonic components. The effect of *Music for Clarinet and ISPW*, which combines an often-aggressive clarinet part with extraordinarily complex computer-generated sound, is a breathtaking experience not normally achieved in solo literature for the clarinet.

While *Music for Clarinet and ISPW* is primarily a solo work for electronically "extended" clarinet, Lippe advocates an important performance role for a technical assistant. The technical assistant is required to monitor the system during performance, advance through system changes manually if necessary (as described in section 8.3.3 below), and most importantly, manage the sound system and the balance between the clarinet and the computer-generated sound. Lippe compares the role of the sound technician "to the work a conductor does for balance, color, etc."^^

8.3 ANALYSIS OF TECHNOLOGY COMPONENTS

The computer music system required for *Music for Clarinet and ISPW* can be separated into two basic parts: a general-purpose interactive synthesis and signal processing instrument, and a set of sound sources and system instructions that are specific to the composition. Therefore, the same synthesis and DSP system could (theoretically) be adapted for use in other musical works. In fact, Lippe's *Music for Piano and Computer* is based on the same system, with a few modifications. While the computer part does indeed function as a sonic extension of the clarinet, its structure can also be seen as following the familiar model of "instrument" and "score."

85 Cort Lippe, email to the author, June 24, 2004
Lippe’s interactive synthesis and DSP system combines live sound and prerecorded audio samples to generate complex audio effects. Within this software “instrument,” a series of discrete modules for sound processing or synthesis may be combined in any order for layered effects. This system is controlled by multiple parameter variables that can be changed in real time (i.e., while the system is still running, without the need to recompile software code or restart the system). In performance, changes to these system parameters occur at numbered “event” points, and the complete sequence of parameter changes for each event is stored in a database, or event list, which functions as an electronic “score” for the computer instrument.

Lippe’s piece is extremely complex, both theoretically and technically. The Max/MSP environment is a graphical programming language that makes interconnections between its parts explicit and is therefore a tremendous aid in understanding the operation of the program. However, Lippe’s software was written for performance functionality, not as a theoretical model for analysis. Therefore it presents some formidable challenges to the researcher trying to describe what is going on “under the hood.” The following is an attempt to thoroughly classify and explain the functions and control parameters of each component in Lippe’s interactive system.

8.3.1 Sound System and Necessary Hardware

For all the complexity of the interactive software system, *Music for Clarinet and ISPW* requires a relatively simple hardware setup. A minimal performance system will include a microphone connected to the audio interface (analog to digital converter, or
ADC) of a computer powerful enough to execute the necessary real-time digital signal processing routines. Output from the computer’s sound card (digital to analog converter or DAC) is routed to a stereo pair of loudspeakers. A simple control device, such as a MIDI foot pedal or the computer keyboard, is needed to advance the system through a series of pre-programmed events. Lippe recommends additional microphones used solely for sound reinforcement (i.e., not routed to the computer for processing) and the application of a mild artificial reverb to the live clarinet in order to balance its sound with that of the computer system. This basic setup is shown in figure 8.1.

![Diagram of sound system and control hardware](image)

Figure 8.1 – Minimal sound system and control hardware

8.3.2 Sound Sources

*Music for Clarinet and ISPW* is based on the transformation of the clarinet sound. A series of pre-recorded clarinet samples, drawn from the score, provide material for complex audio effects and synthesis techniques. Live sound picked up by the
microphone is transformed in several ways, and also provides a source of control information for the interactive system.

*Microphone input.* The live clarinet sound is picked up by a microphone and processed for two distinct purposes. First, it is used as an audio source for signal processing modules that transform the live sound of the clarinet in various ways. Secondly, information about the clarinet sound (i.e., pitch and amplitude measurements attained through real-time analysis) is used to control various parameters of synthesis algorithms. The choice of microphone is left to the performer's discretion, taking into consideration the need for an acceptable signal for both pitch tracking and effects processing while avoiding picking up too much ambient sound.

*Prerecorded Samples.* In addition to the live sound from the microphone, eight ten-second audio samples are used, each of which is a prerecorded clarinet excerpt from the score. Lippe provides the necessary sound files with the current software realization. A clarinetist could also re-record them using score excerpts, if so inclined. The score excerpts for each of the prerecorded samples are shown in figures 8.2–8.9.

![Figure 8.2. Score example: sample 1 (section I, events 5–7)](image)
Figure 8.3. Score example: sample 2 (section I, events 8 – 10)

Figure 8.4. Score example: sample 3 (section I, events 11 – 13)

Figure 8.5. Score example: sample 4 (section I, events 14 – 16)
Figure 8.6. Score example: sample 5 (section II, event 8)

Figure 8.7. Score example: sample 6 (section I, events 3 – 4)

Figure 8.8. Score example: sample 7 (section I, events 9 – 10)
Each sample file must be precisely 10 seconds (10,000 milliseconds) in duration, regardless of the actual length of the notated excerpt. In most cases the sample must be truncated to conform to the 10-second limit. Sample 5, which may be considerably shorter than ten seconds, should be filled out to 10 seconds with recorded silence.

8.3.3 Control Sources

The interactive system is controlled from several sources simultaneously, in the form of system parameters stored in score-related data files, real-time analysis of the clarinet pitch and amplitude, embedded algorithmic processing modules, the graphical user interface, and (optional) score-following algorithms.
Event list. The system is controlled by multiple variables, each of which can be changed individually during performance. Numbered cue points in the score refer to system events in which multiple system parameters may be changed according to a sequential list. Each line of text in the list serves as a command to the software system assigning a new value to a particular variable. Variable values in the event list may also trigger complex automated processes embedded in subprograms within the system software.

The event list is stored in an external database loaded by the software into an event cue. Variables are set as a block unless preceded by an integer, which initiates a time delay in milliseconds for execution of the next event or block of events. As shown in figure 8.10, lines 1 – 7 are executed immediately. Lines 8 and 9 are executed after a one-second (1000 millisecond) delay. Lines 10 – 12 are executed after a further one-second delay, and lines 13 – 15 one second after that. Each line contains one named variable followed by its new value or array of values.

```
ptof 0;
spatinc 4;
spaton 1;
ttoh 127;
htoh 107;
hdel 60;
ptoh 127;
1000 which2_table 3;
play-rand-pit2 7600;
1000 which2_table 2;
play-rand-pit2 4500, 8300 1000;
play-rand-pit2 8000;
1000 which2_table 4;
play-rand-pit2 8300, 5000 1000;
spatinc 55;
```

Figure 8.10. Event list excerpt: section III, event 14
Music for Clarinet and ISPW is divided into five sections, with approximately 10 to 30 events per section totaling some 2,225 lines of code in the event list. Therefore, I will not reproduce the entire event list here. Variable names used in the event list and in the system software are shorthand terms created by the composer specifically for this work. They are not in any way standard signal processing terms, and therefore, a key is required to understand the meaning of each variable and its intended effect on the relevant synthesis and signal processing routines. Each variable and its range of values will be explained below with the processing module in which it is used. A summary of all variables used in the piece is included in appendix D and variable names given throughout this text will appear in italics (e.g. \textit{hfreq}).

Many of the control variables found in the event list use value ranges based on the MIDI specification. Therefore, a number of pitch or amplitude parameters are expressed as MIDI note or velocity values between 0 and 127. These variable values are often converted within synthesis/DSP modules to frequency or amplitude values. In the case of MIDI pitch values, Lippe extends the standard MIDI note range to incorporate microtonal variations. These pitch variations are expressed in cents, appended to the MIDI note number by the operation: MIDI note number * 100 + microtonal variation in cents (0 – 99). For example, a pitch that is a quartertone above middle C (MIDI note 60) would be expressed as 6050. For the remainder of this discussion, this system of pitch notation will be referred to as "MIDI+."
Pitch Tracking. The incoming signal from the clarinet microphone is continuously analyzed for pitch content. Two pitch variables are generated: pitch-track-out (as MIDI+) and continuous_pitch (as a frequency in Hertz). Lippe currently uses the Max/MSP fiddle~ object (based on Fourier analysis) to track clarinet pitch. The goal is to reliably identify the pitches being played by the clarinetist for purposes of automated score following and to provide continuous monitoring of the clarinet pitch for use as a control value in several signal-processing routines. Alternate strategies for pitch tracking would in no way alter the essential functionality of the computer system, as long as the requirements outlined above are met.

Envelope following. The envelope follower tracks the volume contour of the incoming clarinet signal from the microphone. The amplitude envelope of an audio signal is a chart of its signal strength (volume) from its initial attack through its sustain and eventual release. In one particular synthesis/processing effect (frequency/amplitude modulation) the continuous amplitude envelope of the clarinet signal is used to directly control signal gain of multiple sound-generating oscillators.

Automated processes. Several score events trigger manipulation of various sound-processing modules according to complex algorithmic processes. These control algorithms will be described in more detail in the contexts of the synthesis/DSP modules upon which they primarily act. Synthesis/DSP modules that are affected by algorithmic processing include the sampler, frequency shifter, harmonizer, and spatializer.
Graphical User Interface (GUI). The front panel of the current software implementation of *Music for Clarinet and ISPW* includes direct access to all of the variable parameters for the DSP modules and the audio signal routing network. Most of these controls are provided as a convenience for testing or for direct intervention during performance if necessary.

Most importantly for performance, the graphical user interface (GUI) allows an assistant to advance through the event list manually, by mouse or keypad actions. Additional inputs may be added for performer control of the system, using a foot pedal or similar triggering device.

The GUI also provides a convenient method for setting up the system and resetting it immediately prior to a performance. The details of the various initialization and safety functions incorporated into the GUI and control system are specific to the current software platform and implementation, and therefore are not essential to an understanding of the musical use of interactive electronics in this work.

Automated Score Following. The NeXT/ISPW version of *Music for Clarinet and ISPW* relied on score-following algorithms developed by Miller Puckette to advance through the event list in synchronization with the clarinetist's performance. The score-following system was intended to avoid the need for system control by means of foot pedals or computer keypad control, since the computer would follow the performer and execute events in the list automatically at the appropriate points in the score. The computer system accomplished this task by comparing incoming note data from the pitch-tracking
module to a simplified electronic version of the score. Puckette and Lippe thoroughly
describe the algorithms used for score following in a paper given at the 1992

Lippe strongly recommends against using the score-following feature in the
current Max/MSP implementation, though it is fully functional. Instead, the composer
advocates manual advancement of events during performance by a technical assistant.
This arrangement adds an element of ensemble performance to what would otherwise be
a completely solo work. The role of the technician in this regard is minimal, but it is
critical to the success of the work since events must be advanced in time with the clarinet
performance.

Whether the original ISPW score-following algorithm is used or a newly invented
strategy is followed, the same basic principle should apply: the clarinetist’s performance
is compared to a stored version of the score, and events are cued according to the player’s
progress through the piece.

8.3.4 Synthesis and Signal Processing

Lippe’s signal processing instrument is highly complex and generates a number of
unusual audio effects. The algorithms that implement these effects are based on simple
techniques and standard principles of digital signal processing, such as amplitude or
frequency modulation, filtering, and delay lines. However, Lippe tends to layer and
interconnect these techniques in ways that defy easy description. Therefore, full
technical specifications for each of the required signal processing components are given
in the form of block diagrams, found in appendix C. The following is a brief description
of each component in terms of its basic audio processing technique, its general effect on
an incoming audio signal, and the functions and parameters of its variable controls.

*Sampler.* The sampler plays back prerecorded audio from a set of stored audio files. It
can play up to sixteen voices simultaneously (eight per channel) and allows for dynamic
control of playback speed (affecting pitch) and direction, glissando (through accelerating
or decelerating playback speed), amplitude envelope, and onset time within the file. The
following table shows the variable parameters for controlling sample playback. Because
multiple sampler “voices” can play at once, there are two sets of named variables for each
parameter per output channel (four named variables for pitch, four for velocity, etc.;
actual variable names are given in appendix C). Separately named variables can be
assigned new values individually (i.e., variables “pit1” and “pit2” would store pitch
values for two separate notes that may sound simultaneously).
Table 8.1. Variable parameters for sampler playback

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitch</td>
<td>Transposition. Middle C (60) = normal pitch. Transposition is expressed as MIDI note + cents (e.g. 6050 = quarter tone above middle C). Playback is initiated when a new pitch value is received.</td>
</tr>
<tr>
<td>Velocity</td>
<td>Amplitude scaling. Sent as MIDI values (0-127). Playback volume = velocity * .01</td>
</tr>
<tr>
<td>Onset</td>
<td>Start time. Playback is offset from the start of the file in milliseconds. Also controls forward/backward playback.</td>
</tr>
<tr>
<td>Attack</td>
<td>Envelope attack. Sets time in milliseconds for volume ramp from 0 to amplitude set by velocity.</td>
</tr>
<tr>
<td>Decay</td>
<td>Envelope decay. Sets time in milliseconds for volume ramp from playback amplitude to 0.</td>
</tr>
<tr>
<td>Gliss</td>
<td>Pitch glissando. Value sets glissando effect. Playback rate accelerated or decelerated to create pitch shift.</td>
</tr>
<tr>
<td>Gliss time</td>
<td>Glissando time. Time in milliseconds for gliss to last.</td>
</tr>
<tr>
<td>Sample</td>
<td>Sample table. Number (1-8) of the sample to play.</td>
</tr>
</tbody>
</table>

Granular Sampling. Three sub-routines control sample playback according to algorithmic processes, using a technique known as “granular sampling.” The basic concept of granular sampling is described by the composer as:

... the production of a multitude of short sounds (grains) consisting of a waveform, to which an amplitude envelope has been applied, at a specified frequency and amplitude. These grains of sound, produced at a high rate of speed, are usually overlapped with neighboring grains in order to produce a certain density and continuity of sound.\(^\text{87}\)

Compositional variables used by Lippe in his granular sampling algorithms include the waveform (the specific recorded sound from which to sample), amplitude envelope, peak amplitude, pitch (controlled by playback speed of each individual grain),

---

onset into the stored sample, grain duration, grain density, and grain overlap. Sound file onset time for each grain is critically important to the sound produced, and grains may be played in sequential order corresponding to the normal linear progression of the sample’s timeline. Alternatively, grain onset times may be chosen in non-linear or even random sequences. Furthermore, onset times that follow the normal timeline need not correspond to the normal playback rate. Lippe uses the term “precession” rate to describe the sequence of onset times for each grain relative to the normal timeline of a sample. The precession rate therefore describes the rate of progress of granular playback through a stored sound file, which may be faster, slower, or the same rate as normal playback. It may also remain stationary, continuously sampling from the same point in the file. This technique allows both pitch and playback speed to be treated completely separately.

Granular sampling can be used for “time-stretching” a recorded sound in the following manner: 50 millisecond bursts of sound (grains) are played back from a stored sound file every 40 milliseconds (resulting in a 10 millisecond overlap between grains). If the onset time of each grain is progressively 10 milliseconds (the precession rate) further into the file from the start point, it will take four times longer to progress through the file than normal playback speed. Pitch is unaffected because each individual grain is played at the normal speed, but overall playback speed is radically altered.

Lippe controls granular sampling with three separate subroutines: Trevor (named for the British composer Trevor Wishart), PLAY-RAND, and Trevor_back. Trevor is a straightforward implementation of Lippe’s granular sampling algorithm for time

---

88 Ibid., 4.
stretching. Playback may be initiated at any one of five points (0, 2, 4, 6, or 8 seconds into the sample table) by setting a trigger message for one of the variables `t-start1, t-start2, t-start3, t-start4, or t-start5`. Pitch transposition of the grains is set by `t-transpose` (MIDI + cents) and precession rate is set by `t-precess` ($0 - 238, 127 = \text{approximately normal playback speed}$). Grain duration is fixed at 50 milliseconds and velocity (determining peak amplitude) is fixed at 75 (on a scale from $0 - 127$). Grain onset (and therefore grain overlap as well) is not directly controlled by a system variable, but is instead controlled by a repeating trigger with its timer randomized from 10 to 19 milliseconds. Values generated by the granular synthesis engine are used to play short segments of audio from the stored sample tables by sending out values for the sampler variables described in Table 8.1 (above).

*Trevor_back* is essentially identical to *Trevor*, except that its precession moves backwards through the sample file. Start points are given at 2, 4, 6, 8, and 9.7 seconds into the file. Its variables are functionally the same as those of *Trevor* and are named in a similar manner. The full list of *Trevor_back* variables will be included in the complete table of system variables found in Appendix D.

The *PLAY_RAND* module accomplishes granular sampling in the same manner as the *Trevor* modules, with the exception that stochastic (random) processes control the parameters for sound-grain production. Two granular sampling units are placed in parallel. For each unit, grain pitch fluctuation, onset, duration, and overlap/grain density are controlled by a series of pseudo-random number generators. Pitch transposition, grain amplitude, grain envelope shape, and sample table number are set by event-list
variables. The first granular sampling unit has an additional control for producing randomized pitch glissandi within each grain. Full specifications for this sub-program will be included with the block diagrams in Appendix C.

Additional algorithmic processes are used to control the sampler at specific events in the score. In Section I, events 12 – 16, note values from the pitch tracker are used to control the Trevor and Trevor-back granular sampling modules. The algorithm, triggered in the event list by the variable ichgate (1 = on, 0 = off), is shown in table 8.2, below.

IF the value of ichgate is 1, THEN
SET the value of tt02, sto2, tt04, and sto4 to 127; (full volume)
IF the value of pitch-track-out is between 50 and 52,
THEN {
TRIGGER spatXY spatialization algorithm;
SET b-precess value to [pitch-track-out * b-precess-val];
SET b-transpose value to [pitch-track-out - 3];
TRIGGER granular sampling module Trevor-back with onset value set by brevor-onset;
}
IF the value of pitch-track-out is between 76 and 79,
THEN {
SET t-precess value to [pitch-track-out * t-precess-val];
SET t-transpose value to pitch-track-out;
TRIGGER granular sampling module Trevor with onset value set by trevor-onset;
}

Figure 8.11. Algorithmic control of granular sampling

Similar sampler control algorithms are used in Section I events 1, 5 – 6, 5 – 10, and 9, Section II events 11 and 26, Section III event 23, Section IV event 3, and section V events 3 – 6. These 8 additional algorithms will be fully explained in Appendix C.

Harmonizer. Pitch is shifted a fixed amount using a specialized implementation of varying-time delay. Variable parameters include delay time (hdel, in milliseconds), pitch
(hfreq, 0 – 127), modulation depth (hwindl, 0 – 127), direct signal output amplitude (diramp, 0 – 127), harmonizer output amplitude (hamp, 0 – 127), and final output feedback to the harmonizer input (htoh, 0 – 127).

Two delay lines are used in parallel and the delay time for each is incremented according to a linear function from its start point (hdel) to an offset (hwindl) at a given speed (hfreq).\(^8^9\) The delay-time increment ramps are phase-offset from each other by 180 degrees: as the first ramp reaches the offset value, the other is at the halfway point. The output of the two delays are then cross-faded at the rate set by hfreq to produce a smooth alternation between the two delay lines. Alternation between the two delay lines creates the illusion of a constantly increasing (or decreasing) delay time and therefore a constant and stable interval of pitch shifting.

Pitch shifting in this manner is not dependent on the initial delay time. The critical element is the increase or decrease in delay time at a constant linear rate. The continuously changing delay time shifts the pitch by effectively speeding up or slowing down playback of the delayed signal. Pitch offset from the original signal is a function of both speed (hfreq) and offset (hwindl) of the delay modulation. Increasing the delay time lowers the pitch, while decreasing the delay time raises the pitch.

Amplitude of the input signal and the output of the harmonizer are controlled separately (by diramp and hamp respectively) and combined on output, allowing for control of the mix between the original signal and its harmonization. The output signal

\(^{8^9}\) Note that this type of delay modulation is used in Thea Musgrave's Narcissus, with variable parameters for delay time and speed/depth of time modulation.
may be fed back into the harmonizer input, set by the system variable \( htoh \), \((0 - 127)\), to shift the sound a second time by the amount specified by \( hwindl \) and \( hfreq \).

**Reverb.** The input signal is fed into a series of fixed short delays to create artificial reverberation. Variable feedback of the reverberation creates the illusion of a larger or smaller space, depending on the amount of feedback (the amplitude of the output signal fed back into the reverb input). Input values for \( Rgate \), \( Revfb \), and \( Rout \) control input level, feedback level, and output level respectively. Values for each variable are in the range of \( 0 - 127 \). \( Rgate \) and \( Rout \) scale these values between 0 and 1 exponentially (as described in the equation included with the Reverb diagram in Appendix C). Values for \( Revfb \) are scaled linearly from 0 to 0.5 (values from 0 to 127 are divided by 254).

“Infinite reverb” effects are achieved by closing the input \((Rgate = 0)\), and setting feedback to 1.0 \((Revfb = 254)\), trapping sound in the reverb system and looping it indefinitely.

**Noise modulation.** The input signal is fed through a series of short delays, each of which is amplitude modulated by a set of eight variable speed noise wavetable oscillators.\(^{90}\) The input variable \( fnois \) controls the frequencies of the noise oscillators. Values for \( fnois \) \((0 - 127)\) are scaled slightly differently for each oscillator, but fall within an approximate

---

\(^{90}\) A wavetable is a series of digital audio sample values describing a single cycle of a waveform. A wavetable oscillator generates sound by storing the waveform in a buffer and looping its playback at a specified frequency. In Max/MSP, wavetable oscillators are most often implemented using the cycle~ object, which loads a 512-sample wavetable and takes variable arguments for frequency and phase offset. The default waveform for the cycle~ object is a simple cosine wave.
range of 0.016 – 29.4 Hz. At the lowest oscillator speeds, the effect is a slight fluctuation of the signal volume (a sort of random tremolo). At higher speeds, the signal begins to distort and additional frequency components are generated (due to the side-band effects of amplitude modulation). At the upper extreme, the signal resembles white noise, but retains its amplitude contour (i.e., rhythms are recognizable, but the tone quality is not).

**Frequency shifter.** Individual harmonic components of the input signal are shifted up or down by a specified frequency interval. The frequency shifter takes variable arguments for frequency ($f_{sh01}$, MIDI+ converted to frequency in Hertz) and amplitude ($f_{amp01}$, 0 - 127). Positive values for $f_{sh01}$ shift frequency components up, negative values shift them down. The second variable, $f_{amp01}$, controls the final output amplitude of the pitch shift module. Values from 0 to 127 are scaled from 0 - 1.

Frequency shifting in this manner radically changes the timbre of the sound since the shifted frequencies no longer bear the same harmonic relationship to the one another. For example, a signal with frequency components of 200, 400, and 600 Hz, shifted up 100 Hz, will create a tone with frequency components at 300, 500, and 700 Hz. In this example, the input signal consists of a tone and its first two harmonic overtones (defined as integer multiples of the fundamental) but the shifted signal contains frequencies that lie outside the harmonic series. The result is a radical “detuning” of the sound in which all its spectral components are shifted out of alignment. This effect is achieved using a form of amplitude modulation (AM) in which only one sideband component is used and
the original signal is filtered out. Complete details of the frequency shifter algorithm are found in the block diagram in Appendix C.

Frequency shift variable values are normally set directly by commands in the event list. However, in section I event 18 and section II event 1, frequency shift (and harmonizer) values are controlled algorithmically using live input from the pitch tracker. The event-list variable fsgate18 turns the control algorithm on when its value is set to 1. An fsgate18 value of 0 turns it off. The processing algorithm is as follows:

```
IF the value of fsgate18 is 1, THEN:
  IF pitch-track-out value is 77,
    THEN {
      SET fsh01 value to a random number from -2000 to -6499. 
    }
  IF the value of pitch-track-out is between 50 and 59,
    THEN {
      SET hfreq value to a random number from 40 to 63;
      SET ptoh value to 122; (clarinet signal vol. to harmonizer)
    }
  ELSE, SET ptoh value to 0.
  IF pitch-track-out value is between 71 and fstend18 value,
    THEN {
      SET ptos value to 127; (clar. signal vol. to freq. shifter)
    }
  ELSE, SET ptos value to 0.
```

Figure 8.12. Algorithmic control of harmonizer and frequency shifter

The musical effect of this algorithm is illustrated in figure 11 below. Clarinet pitches below C## are harmonized, while notes played above G## are frequency shifted.

---

Furthermore, each time the g above the staff is played, a new frequency shifter value is chosen at random from -2000 to -6499.

![Score example: algorithmic control of frequency shifter and harmonizer](image)

Figure 8.13. Score example: algorithmic control of frequency shifter and harmonizer

**Flange.** A flange (or flanger) is a standard signal processing effect most commonly associated with the electric guitar in Rock and Roll applications. The effect is more properly described as “phasing” and is based on a variable delay with a very short delay time (usually under 10 milliseconds). The effect is similar to a comb filter, in that the short delay produces phase cancellations at certain frequencies. When the delay time is modulated by LFO (low frequency oscillator), the cancelled frequency bands move up and down the audio spectrum, producing a kind of “swishing” sound.\(^2\)

---

A standard flange effect found on most commercially produced effects processors usually includes controls for input level, LFO speed, depth, feedback, and output level. Lippe’s flange module assigns the variables \( \text{flange-index}, \text{flange-speed}, \text{flange-del}, \text{flange-loop}, \text{flange-amp}, \text{and flange-master} \) (each with a range from 0 –127) to dynamically control these same parameters respectively. Full specifications for Lippe’s flange module, including scaling factors for input values, will be included in the block diagrams in appendix C.

Frequency/amplitude modulation. This module features an unconventional use of frequency modulation (FM) and amplitude modulation (AM) synthesis in that it transforms the live sound of the clarinet rather than simply generating a completely synthetic tone. Pitch and amplitude information derived from real-time analysis of the incoming clarinet signal is used to control FM and AM synthesis operators that in turn modulate the live clarinet sound. An envelope follower controls the amplitudes of several oscillators used for FM synthesis, based on the amplitude of the incoming clarinet signal from the microphone. Pitch data from the pitch-tracking module is used to set the carrier frequencies of two separate FM operators. Finally, the clarinet signal itself is used as a modulator signal.

This module contains two parallel FM synthesizers, \( FM1 \) and \( FM2 \). \( FM1 \) is a simple FM instrument. Its carrier is a 555 Hz cosine wave, modulated by a cosine wave with its frequency dynamically determined by the value of the system variable \( \text{continuous_pitch} \) (set by the pitch-tracker). The envelope follower controls amplitudes of
both the carrier and the final $FM1$ output. $FM2$ uses both FM and AM synthesis principles. The carrier frequency (also a cosine wave) of $FM2$ is set by the $continuous\_pitch$ variable from the pitch-tracking module. The modulator is the direct clarinet signal from the microphone. The output amplitude for $FM2$ is controlled by the envelope follower and then modulated by a 223 Hz cosine wave. The output of this AM operation is then modulated a second time by a 50 Hz cosine wave. The output of $FM1$ and $FM2$ are summed before output. Final amplitude for the entire module is controlled by the $fmam-master$ variable, which has a range of values from 0 – 127 and is scaled exponentially from 0 – 1.

The final output of the frequency/amplitude modulation module is mixed with the final output of the flange module. This combined signal is then mixed with the output of the frequency shifter. All three are therefore treated as a single module for purposes of internal signal routing and final output/spatialization, described below.

Signal Routing. The internal audio signal routing network of the signal processing and synthesis instrument is set up so that the signal input to each module is a customized mix of the outputs from all the other modules. For example, the reverb module can take as its input signal a combination of microphone signal, sampler outputs, noise, reverb, harmonizer, and frequency shifter, all scaled individually and controlled by variables set in the event list. Variables used to control final output levels for each module are $pto2/pto4$ (clarinet signal), $sto2/sto4$ (sampler 1), $tto2/tto4$ (sampler 2), $nto2/nto4$ (noise
modulation), $R_{to2}/R_{to4}$ (reverb), $hto2/hto4$ (harmonizer), and $fto2/fto4$ (frequency shifter).

**Spatializer.** The output of each DSP unit is placed within the left-right stereo field by setting its signal level for each DAC output channel. Each sound source is controlled by a separate variable, allowing for precise control of each sound element's stereo placement. The Spatializer also controls the final amplitude of the entire DSP section output sent to the DAC according to the value of the global variable $tgp$ (0 – 157, scaled exponentially). The DAC output is routed directly to the loudspeakers. Therefore, the values used to control the Spatializer determine the mix of computer-generated sound heard by the audience.

Input signals to the Spatializer are controlled by a set of variables, each with a range from 0-127 (based on MIDI controller values) scaled to control signal amplitude. These variables, in left/right pairs, are: $fio2/fio4$ (frequency shifter); $hto2/hto4$ (harmonizer); $R_{to2}/R_{to4}$ (reverb); $ nto2/nto4$ (noise); $tto2/tto4$ (sampler 1); $sto2/sto4$ (sampler 2); $pto2/pto4$ (clarinet from the microphone).

At certain score events, theSpatializer is controlled algorithmically. Left/right speaker placement of the final output from the sampler, noise envelope generator, reverb, harmonizer, and frequency shifter are selectively put under algorithmic control. Only the modules selected for algorithmic control are affected by the following three automated panning routines.
In section II, event 19 through two seconds after the onset of event 20 (controlled by \textit{spatXY-19}), sound output for all modules is set hard-left or hard-right at random intervals between one and six seconds, with both sides set to zero after a one-second delay. In section I, events 12 – 16, hard-left or hard-right placement of the computer generated sound is chosen randomly each time a note between 50 and 62 is played (clarinet lowest E to the E one octave above) and is detected by the pitch tracker.

Throughout the event list, the variable \textit{spaton}, with values from 0 – 4, is used to choose from among four automated cross-fade panning patterns. These cross-fade patterns are determined by a set of tables, each containing a sequence of 128 left-speaker and right-speaker amplitude values. A second variable, \textit{spatinc}, is used to control the speed and direction through which the tables are read. Values from 0 – 63 cause the tables to be read backwards, while values from 65 – 127 cause the table to be read forward. At the extreme ends (\textit{spatinc} values of 0 or 127) the table is read at a rate of one value every 20 milliseconds. Values approaching the mid-point of 64 become progressively slower. A value of 64 freezes the panning effect in place. A diagram of this effect algorithm, with a representation of the tables, is included in Appendix C.

8.4 SUMMARY

Lippe’s \textit{Music for Clarinet and ISPW} presents a very complicated test case for the type of analysis I advocate for the preservation and reconstruction of older interactive
electroacoustic works. The software is simultaneously a composition and a working instrument and was not designed for abstract analysis.

Despite the difficulties in comprehending not only Lippe’s compositional and synthesis algorithms (not to mention his source code) careful analysis shows that all of the processes employed in this work are based on commonly accepted principles of audio signal processing and algorithmic composition. Each effect or technique used in *Music for Clarinet and ISPW* can be found, at least in its simplest form, in one or more standard computer music textbooks. Therefore the problem is one of classifying each effect, breaking it down to its component parts, identifying its variable parameters (and their values), and describing the controls needed to operate this system during performance.

One further application of this analysis might be to re-engineer the software in a way that completely separates the synthesis and signal-processing instrument from all of the score-related data and processes. This result would be a very sophisticated general-purpose virtual machine and a transparent format for preparing score-related data. This would provide a clearer window into Lippe’s compositional technique, and may also provide a practical resource for composers interested in pursuing similar methods.
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 AND 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.

![Diagram of stage setup](image)

**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.\(^3\) 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.

---

\(^3\) 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).

![Diagram of Multi-tap delay line]

**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.

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.
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.

![Modulation Control Module](image)

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.
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.

![Diagram of Bypass Control Module]

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.
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"**

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
<th>FB</th>
<th>Mod</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2,</td>
<td>0.5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3,</td>
<td>1.0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4,</td>
<td>1.0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>5,</td>
<td>0.5</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>6,</td>
<td>0.5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7,</td>
<td>0.5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>8,</td>
<td>0.5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>9,</td>
<td>2.0</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>10,</td>
<td>2.0</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>11,</td>
<td>2.0</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>
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.

![Diagram of data routing between control and signal processing subprograms](image)

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.
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. His 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 AND 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).
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.
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.
Praescio IV - Clarinet and Interactive MIDI System

MONITORS AND CONTROLS

--- EVENT MONITOR/REHEARSAL CONTROL ---

Current Event #: 1
Start at: $0\rightarrow \ll$ (begin)

INPUT MONITOR

<table>
<thead>
<tr>
<th>Trigger:</th>
<th>Pitch Tracking:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$0\rightarrow \ll$</td>
</tr>
</tbody>
</table>

Current Event #:

<table>
<thead>
<tr>
<th>Note:</th>
<th>Velocity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>$0\rightarrow \ll$</td>
</tr>
</tbody>
</table>

Manual override (keyboard):

$T = $ trigger | $N = $ note | Spacebar = next event

Sustain: on/off
Volume: $\ll$

EMERGENCY MEASURES

STOP! VOL RESET

PERFORMANCE SETUP

1. Mic Input & Pitch Tracking
   - Audio input $\ll$
   - L/R Mic Input Levels

2. Load Standard MIDI Files
   - click here: Load SMF A
   - then here: Load SMF B

3. Set Up MIDI Controllers
   - click here: MIDI Controller Setup

UTILITIES

- Monitors
- Open Control Panels:
  - Play SMF Section A
  - Play SMF Section B
  - Pitch Tracker
  - MIDI Inputs
  - MIDI Outputs
  - Thru Controls
  - Info & Help
- Choose MIDI Output:
  - FastLane $\ll$

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 \textit{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 \textit{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 \textit{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 \textit{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 \textit{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.

\begin{table}[h]
\centering
\caption{Excerpt from \textit{coll} file 1: event control}
\begin{tabular}{c}
1, 1 0;
2, 1 0;
3, 0 50;
4, 0 64;
5, 0 59;
\end{tabular}
\end{table}
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](image)

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.

**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.

![Diagram of THRU Event control module](image)

**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.
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*

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

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.
Chapter 11
PROJECT SUMMARY

This project demonstrates a new model for the analysis and preservation of interactive electroacoustic works. Rather than updating or transcribing older electroacoustic works for new technology, this analysis seeks to describe in detail the features and functions of the required interactive electronic systems using a format that is independent of particular technologies, devices, or specialized programming languages. Using this approach, I have so far analyzed the interactive electronic systems for four works for clarinet. My analysis is based on a combination of sources, including examination of original equipment, reverse engineering of software, close examination of scores, supporting documentation, and available literature, and direct conversations with the composers and with several performers involved in the early history of these works.

In order to promote the widest possible accessibility, I have chosen the most technologically independent format possible: text and graphics on paper. Therefore, rather than simply creating new electronic realizations of these works in a common computer music programming language, I have chosen to describe the electronics using only score examples, mathematics, schematic diagrams of generalized audio processing algorithms, and plain English. I believe this to be the most useful format for preserving the viability of such works for three important reasons: 1) the analysis itself is not subject to technological obsolescence beyond that which affects any other printed document; 2) though perhaps technically challenging, it is readily accessible to a wider range of computer music performers, engineers, and scholars regardless of programming expertise
or specialization; 3) factual errors in the analysis (where they exist) can be more easily found and corrected by subsequent researchers working from primary source materials. The possibility of peer review and accessibility to a wider audience are two features of this analysis that I hope will make it useful as a model for documenting more such works.

I have made every effort to balance the need for readability with the need for technical precision and detail. My goals are two-fold: to provide on the one hand sufficient technical detail to guide engineers creating new implementations of the required electronic systems, and on the other hand to guide performers in their interpretation and implementation of the electronics in relation to the score by providing insight into the musical relationships between performer and machine. As demonstrated by the varying levels of complexity among the four works discussed, this balance is easier to strike in some cases than in others. Musgrave's *Narcissus* and Kramer's *Renascence* are relatively simple and straightforward, while Pennycook's *Praescio IV* and Lippe's *Music for Clarinet and ISPW* require enormous amounts of supporting technical information in order to describe adequately. I hope that this project demonstrates that it is indeed possible to document even extremely complex interactive electroacoustic works without having to refer to the functions of any one specific programming language, device, or system.

I have given real-world tests to my analyses of the works by Musgrave and Pennycook by presenting new realizations of them in concert. In the case of Musgrave's *Narcissus*, I had the benefit of modeling my performance software directly on the capabilities of the composer's original instrument (on loan from co-commissioner Wendy
Rolfe). In the case of Pennycook’s *Praesdio IV*, I had the benefit of direct supervision (via Internet) of the composer himself. Thanks to new insights into Kramer’s *Renascence* given in chapter 7, any clarinetist attempting this work may reconstruct the background tape using recorded excerpts of his or her own playing. Any new performance that takes advantage of this option will be more in line with Kramer’s ideal “to have the same clarinetist’s sound heard both live and on the tape.”

The four compositions chosen for this study were selected partially for the range of technical challenges they present, but mainly because of the musical quality they embody. Historical importance from a musicological standpoint was not the principal criteria on which I selected these pieces. Rather, my motivations for choosing them came directly from a search for a viable electroacoustic repertoire.

Kramer’s *Renascence* was a central part of my 1992 undergraduate honors thesis and lecture-recital at Lawrence University. My performance of that work, in its original version using “period” instruments, was a formative experience that attracted me to subsequent advanced studies in electroacoustic performance. The difficulty of performing *Renascence* in its original version finally led me to ask the composer for his thoughts on a possible digital reconstruction. His enthusiastic response to my query led to the inclusion of his piece in this document. My 1996 performance of Lippe’s *Music for Clarinet and ISPW*, featuring the original NeXT/ISPW system and the composer’s direct participation, was a central part of my 1996 degree-recital in partial fulfillment of a master of music degree in computer music performance and concert production at the

---

94 Email from Jonathan Kramer, May 28, 2004
Peabody Conservatory of Music. The accompanying recital commentary featured an exhaustive account of my concert preparations, but very little technical analysis of the work itself or the interactive system. I hope that the analysis presented in chapter 8 atones for this oversight.

The works presented here by Musgrave and Pennycook are ones I have not previously studied in depth. Therefore, I have given them the most attention by presenting not only a formal analysis, but also new realizations to be tested in performance. The musical uses of technology employed by both Musgrave and Pennycook were formative influences on my own approach to computer technology as an expressive tool for performance. Therefore, this project is a capstone to my academic work in the field of interactive electroacoustic music, spanning twelve years and three institutions.

My analysis is only useful if other performers wish to continue presenting these works. Preliminary evidence from concert programs as well as remarks by the individual composers and other performers with whom I have had contact indicate that there is sufficient interest in these works to justify their preservation. Therefore it is worth finding solutions to the problems of technological obsolescence that impede their accessibility to a wider pool of interpreters. Technological obsolescence is a barrier not only to individual performers, but also to the entire process of repertoire creation. I base this assumption on the theory that the value of a musical work to scholars, performers, and audiences is best assessed through repeated performance, interpretation, and listening.
CHAPTER 12
FUTURE DIRECTIONS FOR THIS RESEARCH

This project grew out of a personal search for interesting and performable works for clarinet and interactive electronics. However, the analysis and preservation of older electroacoustic works has become an issue of concern for many musicians interested in the long-term viability of such pieces in the face of rapid technological change. With the recent advancements in interactive music programming environments running on general-purpose computers, we may be fast approaching the day in which a standard methodology for specifying electroacoustic instruments and compositional or performance techniques is indeed possible. In the meantime, I offer my approach as one possible model for such a standard. The test of its viability will come from its usefulness in the hands of other researchers.

12.1 REAL-WORLD TESTING

I would be very interested to see the results of a third party realization of one of the works I have described in this paper. To date, I know of one performance that has already taken place using my analysis and reconstruction of the digital delay for Musgrave’s Narcissus. An even better test might be for a performer/engineer to create a new realization from scratch using only this paper and the score as a guide.

---

^{95} Todd Welbome, a professor of piano and music technology at the University of Wisconsin, Madison used a prototype version of my Narcissus software in preparing for flutist Elizabeth Marshall’s May 2004 performance of Musgrave’s work.
One further practical application for this analysis is the possibility of preparing critical editions of the works analyzed. Such an edition would include a version of the analysis in the form of technical notes to accompany the score. A working software realization (or even several alternate versions) of the required interactive system could be distributed with the score as well (either on fixed media or via Internet). In this case, most performers would ideally have to be concerned only with the software. They would, however, be armed with a detailed analysis in case they reach a situation in which the software needs modification, is no longer supported, or will not run on available machines.

12.2 LIMITATIONS OF THIS MODEL

I have developed my analysis according to the immediate requirements of the four pieces under consideration. If this model for analysis is extended to cover a wider range of works, there will be instances in which new descriptive techniques will have to be employed and possibly invented.

Two situations spring to mind that might stretch the ability of mere text to cope with the demands of adequately describing the electronic resources called for by a composer. The first awkward situation for abstract analysis of electronic systems is encountered in when the original instrument is itself the exact and irreproducible embodiment of the composer's intentions. A case in point is given in Stockhausen's remarks regarding a new realization of Mikrophonie I:
There was the problem that we could no longer rebuild the old filters. They were so-called 
Hörspiel-Verzerrer W 49 [W 49 Radio-play distorters], built in-house at the WDR in Cologne: filters with carbon strips. It is really interesting how very old-fashioned that sounds (after all, violins with catgut are used today). Such materials are glorious, aren't they? The two metal levers of the filters scrape along the carbon strips, and spray must now and then be used. Today if you try to substitute computerized filter simulations, the characteristic sound goes to hell. The scraping and the skips between the filter levels is lost; but they actually belong to such a sound, when it is brightened up from below to above, or vice-versa.\textsuperscript{66}

In such cases, computer simulation may be the only option, but we are aware that something important is missing. Here the performance-practice problem of authenticity and original instruments has crept into the repertoire of the very recent past. Perhaps complete schematics for reconstructing the original instrument would be necessary to achieve a fully authentic realization of the composer's intentions. We therefore have a situation not unlike that faced by a clarinetist preparing a performance of Mozart's Concerto in A Major, K. 622, for which an "authentic" realization requires a reproduction of an instrument that was unique to Vienna in 1791.

A second difficult situation arises when a piece of music requires recorded sounds that cannot be recreated according to a set of instructions. While any clarinetist using the provided excerpts can recreate the pre-recorded sounds required in the works by Kramer and Lippe discussed in this paper, some pieces require concrète sound samples that are unique and irreproducible. A case in point is my own Memories of You... (for clarinet and recorded voice), written in 1992 and revised in 1994 for interactive computer electronics. The recorded voice referred to in the title is that of my maternal grandfather.

\textsuperscript{66} Karlheinz Stockhausen, "Electroacoustic Performance Practice," Perspectives of New Music 34, no. 1 (1996): 97.
Alan J. Davidson (1899 – 1996). Excerpts from his spoken memoirs, stored as digital audio files that are triggered in response to footswitch signals from the stage, are used as a counterpoint to the solo clarinet line. If another clarinetist wanted to perform this work twenty years from now, could this piece be preserved? Since the samples are spoken-word recordings they can be easily transcribed as text. However, this leaves out the inflection, vocal quality, and historical authenticity that give these sounds their essence. The sound files are stored in a standard audio file format. Can we be certain they will be playable in 20 years? On what media should they be stored? One extreme strategy for avoiding the eventual obsolescence of digital media formats is to store the streams of audio sample data as text on paper. This would create very large appendices indeed, since each second of stereo sound is represented by 88,200 sample values. Existing analysis/re-synthesis and audio compression techniques might aid in reducing the necessary data somewhat, but any strategy of that type would also require thorough accompanying documentation describing the decoding process. I imagine the Rosetta stone of the future to be a multi-lingual digital file format specification, chiseled in some durable, non-magnetic medium. The best strategy for short-term preservation at the present time seems to be to store the necessary recorded sounds in as many formats on as many types of standard media as possible, update when possible, and otherwise hope for the best. Long-term viability of recorded sound is a major issue that requires a concerted effort on the part of the digital media community at large.
12.3 NEXT STEPS

I now have a small repertoire of four works for clarinet and interactive electronics fully analyzed and ready either for performance or for a new realization. My immediate plans for this work include a realization and performance of Jonathan Kramer’s *Renascence* at the next opportunity. Furthermore, there are a number of other works for clarinet and interactive electronics that I considered for this project but was unable to accommodate for various reasons. These include Richard Boulanger’s *from Temporal Silence* (1989), Todd Winkler’s *Snake Charmer* (1992), and Morton Subotnick’s *Passages of the Beast* (1978). Each of these works would be an excellent candidate for the same type of analysis I have presented in this paper, though some of the particular challenges would be different.

Naturally, as a clarinetist, I am primarily concerned with the repertoire for my own instrument. However, there is nothing inherent in my analysis model that would limit its application to clarinet pieces. Therefore, I would welcome the efforts of performers and engineers from diverse backgrounds to expand this project. Eventually, this type of analysis could encompass a general approach to building a viable repertoire of works for interactive electronics and acoustic instruments of all types.
### APPENDIX A

SELECTED LIST OF WORKS FOR SOLO CLARINET AND INTERACTIVE ELECTRONICS

**KEY TO PUBLISHERS**


**WORKS**

<table>
<thead>
<tr>
<th>Composer</th>
<th>Year</th>
<th>Title</th>
<th>Electronic Instruments</th>
<th>Pub.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bestor, Charles L.</td>
<td>1995</td>
<td><em>About Her</em></td>
<td>Interactive electronics</td>
<td>TM</td>
</tr>
<tr>
<td>Bestor, Charles L.</td>
<td>1993</td>
<td><em>Conversations with Myself</em></td>
<td>Interactive electronics</td>
<td>TM</td>
</tr>
<tr>
<td>Bestor, Charles L.</td>
<td>2002</td>
<td><em>Music for Gerry</em></td>
<td>Interactive electronics</td>
<td>TM</td>
</tr>
<tr>
<td>Boulanger, Richard</td>
<td>1991</td>
<td><em>From Temporal Silence</em></td>
<td>Currently 2 radio batons</td>
<td>MS</td>
</tr>
<tr>
<td>Boulez, Pierre</td>
<td>1985</td>
<td><em>Dialog</em></td>
<td>Tape or interactive electronics</td>
<td>UE</td>
</tr>
<tr>
<td>Brockman, Jane</td>
<td>1989</td>
<td><em>Ningana</em></td>
<td>Pitch-to-MIDI interface, synthesizer, tape</td>
<td></td>
</tr>
<tr>
<td>Druckman, Jacob</td>
<td>1969</td>
<td><em>Animus III</em></td>
<td>Tape, Feedback</td>
<td>BH</td>
</tr>
<tr>
<td>Errante, F. Gerard</td>
<td>1995</td>
<td><em>Shadows of Ancient Dreams</em></td>
<td>Digital delay</td>
<td></td>
</tr>
<tr>
<td>Errante, F. Gerard</td>
<td>1999</td>
<td><em>Silent Tears</em></td>
<td>Digital delay</td>
<td></td>
</tr>
<tr>
<td>Fulkerson, James</td>
<td>1981</td>
<td><em>Musing</em> (cl left-hand)</td>
<td>Digital delay</td>
<td></td>
</tr>
<tr>
<td>Gehilhaar, Rolf</td>
<td>1977</td>
<td><em>Polymorph</em> (bcl)</td>
<td>Tape delay</td>
<td>SM</td>
</tr>
<tr>
<td>Hamel, Kieth</td>
<td>1995</td>
<td><em>Traces</em> (Commissioned by Jean-Guy Boisvert)</td>
<td>Interactive electronics</td>
<td>CM</td>
</tr>
<tr>
<td>Composer</td>
<td>Year</td>
<td>Title</td>
<td>Electronic Instruments</td>
<td>Pub.</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Harrison, Jonty</td>
<td></td>
<td><em>Monodies</em> (bcl)</td>
<td>Live elec.</td>
<td>BM</td>
</tr>
<tr>
<td>Johnston, Ben</td>
<td>1970</td>
<td><em>Casta</em> (any instrument)</td>
<td>3 tape recorders, 4 speakers, 3 tape loops, mixers, microphones</td>
<td>MP</td>
</tr>
<tr>
<td>Nagel, Jody</td>
<td>1996</td>
<td><em>Kaleidoscope</em></td>
<td>Computer Processing</td>
<td></td>
</tr>
<tr>
<td>Kramer, Jonathan</td>
<td>1974</td>
<td><em>Renascence</em></td>
<td>Tape, Tape Delay</td>
<td>GS</td>
</tr>
<tr>
<td>Krieger, Ulrich</td>
<td>1993</td>
<td><em>Do you know what heaven sounds like?</em> (bcl)</td>
<td>Tape, live elec.</td>
<td>AR</td>
</tr>
<tr>
<td>Lake, Larry</td>
<td>1978</td>
<td><em>Options for Howard K</em> (electrified bcl)</td>
<td>4 synthesizers, tape.</td>
<td>CM</td>
</tr>
<tr>
<td>Lippe, Cort</td>
<td>1992</td>
<td><em>Music for Clarinet and ISPW</em></td>
<td>Interactive software system (DSP and sampling)</td>
<td>MS</td>
</tr>
<tr>
<td>Lowenstein, Michael</td>
<td>1992</td>
<td><em>After the Rain</em> (bcl)</td>
<td>Interactive electronics</td>
<td></td>
</tr>
<tr>
<td>Malsky, Matthew</td>
<td>1998</td>
<td><em>Ancient Devices</em></td>
<td>Live computer processing</td>
<td>HS</td>
</tr>
<tr>
<td>Musgrave, Thea</td>
<td>1987</td>
<td><em>Narcissus</em></td>
<td>Digital delay</td>
<td>NM</td>
</tr>
<tr>
<td>Pennycook, Bruce</td>
<td>1990</td>
<td><em>Praescio IV</em></td>
<td>Interactive MIDI system software</td>
<td>PR</td>
</tr>
<tr>
<td>Pinkston, Russell</td>
<td>2002</td>
<td><em>Gerrymander</em></td>
<td>Computer processing</td>
<td></td>
</tr>
<tr>
<td>Polansky, Larry</td>
<td>1987</td>
<td><em>17 Simple Melodies of the Same Length (for Dan Goode)</em></td>
<td>Interactive computer</td>
<td></td>
</tr>
<tr>
<td>Raes, Godfried-Willem</td>
<td>2001</td>
<td><em>WoodStock</em></td>
<td>PC</td>
<td></td>
</tr>
<tr>
<td>Rovan, Joseph Butch</td>
<td>1997</td>
<td><em>Continuities II</em></td>
<td>interactive electronics</td>
<td>SI</td>
</tr>
<tr>
<td>Rovan, Joseph Butch</td>
<td>1997</td>
<td><em>L'Obvie / l'obtus</em></td>
<td>gestural controller and interactive electronics</td>
<td>SI</td>
</tr>
<tr>
<td>Sandroff, Howard</td>
<td>1990</td>
<td><em>Tephilla</em></td>
<td>Digital Signal Processors</td>
<td>GP</td>
</tr>
<tr>
<td>Scheidt, Daniel</td>
<td>1990</td>
<td><em>Squeeze</em> (bcl)</td>
<td>Interactive software</td>
<td></td>
</tr>
<tr>
<td>Shatin, Judith</td>
<td>1997</td>
<td><em>Sea of Reeds</em></td>
<td>Digital delay and reverb</td>
<td>WM</td>
</tr>
<tr>
<td>Shing, Seongah</td>
<td>1994</td>
<td><em>Regress in Infinity</em></td>
<td>Interactive effects processing</td>
<td>MS</td>
</tr>
<tr>
<td>Smith, William O.</td>
<td>1985</td>
<td><em>Asana</em></td>
<td>Digital delay and pitch transposer</td>
<td>RE</td>
</tr>
<tr>
<td>Composer</td>
<td>Year</td>
<td>Title</td>
<td>Electronic Instruments</td>
<td>Pub.</td>
</tr>
<tr>
<td>--------------------</td>
<td>------</td>
<td>------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Spaarnay, Harry</td>
<td>1982</td>
<td><em>Bouwstenen</em>, bcl, tape delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steinberg, Paul</td>
<td>1988</td>
<td><em>Elegy for Ray and Dale</em></td>
<td>Digital effects</td>
<td>DN</td>
</tr>
<tr>
<td>Steinberg, Paul</td>
<td>1983</td>
<td><em>Micro-Electroecho</em></td>
<td>Three tape recorder delay</td>
<td>DP</td>
</tr>
<tr>
<td>Stockhausen, Karlheinz</td>
<td>1965</td>
<td><em>Solo</em></td>
<td>Multi-tap delay with feedback</td>
<td>UE</td>
</tr>
<tr>
<td>Subotnick, Morton</td>
<td>1978</td>
<td><em>Passages of the Beast</em></td>
<td>“Ghost Electronics” (tape-controlled Stereo Location Processor, Ring Modulator, Frequency Shifter, VCA)</td>
<td>TP</td>
</tr>
<tr>
<td>Westlake, Nigel</td>
<td>1984</td>
<td><em>Onomatopoeia</em> (bcl)</td>
<td>digital delay</td>
<td>AU</td>
</tr>
<tr>
<td>Winkler, Todd</td>
<td>1992</td>
<td><em>Snake Charmer</em></td>
<td>Computer processing/synthesizer</td>
<td></td>
</tr>
<tr>
<td>Yun, Seunghyun</td>
<td>1995</td>
<td><em>mm-ah-uh-ee-oo</em></td>
<td>Interactive effects processing</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

EVENT LIST FOR BRUCE PENNYCOOK’S PRAESCIO IV

T = Footswitch trigger (i.e, MIDI cc#64, val 127); P = MIDI note # from Pitch Tracker; Track = SMF sequence track name; Chan = MIDI output channel(s); Trans = transposition of track as semitone offset; Harm = intervallic doubling of sequence as semitone offset (not used); Vel = velocity scaling (multiplier); Temp = tempo scaling (* sequence durations); H1 = Thru harmony note 1 (as semitone offset from tracked clarinet pitch); H2 = harmony 2; H3 = harmony 3; H4 = harmony 4; Hvel = Harmonization velocity scaling (multiplier). Event list printed by permission from the composer.

<table>
<thead>
<tr>
<th>Ev. #</th>
<th>Trigger</th>
<th>Play Event Parameters:</th>
<th>Thru Event Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>P</td>
<td>Track</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>3a</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3b</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>3c</td>
<td>12</td>
<td>-2</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>4</td>
<td>4, 11</td>
</tr>
<tr>
<td>5</td>
<td>59</td>
<td>5a</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5b</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>6</td>
<td>59</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>59</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>53</td>
<td>8a</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8b</td>
<td>14</td>
<td>-2</td>
</tr>
<tr>
<td>9</td>
<td>82</td>
<td>9a</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>9b</td>
<td>8</td>
<td>-2</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>88</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>85</td>
<td>14a</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td></td>
<td>19a</td>
</tr>
<tr>
<td></td>
<td>19b</td>
<td>6</td>
<td>-2</td>
</tr>
<tr>
<td>20</td>
<td>51</td>
<td>20a</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>20b</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>20c</td>
<td>4</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>20d</td>
<td>5</td>
<td>-2</td>
</tr>
<tr>
<td>21</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>62</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>X</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>65</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Ev. #</td>
<td>Trigger:</td>
<td>Play Event Parameters:</td>
<td>Thru Event Parameters:</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>P</td>
<td>Track</td>
</tr>
<tr>
<td>26</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>65</td>
<td></td>
<td>20a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20c</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20d</td>
</tr>
<tr>
<td>29</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>31</td>
<td>X</td>
<td></td>
<td>31a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31c</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31e</td>
</tr>
<tr>
<td>32</td>
<td>54</td>
<td></td>
<td>32a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32c</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32f</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32g</td>
</tr>
<tr>
<td>33</td>
<td>X</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>34</td>
<td>90</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>35</td>
<td>65</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>36</td>
<td>73</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>37</td>
<td>74</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>38</td>
<td>63</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>39</td>
<td>54</td>
<td></td>
<td>39a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>40</td>
<td>X</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40a</td>
</tr>
<tr>
<td>41</td>
<td>60</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>42</td>
<td>X</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>43</td>
<td>68</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>44</td>
<td>X</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>45</td>
<td>X</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>46</td>
<td>72</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>47</td>
<td>76</td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>49</td>
<td>89</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>50</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>51</td>
<td>76</td>
<td></td>
<td>49a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>49b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>49c</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>49d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>49e</td>
</tr>
<tr>
<td>Ev. #</td>
<td>Trigger:</td>
<td>Play Event Parameters:</td>
<td>Thru Event Parameters:</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>P</td>
<td>Track</td>
</tr>
<tr>
<td>53</td>
<td>X</td>
<td></td>
<td>51a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51c</td>
</tr>
<tr>
<td>54</td>
<td>X</td>
<td></td>
<td>52a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>52a</td>
</tr>
<tr>
<td>55</td>
<td>X</td>
<td></td>
<td>52a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>52b</td>
</tr>
<tr>
<td>56</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>57</td>
<td>X</td>
<td></td>
<td>55a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>55a</td>
</tr>
<tr>
<td>58</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>59</td>
<td>X</td>
<td></td>
<td>55a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>55a</td>
</tr>
<tr>
<td>60</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>61</td>
<td>X</td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>62</td>
<td>50</td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>63</td>
<td>51</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>64</td>
<td>55</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>65</td>
<td>59</td>
<td></td>
<td>62</td>
</tr>
<tr>
<td>66</td>
<td>61</td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>67</td>
<td>72</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>68</td>
<td>88</td>
<td></td>
<td>65a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>65b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>65c</td>
</tr>
<tr>
<td>69</td>
<td>89</td>
<td></td>
<td>66a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>66b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>66c</td>
</tr>
<tr>
<td>70</td>
<td>91</td>
<td></td>
<td>67a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>67b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>67c</td>
</tr>
<tr>
<td>71</td>
<td>55</td>
<td></td>
<td>68a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>68b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>72a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>72b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>72c</td>
</tr>
<tr>
<td>72</td>
<td>50</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>73</td>
<td>81</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>74</td>
<td>X</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>75</td>
<td>64</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>76</td>
<td>64</td>
<td></td>
<td>72a</td>
</tr>
<tr>
<td>77</td>
<td>X</td>
<td></td>
<td>72a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>72d</td>
</tr>
<tr>
<td>78</td>
<td></td>
<td></td>
<td>72a</td>
</tr>
</tbody>
</table>
APPENDIX C
SPECIFICATIONS AND BLOCK DIAGRAMS FOR SIGNAL PROCESSING MODULES AND SYSTEM CONTROL ALGORITHMS USED IN CORT LIPPE’S 
MUSIC FOR CLARINET AND ISPW

The following series of signal processing module block diagrams and pseudo-code algorithm outlines are intended to supplement the analysis given in chapter 8. For the block diagrams, I have adopted a set of symbols based on those found in many standard textbooks on computer music and digital synthesis. A key to the symbols used in this appendix is shown in figure C.1. Examples of Max/MSP signal processing patchers with corresponding diagrams are given in figures C.2 – C.4. Detailed specifications and signal routing schematics are given for the DSP modules (Sampler, Harmonizer, Frequency Shifter, Flange, Frequency/Amplitude Modulation, Reverb, Noise Modulation, and Spatializer) along with pseudo code outlines of their associated control algorithms.

Figure C.1. Basic symbols used in the block diagrams of Cort Lippe’s signal-processing software instruments
Figure C.2. A simple Max/MSP instrument for additive synthesis and its corresponding block diagram

Figure C.3. A Max/MSP instrument for amplitude modulation (AM synthesis) and its corresponding block diagram
Figure C.4. A phase-driven wavetable oscillator in Max/MSP and its corresponding block diagram

The following are block diagrams of Lippe’s signal processing modules and pseudo-code versions of their associated control algorithms, if applicable. The sampler is the central component of the system, and is therefore subject to multiple complex control algorithms. Figures C.5 and C.6 show the sampler instrument. Figures C.7, C.8, and C.9 show pseudo-code outlines of the three main granular sampling algorithms “Trevor,” “Trevor-Back” and “PLAY_RAND.” Eight more sampler control algorithms are associated with specific score events, and these are shown in figures C.10 – 17. The remaining DSP instruments (harmonizer, frequency shifter, flange, frequency/amplitude modulation, reverb, noise modulation, and spatializer, along with associated control algorithms) are given as signal-processing block diagrams and pseudo-code outlines in figures C.18 – C.30.
**SAMPLER**

### Sampler 1

**Variable Parameters:**
- Pitch, Amp, Onset, Attack,
- Duration, Decay, Gliss range,
- Gliss time, Sample number

**Note Generator:**
- Pitch value triggers playback; all other variables carry over until replaced.
- Playback is handled by next available Sampler Voice Module

### Sampler 2

**Variable Parameters:**
- Pitch, Amp, Onset, Attack,
- Duration, Decay, Gliss range,
- Gliss time, Sample number

**Note Generator:**
- Pitch value triggers playback; all other variables carry over until replaced.
- Playback is handled by next available Sampler Voice Module

---

**Caveats:**
1. **Note Generators:** List of parameter values for sampler note data is sent to a Sampler Voice Module when a pitch value is received from the event list or from another processing module. Previously stored values (for parameters other than pitch) will be carried over unless a new value is received.

2. Eight-voice polyphonic voice allocation: each note generated is automatically allocated to an available Sampler Voice Module. “Voice Stealing” grabs the least recently used module if more than eight voices are needed simultaneously.

3. Sampler sounds are generated as MIDI notes: Note number determines pitch, velocity determines amplitude. Duration is determined by sending a corresponding note with a velocity (amp) of 0 at the appropriate time.

---

**Figure C.5. Samplers 1 and 2:** playback is controlled by formulating MIDI-style note packets of variable parameters for pitch, velocity, duration, envelope, and glissando.
real-time input variables:

- **sample number**: (1-8)
- **pitch**: MIDI (note x 100 + cents; middle C = 6000)
- **onset**: msec
- **gliss range**: %
- **gliss time**: msec

### Sample Player

**Load sample table**: sample number 1-8

* **Pitch control**: \(( \text{speed} = \frac{2616256}{\text{freq. in Hz of (pitch x 0.01)}} \) *

**Onset/direction**: start point = onset (positive values play forward, negative values play backwards)

**Gliss range/time**: playback speed ramped from “pitch” to “pitch x gliss range” over period set by gliss time.

Values received for sample number, gliss time, gliss range, decay time, attack time, onset, and amplitude are stored until pitch value is received, initiating playback.

* Pitch control is based on playback duration in msec of a 10000 msec file, scaled to map across the tempered scale, e.g., pitch (MIDI+): 6000 = 10000 (normal pitch); 7200 = 5000 (twice as fast/up one octave); 5950 = 1 quarter tone flat

### Sound Output

Figure C.6. Sampler Voice Module: each of the 16 sampler voice modules can read from any of the buffered sample files. Input variables control pitch, amplitude, file onset, envelope attack and decay rates, and glissando (range and duration)
GRANULAR SAMPLING ALGORITHMS

START is set by one of the following variables received:

\[
\begin{align*}
\text{START} & = \text{t-start} = 0; \text{ OR} \\
\text{START} & = \text{t-start1} = 0; \text{ OR} \\
\text{START} & = \text{t-start2} = 200000; \text{ OR} \\
\text{START} & = \text{t-start3} = 400000; \text{ OR} \\
\text{START} & = \text{t-start4} = 600000; \text{ OR} \\
\text{START} & = \text{t-start5} = 800000; \text{ OR} \\
\text{START} & = \text{tt-start} = (0 - 10000) * 100 \\
\end{align*}
\]

LOOP every 10 to 20 milliseconds (10 + random number from 0 to 10) UNTIL t-stop message is received {
  onset = (START + (value of t-process * 11)) / 100
  velocity = 75
  duration = 50
  pitch = current value of t-transpose (may be changed while the loop is running)
  PLAY note on Sampler1 (previous values for sample table, attack, decay, gliss, and gliss-duration are maintained)
}

Figure C.7. “Trevor.” Granular sampling playback initiated by event list variable t-start

START is set by one of the following variables received:

\[
\begin{align*}
\text{START} & = \text{b-start} = 200000; \text{ OR} \\
\text{START} & = \text{b-start1} = 200000; \text{ OR} \\
\text{START} & = \text{b-start2} = 400000; \text{ OR} \\
\text{START} & = \text{b-start3} = 600000; \text{ OR} \\
\text{START} & = \text{b-start4} = 800000; \text{ OR} \\
\text{START} & = \text{b-start5} = 970000; \text{ OR} \\
\text{START} & = \text{bb-start} = (0 - 10000) * 100 \\
\end{align*}
\]

LOOP every 10 to 20 milliseconds (10 + random number from 0 to 10) UNTIL b-stop message is received {
  onset = (START - (value of b-process * 11)) / 100
  velocity = 75
  duration = 50
  pitch = current value of b-transpose (may be changed while the loop is running)
  PLAY note on Sampler2 (previous values for sample table, attack, decay, gliss, and gliss-duration are maintained)
}

Figure C.8. “Trevor-back.” Backwards granular sampling initiated by event list variable b-start
IF \textit{play-rand} message is received THEN {
    CALL FUNCTION \textit{play-rand1}
    CALL FUNCTION \textit{play-rand2}
}

FUNCTION \textit{play-rand1} {
    LOOP every 10 to 60 milliseconds \((10 + \text{random number from 0 to 50})\) {
        \begin{align*}
            &\text{CALL FUNCTION \textit{rnd-gliss}} \\
            &\text{CALL FUNCTION \textit{playnote-1}}
        \end{align*}
    }
}

FUNCTION \textit{play-rand2} {
    LOOP every 10 to 60 milliseconds \((10 + \text{random number from 0 to 50})\) {
        \text{CALL FUNCTION \textit{playnote2}}
    }
}

FUNCTION \textit{Playnote-1} {
    \begin{align*}
        \text{onset} &= \text{\textit{play-rand-ondurl}} + \text{random number from 0 to 4000} \\
        \text{velocity} &= 80 \\
        \text{duration} &= 10 + \text{random number from 0 to 400} \\
        \text{pitch} &= (\text{\textit{play-rand-pitl}} + \text{random number from 0 to 100}) \\
            &\quad - (\text{\textit{play-rand-pchvall}} / 2)
    \end{align*}
    \text{PLAY note on sampler1}
}

FUNCTION \textit{rnd-gliss} {
    \begin{align*}
        \text{IF value of \textit{rand-gliss-gate} is 1, THEN} & \quad \{ \\
            N &= \text{value of counter} \% 8 \\
            \text{counter} &= \text{counter} + 1 \\
            \text{IF value of } N &= \{ \\
            0: \text{gliss} &= 0.8 \\
            1: \text{gliss} &= 1.05 \\
            2: \text{gliss} &= 0.7 \\
            3: \text{gliss} &= 1.1 \\
            4: \text{gliss} &= 0.6 \\
            5: \text{gliss} &= 1.15 \\
            6: \text{gliss} &= 0.5 \\
            7: \text{gliss} &= 1.2 \\
            \}
        \}
        \text{IF value of \textit{rand-bang8-gate} is 1 THEN} \quad \{ \\
            \text{CALL FUNCTION \textit{playnote-1} 8 times in immediate succession using current parameters} \\
        \}
        \text{IF value of \textit{ranp-bang4-gate} is 1 THEN} \quad \{ \\
            \text{CALL FUNCTION \textit{playnote-1} 4 times in immediate succession using current parameters} \\
        \}
    \}
}

Figure C.9. “PLAY_RAND.” Granular sampling controlled by random processes
ADDITIONAL SAMPLER CONTROL ALGORITHMS

IF value of Runsgate is 1 THEN
{
glisstot = RAMP from previous value to value 1 of glisstot over duration specified by glisstot value 2
N = value of metro_playtot_val
LOOP every N msec
{
    PLAY note on Sampler1
    {
        Duration = 261 / frequency in Hz of (glisstot + random number from 0 to 50) * 9500
        Pitch = glisstot + (random number from 0 to 50)
        Onset = 0
        Velocity = 75
    }
}
}

Figure C.10. Sampler control algorithm 1: Section I, events 5 – 6 and 9

IF the value of metro_evt5 is 1, THEN
SET sampler1 onset to 0;
LOOP: every N milliseconds
{
    N = rspeed_evt5 + a random number from 0 to 400 (seed: speed_evt5)
    PLAY Sampler1
    {
        Pitch = 6000 + value of glis_evt5
        Velocity = 120
        Duration = value of speed_evt5_dur
    }
}

Figure C.11. Sampler control algorithm 2: section I, events 5 – 10
IF the value of `ichgate` is 1, THEN
   
   SET the value of `tt02`, `sto2`, `tt04`, and `sto4` to 127; (full volume)
   IF the value of `pitch-track-out` is between 50 and 52, THEN
      
      TRIGGER `spatXY` spatialization algorithm;
      SET `b-precess` value to `[pitch-track-out * b-precess-val]`;
      SET `b-transpose` value to `[pitch-track-out - 3]`;
      TRIGGER granular sampling module `Trevor-back` with onset value set by `brevor-onset`;
   }
   IF the value of `pitch-track-out` is between 76 and 79, THEN
      
      SET `t-precess` value to `[pitch-track-out * t-precess-val]`;
      SET `t-transpose` value to value of `pitch-track-out`;
      TRIGGER granular sampling module `Trevor` with onset value set by `Trevor-onset`;
   }

Figure C.12. Sampler control algorithm 3: “Trevor” in section I, events 12 – 16  
(also shown in figure 8.11)

IF value of `evt11-sec2` is 1, THEN
   
   PLAY `Sampler2`
   {
      pitch = 6000  
duration = 5000
      onset = 0
      attack = 10
      decay = 50
      gliss = 1.0
   }
   LOOP every (150 + random number from 0 to 700) msec
   {
      PLAY `Sampler2`
      {
         pitch = 5300 + random number from 0 to 1500  
(Dur., Onset, etc. carry over from previous)
      }
   }

Figure C.13. Sampler control algorithm 4: Section II, events 11 and 26
IF start-23 = 1 THEN 
{
LOOP every 0 to 2000 msec (randomized) 
{
N = random number from 0 to 2000
IF value of N is 0, THEN t-stop (stop Trevor)
IF value of N is between 1 and 999, THEN 
{
LOOP every 20 to 520 msec 
{
t-precess = random number from 0 to 256
    t-transcent = 3000 + (random number from 0 to 3000)
    t-start (start Trevor granular sampling module)
    SEND t-stop after 500 msec delay
}
}
IF value of N >= 1000 
{
    Play-rand2 = 1 (see PLAY_RAND)
    Stop PLAY_RAND after 100 msec delay 
}
}
}

Figure C.14. Sampler control algorithm 5: Section III, event 23

IF value of ichgate3 is 1 THEN 
{
IF (value of pitch_track_out - 6) is between 64 and 76, THEN 
{
t-precess = (pitch_track_out - 6) * 10
    t-transcent = (pitch_track_out - 6) * 100
    SEND tt-start message to Trevor
}
ELSE IF (value of pitch_track_out - 6) is between 77 and 92, THEN 
{
    N = random number from 0 to 5
    IF N is 
        0: t-precess = 840; t-transcent = RAMP to 8400 in 400 msec
        1: t-precess = 830; t-transcent = RAMP to 8300 in 400 msec
        2: t-precess = 820; t-transcent = RAMP to 8200 in 400 msec
        3: t-precess = 810; t-transcent = RAMP to 8100 in 400 msec
        4: t-precess = 800; t-transcent = RAMP to 8000 in 400 msec
        5: t-precess = 790; t-transcent = RAMP to 7900 in 400 msec
    SEND tt-start to Trevor

}
}

Figure C.15. Sampler control algorithm 6: Section III, event 26
IF `bang4-line` = 1 THEN
{
  LOOP every 1000 to 4000 msec (1000 + random number from 0 to 3000 (seed: bang-ran))
  
  START RAMP: go from previous value to new value over ramp-time, incrementing at 30 msec intervals
  
  Ramp-time = `bang-ran` + random 3000 (seed: `bang-ran`)
  FOR each RAMP increment
  
  Pitch = bang-pit + random number from 0 to 3000
}

Figure C.16. Sampler control algorithm 7: send ramping pitch values to PLAY_RAND in Section IV, event 3

IF value of `runsgate` is 1 THEN
{
  FOR every `pitch-track-out` value received
  
  PLAY note on Sampler1
  
  Duration = 261 / frequency in Hz of (value of `pitch-track-out`) * 9500
  Pitch = `pitch-track-out` * 100
  Onset = 0
  Velocity = 75
}

Figure C.17. Sampler control algorithm 8: play Sampler1 in response to pitch tracker values in Section V, events 3 - 6
System variable "hwindl" (0-127) scaled logarithmically:

\[
hwindl = 210 \cdot e^{-127 \log 1.06848} \cdot e^{hwindl \log 1.06848}
\]

\(e\) is the base of the natural logarithm (approx. 2.718282).

 OSC1: sawtooth wave generates linear ramp of values incrementing from 0 - 1

 OSC2: driven by OSC1

 OSC3: wind wavetable oscillator (see below)

 OSC4: wind wavetable oscillator (see below)

Figure C.18. Two delay lines with delay time increased at a constant linear rate. Delays are synchronized by OSC1, but delay time increments are out of phase by 180 degrees. Amplitude window ("wind" oscillators) controls envelope for delay output, cross-fading between the two delay lines.
Figure C.19. Frequency shifter: all harmonic components are shifted individually up or down by a fixed frequency interval
**FLANGE**

209

---

**Figure C.20.** A typical flange effect created by a short delay with LFO modulation

---

**FLANGE**

---

**Figure C.20.** A typical flange effect created by a short delay with LFO modulation
FREQUENCY/AMPLITUDE MODULATION

Figure C.21. Combination of FM and AM synthesis based on real-time analysis of clarinet pitch and amplitude.
ALGORITHMIC CONTROL OF HARMONIZER AND FREQUENCY SHIFTER

IF the value of \textit{fsgate18} is 1, THEN:
  IF \textit{pitch-track-out} value is 77,
    THEN {
      SET \textit{fsh01} value to a random number from -2000 to -6499.
    }
  IF the value of \textit{pitch-track-out} is between 50 and 59,
    THEN {
      SET \textit{hfreq} value to a random number from 40 to 63;
      SET \textit{ptoh} value to 122; (clarinet signal vol. to harmonizer)
    }
  ELSE, SET \textit{ptoh} value to 0.
  IF \textit{pitch-track-out} value is between 71 and \textit{fstend18} value,
    THEN {
      SET \textit{ptof} value to 127; (clar. signal vol. to freq. shifter)
    }
  ELSE, SET \textit{ptof} value to 0.

Figure C.22. Harmonizer/Frequency Shifter control:
Section I, event 18 and Section II, event 1
REVERB

Figure C.23. Reverb: the input signal is subjected to a series of delays with variable feedback.
Figure C.24. Noise Modulation: random amplitude envelopes generated by multiple noise wavetable LFOs
Figure C.25. Spatializer input crossbar: inputs from all DSP modules are mixed individually for left and right channel output
Left-Right placement of DSP and Sampler output is controlled per output channel by individual variables for each DSP module. The input crossbar method shown in figure C.25 is used to manage input to all of the individual DSP modules as well (except the Sampler). Final output to the loudspeakers is shown in figure C.26.

**Spatializer Output**

![Diagram of Spatializer Output]

**Figure C.26.** Spatializer output: signal from the input crossbar is scaled and sent to the Digital to Analog Converters (DAC) for amplification via loudspeakers
SPATIALIZER CONTROL ALGORITHMS

IF value of $ichgate$ is 1 THEN {
  IF value of $pitch\_track\_out$ is between 50 and 62 THEN {
    IF $spagate$ is 1 THEN {
      $N$ = random number from 0 to 1
      IF value of $N$ is 0, THEN $spatx = 127$ AND $spaty = 0$
      ELSE $spatx = 0$ AND $spaty = 127$
    }
    IF $s\_spatR$ is 1, THEN $stoR = spatx$
    IF $t\_spatR$ is 1, THEN $ttoR = spaty$
    IF $f\_spat$ is 1, THEN $fto2 = spatx$ AND $fto4 = spaty$
    IF $h\_spat$ is 1, THEN $hto2 = spatx$ AND $hto4 = spaty$
    IF $R\_spat$ is 1, THEN $Rto2 = spatx$ AND $Rto4 = spaty$
    IF $n\_spat$ is 1, THEN $nto2 = spatx$ AND $nto4 = spaty$
    IF $t\_spat$ is 1, THEN $tto2 = spatx$ AND $tto4 = spaty$
  }
}

Figure C.27. Spatializer control algorithm: section I, events 12-16. Hard-left or hard-right placement of the computer-generated sound is chosen randomly each time a note between 50 and 62 is played and is detected by the pitch tracker.

IF value of $spatXY\_19$ is 1 THEN {
  LOOP every 1000 to 6000 msec (1000 + random value 0 to 5000
  (seed: $spatXY\_metro$))
  }
  
  $N$ = random value from 0 to 1
  IF $N$ is 0, THEN {
    $spatx = 127$
    $spaty = 0$
  }
  ELSE {
    $spatx = 0$
    $spaty = 127$
  }
  IF $s\_spatR$ is 1, THEN $stoR = spatx$
  IF $t\_spatR$ is 1, THEN $ttoR = spaty$
  IF $f\_spat$ is 1, THEN $fto2 = spatx$ AND $fto4 = spaty$
  IF $h\_spat$ is 1, THEN $hto2 = spatx$ AND $hto4 = spaty$
  IF $R\_spat$ is 1, THEN $Rto2 = spatx$ AND $Rto4 = spaty$
  IF $n\_spat$ is 1, THEN $nto2 = spatx$ AND $nto4 = spaty$
  IF $t\_spat$ is 1, THEN $tto2 = spatx$ AND $tto4 = spaty$

Figure C.28. Spatializer control algorithm: random left-right placement of DSP modules’ output in section II, event 19 through two seconds after the onset of event 20.
IF value of \textit{spaton} is {
\begin{verbatim}
1: N = 0;
  REPEAT every 20 msec {
    "spaf1.t" table index = (N & 255) + (spatinc - 64)
    "spaf2.t" table index = ((N & 255) + (spatinc - 64)) >> 1
    N = (N & 255) + (spatinc - 64)
  }
2: N = 0;
  REPEAT every 20 msec {
    "spaf3.t" table index = (N & 255) + (spatinc - 64)
    "spaf4.t" table index = ((N & 255) + (spatinc - 64)) >> 1
    N = (N & 255) + (spatinc - 64)
  }
3: N = 0;
  REPEAT every 20 msec {
    "spaf5.t" table index = (N & 255) + (spatinc - 64)
    "spaf6.t" table index = ((N & 255) + (spatinc - 64)) >> 1
    N = (N & 255) + (spatinc - 64)
  }
4: N = 0;
  REPEAT every 20 msec {
    "spaf7.t" table index = (N & 255) + (spatinc - 64)
    "spaf8.t" table index = ((N & 255) + (spatinc - 64)) >> 1
    N = (N & 255) + (spatinc - 64)
  }
\end{verbatim}
}\end{enumerate}

Figure C.29. Algorithmic control of spatializer using event list variable \textit{spaton}.
Figure C.30. Amplitude tables used by *spaton* algorithm
APPENDIX D
KEY TO SYSTEM VARIABLES USED IN CORT LIPPE’S
MUSIC FOR CLARINET AND ISPW

The following is a list of system variables found in the event list for *Music for Clarinet and ISPW*. The variable name is given as it is used in Lippe’s software, followed by its range of values and a brief description of its target. Note that for most variables, if two numbers are given the system will ramp to the first number over a period of time in milliseconds set by the second number (e.g., “pitch 60 1000” would send a continuous stream of values for “pitch” from its previous value to 60 over a period of 1 second). A value followed by a comma and two more numbers will cause the system to ramp from the first value to the second over the period defined by the third number. The term “bang” is a Max/MSP message that triggers an action.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bang4-line</td>
<td>1/0</td>
<td>Start/stop sampler control algorithm 7 (PLAY_RAND)</td>
</tr>
<tr>
<td>bang-pit</td>
<td>MIDI+</td>
<td>Pitch value for sampler control algorithm 7 (PLAY_RAND)</td>
</tr>
<tr>
<td>bang-ran</td>
<td>Time (msec)</td>
<td>Ramp time/seed value for sampler control algorithm 7 (PLAY_RAND)</td>
</tr>
<tr>
<td>brevor-onset</td>
<td>0 - 10000 msec</td>
<td>Sample file start point (Trevor_back)</td>
</tr>
<tr>
<td>b-stop</td>
<td>Bang</td>
<td>Stop (Trevor_back)</td>
</tr>
<tr>
<td>diramp</td>
<td>0 - 127</td>
<td>Dry signal amplitude (HARM)</td>
</tr>
<tr>
<td>evt11-sec2</td>
<td>0/1</td>
<td>Start/stop sampler control algorithm (see fig. C.13)</td>
</tr>
<tr>
<td>vamp01</td>
<td>0 - 127</td>
<td>Frequency shifter output</td>
</tr>
<tr>
<td>flange-amp</td>
<td>0 - 127</td>
<td>Flange output</td>
</tr>
<tr>
<td>flange-del</td>
<td>0 - 127</td>
<td>Flange delay</td>
</tr>
<tr>
<td>flange-index</td>
<td>0 - 127</td>
<td>Flange input</td>
</tr>
<tr>
<td>flange-loop</td>
<td>0 - 127</td>
<td>Flange feedback</td>
</tr>
<tr>
<td>flange-master</td>
<td>0 - 127</td>
<td>Flange final output</td>
</tr>
<tr>
<td>Variable</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>flange-speed</td>
<td>0 - 127</td>
<td>Flange LFO speed</td>
</tr>
<tr>
<td>fmam-master</td>
<td>0 - 127</td>
<td>Output from FM-AM</td>
</tr>
<tr>
<td>fnois</td>
<td>0 - 127</td>
<td>Speed of modulation (NOISE)</td>
</tr>
<tr>
<td>frog1-gliss</td>
<td>1.</td>
<td>Glissando curve value (SAMP. 1)</td>
</tr>
<tr>
<td>frog1-off</td>
<td>Time (msec)</td>
<td>Decay envelope time (SAMP. 1)</td>
</tr>
<tr>
<td>frog1-on</td>
<td>Time (msec)</td>
<td>Attack envelope time (SAMP. 1)</td>
</tr>
<tr>
<td>frog1-onset</td>
<td>0 - 10000 msec</td>
<td>Sample file start point (SAMP. 1)</td>
</tr>
<tr>
<td>frog2-gliss</td>
<td>1.</td>
<td>Glissando curve value (SAMP. 2)</td>
</tr>
<tr>
<td>frog2-off</td>
<td>Time (msec)</td>
<td>Decay envelope time (SAMP. 2)</td>
</tr>
<tr>
<td>frog2-on</td>
<td>Time (msec)</td>
<td>Attack envelope time (SAMP. 2)</td>
</tr>
<tr>
<td>frog2-onset</td>
<td>0 - 10000 msec</td>
<td>Sample file start point (SAMP. 2)</td>
</tr>
<tr>
<td>fsgate18</td>
<td>0/1</td>
<td>Start/stop HARM/FREQ control algorithm (see fig. C.22)</td>
</tr>
<tr>
<td>fsh01</td>
<td>MIDI+</td>
<td>Pitch (FREQ)</td>
</tr>
<tr>
<td>fto2</td>
<td>0 - 127</td>
<td>Freq. shift chan. 1 input (SPAT.)</td>
</tr>
<tr>
<td>fto4</td>
<td>0 - 127</td>
<td>Freq. shift chan. 2 input (SPAT.)</td>
</tr>
<tr>
<td>fton</td>
<td>0 - 127</td>
<td>Input from freq. shift (NOISE)</td>
</tr>
<tr>
<td>glis_evt5</td>
<td>On (1)/off (0)</td>
<td>Pitch modifier in sampler control algorithm (see fig. C.11)</td>
</tr>
<tr>
<td>hamp</td>
<td>0 - 127</td>
<td>Effect output amplitude (HARM.)</td>
</tr>
<tr>
<td>hdel</td>
<td>0 - 127</td>
<td>Delay in milliseconds (HARM.)</td>
</tr>
<tr>
<td>hfreq</td>
<td>0 - 127</td>
<td>LFO Frequency for delay time modulation (HARM.)</td>
</tr>
<tr>
<td>h-spat</td>
<td>0</td>
<td>Start/stop spatializer control of Harmonizer output</td>
</tr>
<tr>
<td>hto2</td>
<td>0 - 127</td>
<td>Harmonizer level to Left sound output (SPAT.)</td>
</tr>
<tr>
<td>hto4</td>
<td>0 - 127</td>
<td>Harmonizer level to Right sound output (SPAT.)</td>
</tr>
<tr>
<td>htoh</td>
<td>0 - 127</td>
<td>Harmonizer feedback</td>
</tr>
<tr>
<td>Hton</td>
<td>0 - 127</td>
<td>Harmonizer level (NOISE)</td>
</tr>
<tr>
<td>hwind1</td>
<td>0 - 127</td>
<td>Delay time offset (HARM.)</td>
</tr>
<tr>
<td>ichgate3</td>
<td>0/1</td>
<td>Start/Stop sampler control algorithm (see fig. C.15)</td>
</tr>
<tr>
<td>nto2</td>
<td>0 - 127</td>
<td>Noise Modulation level to Left sound output (SPAT.)</td>
</tr>
<tr>
<td>nto4</td>
<td>0 - 127</td>
<td>Noise Modulation level to Right sound output (SPAT.)</td>
</tr>
<tr>
<td>ntoR</td>
<td>0 - 127</td>
<td>Noise Modulation input (REVB.)</td>
</tr>
<tr>
<td>play_rand</td>
<td>On (1)/off (1)</td>
<td>Start/stop (PLAY_RAND)</td>
</tr>
<tr>
<td>play_rand_metrol</td>
<td>Time in msec</td>
<td>Grain frequency (PLAY_RAND)</td>
</tr>
<tr>
<td>Variable</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>play_rand_meter2</td>
<td>Time in msec</td>
<td>Grain frequency (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand</td>
<td>0/1</td>
<td>Start/stop (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand1</td>
<td>0/1</td>
<td>Start/stop (PLAY_RAND1)</td>
</tr>
<tr>
<td>play-rand2</td>
<td>0/1</td>
<td>Start/stop (PLAY_RAND2)</td>
</tr>
<tr>
<td>play-rand-dur1</td>
<td>Time in msec</td>
<td>Grain duration (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand-dur2</td>
<td>Time in msec</td>
<td>Grain duration (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand-ondur1</td>
<td>0 - 10000</td>
<td>Grain attack envelope (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand-ondur2</td>
<td>0 - 10000</td>
<td>Grain attack envelope (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand-onset</td>
<td>0 - 10000</td>
<td>Sample start point (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand-onset1</td>
<td>0 - 10000</td>
<td>Sample start point (PLAY_RAND1)</td>
</tr>
<tr>
<td>play-rand-onset2</td>
<td>0 - 10000</td>
<td>Sample start point (PLAY_RAND2)</td>
</tr>
<tr>
<td>play-rand-pchval1</td>
<td>MIDI+</td>
<td>Modifies pitch (PLAY_RAND1)</td>
</tr>
<tr>
<td>play-rand-pchval2</td>
<td>MIDI+</td>
<td>Modifies pitch (PLAY_RAND2)</td>
</tr>
<tr>
<td>play-rand-pit1</td>
<td>MIDI+</td>
<td>Sets grain pitch (PLAY_RAND)</td>
</tr>
<tr>
<td>play-rand-pit2</td>
<td>MIDI+</td>
<td>Sets grain pitch (PLAY_RAND)</td>
</tr>
<tr>
<td>play-samp-cpu1</td>
<td>1 - 8</td>
<td>Sets sample table number</td>
</tr>
<tr>
<td>playtot2</td>
<td>MIDI+</td>
<td>Modifies pitch (TREVOR)</td>
</tr>
<tr>
<td>ptof</td>
<td>0 - 127</td>
<td>Input from clarinet (FREQ.)</td>
</tr>
<tr>
<td>ptoh</td>
<td>0 - 127</td>
<td>Input from clarinet (HARM.)</td>
</tr>
<tr>
<td>pton</td>
<td>0 - 127</td>
<td>Input from clarinet (NOISE)</td>
</tr>
<tr>
<td>ptor</td>
<td>0 - 127</td>
<td>Input from clarinet (REVB)</td>
</tr>
<tr>
<td>rand-bang4-gate</td>
<td>0/1</td>
<td>Trigger PLAY_RAND control algorithm (see fig. C.9)</td>
</tr>
<tr>
<td>rand-bang8-gate</td>
<td>0/1</td>
<td>Trigger PLAY_RAND control algorithm (see fig. C.9)</td>
</tr>
<tr>
<td>rand-gliss-gate</td>
<td>0/1</td>
<td>Trigger PLAY_RAND control algorithm (see fig. C.9)</td>
</tr>
<tr>
<td>Revfb</td>
<td>0 - 127</td>
<td>Reverb feedback level (REVB.)</td>
</tr>
<tr>
<td>Rgate</td>
<td>0 - 127</td>
<td>Reverb input (REVB)</td>
</tr>
<tr>
<td>Rout</td>
<td>0 - 127</td>
<td>Reverb output (REVB)</td>
</tr>
<tr>
<td>Rto2</td>
<td>0 - 127</td>
<td>Reverb to Left speaker (SPAT.)</td>
</tr>
<tr>
<td>Rto4</td>
<td>0 - 127</td>
<td>Reverb to Right speaker (SPAT.)</td>
</tr>
<tr>
<td>rtof</td>
<td>0 - 127</td>
<td>Flange output mixed with FREQ out</td>
</tr>
<tr>
<td>Rton</td>
<td>0 - 127</td>
<td>Reverb input (NOISE)</td>
</tr>
<tr>
<td>Runsgate</td>
<td>0/1</td>
<td>Start/stop sampler control algorithm 1 (see fig. C.10)</td>
</tr>
<tr>
<td>scale-samps</td>
<td>0 - 127</td>
<td>Final output amplitude scaling for SAMP. 1, SAMP. 2 (SPAT), + SAMP input to REVB.</td>
</tr>
<tr>
<td>Variable</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Spatinc</td>
<td>0 - 127</td>
<td>Controls speed of table-driven stereo panning (see fig. C.29)</td>
</tr>
<tr>
<td>Spat0n</td>
<td>0 - 4</td>
<td>Start/stop spatializer control algorithm (see fig. C.29)</td>
</tr>
<tr>
<td>spatx</td>
<td>0 - 127</td>
<td>Left channel final output (SPAT)</td>
</tr>
<tr>
<td>spatXY-19</td>
<td>0/1</td>
<td>Start/stop spatializer algorithm (see fig. C.28)</td>
</tr>
<tr>
<td>spaty</td>
<td>0 - 127</td>
<td>Right channel final output (SPAT)</td>
</tr>
<tr>
<td>s-spat</td>
<td>0/1</td>
<td>Start/stop spatializer control of SAMP 1</td>
</tr>
<tr>
<td>s-spatR</td>
<td>0/1</td>
<td>Start/stop spatializer control of SAMP 1 input to REVB</td>
</tr>
<tr>
<td>start-23</td>
<td>0/1</td>
<td>Start/stop sampler control algorithm (see fig. C.14)</td>
</tr>
<tr>
<td>sto2</td>
<td>0 - 127, msec</td>
<td>Sampler 1 to L. speaker (SPAT.)</td>
</tr>
<tr>
<td>sto4</td>
<td>0 - 127, msec</td>
<td>Sampler 1 to R. speaker (SPAT.)</td>
</tr>
<tr>
<td>stof</td>
<td></td>
<td>Sampler 1 input (FREQ.)</td>
</tr>
<tr>
<td>stoh</td>
<td>120 1500</td>
<td>Sampler 1 input (HARM.)</td>
</tr>
<tr>
<td>stor</td>
<td></td>
<td>Sampler 1 input (REVB.)</td>
</tr>
<tr>
<td>t-precess</td>
<td>Any integer</td>
<td>Precess rate (TREVOR)</td>
</tr>
<tr>
<td>trev-note1</td>
<td>List</td>
<td>Note data (TREVOR)</td>
</tr>
<tr>
<td>trev-note2</td>
<td>List</td>
<td>Note data (TREVOR)</td>
</tr>
<tr>
<td>trevor-onset</td>
<td>0 - 10000 msec</td>
<td>Sample start time (TREVOR)</td>
</tr>
<tr>
<td>t-spat</td>
<td>0/1</td>
<td>Start/stop spatializer control of SAMP 2</td>
</tr>
<tr>
<td>t-spatR</td>
<td>0/1</td>
<td>Start/stop spatializer control of SAMP 2 input to REVB</td>
</tr>
<tr>
<td>t-start</td>
<td>Bang</td>
<td>Start (TREVOR)</td>
</tr>
<tr>
<td>t-stop</td>
<td>Bang</td>
<td>Stop (TREVOR)</td>
</tr>
<tr>
<td>tto2</td>
<td>0 - 127</td>
<td>Sampler 2 to L. speaker (SPAT.)</td>
</tr>
<tr>
<td>tto4</td>
<td>0 - 127</td>
<td>Sampler 2 to R. speaker (SPAT.)</td>
</tr>
<tr>
<td>ttoh</td>
<td>0 - 127</td>
<td>Sampler 2 input (HARM.)</td>
</tr>
<tr>
<td>tton</td>
<td>0 - 127</td>
<td>Sampler 2 input (NOISE)</td>
</tr>
<tr>
<td>ttor</td>
<td>0 - 127</td>
<td>Sampler 2 input (REVB)</td>
</tr>
<tr>
<td>t-transc</td>
<td>MIDI+</td>
<td>Microtonal variation (TREVOR)</td>
</tr>
<tr>
<td>t-transpose</td>
<td></td>
<td>Transposition level (TREVOR)</td>
</tr>
<tr>
<td>which1_table</td>
<td>1 - 8</td>
<td>Sample table (SAMP. 1)</td>
</tr>
<tr>
<td>which2_table</td>
<td>1 - 8</td>
<td>Sample Table (SAMP. 2)</td>
</tr>
</tbody>
</table>
REFERENCES


_______. Email correspondence with the author, January 4 – May 28, 2004.


