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."[2] 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.

---

[2] Rowe, Robert. *Interactive Music Systems: Machine Listening and Composing,* (Cambridge, MA: The MIT Press, 1993)

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 "Talharmonium," 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

---

[3] Peter Manning, *Electronic and Computer Music,* Revised and expanded edition (New York: Oxford University Press, 2004), 3 – 4.

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.[4] 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

[4] Manning, *Electronic and Computer Music,* 158.

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

---

[5] Manning, *Electronic and Computer Music,* 162.

[6] David Ernst, *The Evolution of Electronic Music* (New York: Schirmer Books, 1977), 162

[7] Ibid., 153

[8] Benny Sluchin, "A Computer-Assisted Version of Stockhausen's *Solo for Melody Instrument with Feedback," Computer Music Journal* 24, no. 2 (2000), 40

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.[9] 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).[10] Stockhausen's *Mantra* (1970) combines two pianos with shortwave radios, ring modulators, and multi-speaker sound projection.[11] 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

---

[9] Manning, *Electronic and Computer Music*, 158
[10] Ernst, *Evolution of Electronic Music,* 171
[11] Manning, *Electronic and Computer Music,* 160

music synthesizer.[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. These software-based instruments neatly avoid 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

[12] Manning, *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 [13] or Max Matthews and John R. Pierce.[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.[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.

---

[13] Eduardo Reck Miranda, *Computer Sound Design: Synthesis Techniques and Programming,* 2nd edition (Oxford: Focal Press, 2002)

[14] Max Matthews and John R. Pierce, ed., *Current Directions in Computer Music Research* (Cambridge, MA: The MIT Press, 1989)

[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.[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

---

[16] Samuel Pellman, *An Introduction to the Creation of Electroacoustic Music* (Belmont, CA: Wadsworth Publishing Company, 1994), 274

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

Narcissus (1987, considered in detail in chapters 5 and 9), Judith Shatin's *Sea of Reeds* (1997), and William O. Smith's *Asana* (1985).

2.5 THE MIDI STANDARD

MIDI (Musical Instrument Digital Interface) was created in 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.[17] 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

Status Byte	Data Byte 1	Data Byte 2	Interpreted as …
10011000 (152)	00111100 (60)	01000000 (64)	Note On, channel 8, Middle C, *mezzo forte*
10111000 (184)	01000000 (64)	01111111 (127)	Control Change on channel 8, Sustain Pedal, On
11001000 (200)	01111111 (127)	--	Change the synthesizer voice on channel 8 to preset 127
10111000 (184)	01000000 (64)	00000000 (0)	Control Change on channel 8, Sustain Pedal, Off
10011000 (152)	00111100 (60)	00000000 (0)	Note On, channel 8, Middle C, *silent* (effectively a "Note Off" message)

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 Pr*aescio 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.[18] 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.[19]

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.[20] 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-Driven Interactive Software," *Computer Music Journal* 15, no. 3 (1991), 16.

[20] Pennycook, "Machine Songs II," 23.

2.6.3 4X

The 4X machine was a powerful signal processing and synthesis workstation developed at IRCAM in the mid-1980s. 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.[21] 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

[21] Rowe, *Interactive Music Systems,* 21.

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-alone" 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.[22] 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

[22] Rowe, *Interactive Music Systems,* 25

> isolated from other methods, by virtue of their encapsulation in a surrounding object.[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.

---

[23] Rowe, *Interactive Music Systems,* 26

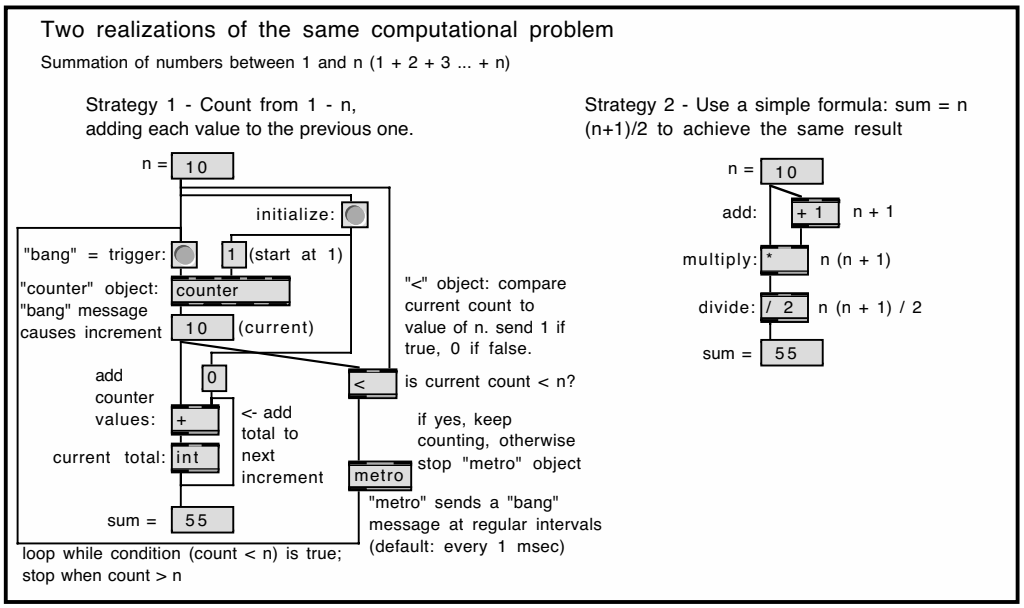

Figure 2.1. Simple mathematics implemented with Max objects

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:

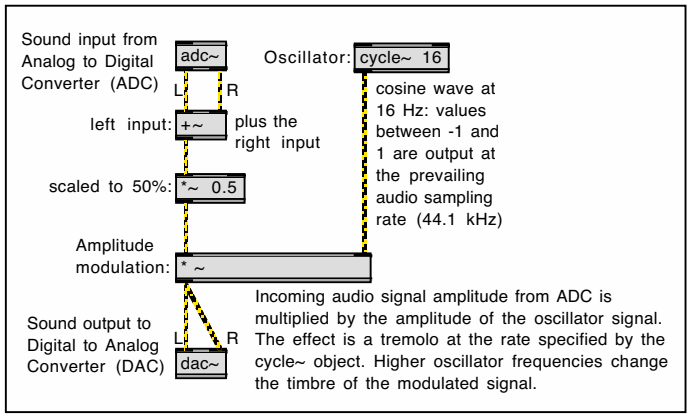

Figure 2.2. A simple signal processing function implemented with MSP objects

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

---

[24] "PD Community Site – About Pure Data," <http://puredata.info/about>, (accessed June 20, 2004)

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.[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.[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

---

[25] James McCartney, "SuperCollider FAQ" http://www.audiosynth.com/scfaq.html (accessed June 13, 2004).

[26] James McCartney, "SuperCollider: a new real time synthesis language" <http://www.audiosynth.com/icmc96paper.html> (accessed June 13, 2004). Originally given as a paper at the International Computer Music Conference in 1996.

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.[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.[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.[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 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

---

[27] Cort Lippe and Miller Puckette, "Musical Performance Using the IRCAM Workstation," *International Computer Music Conference, Montreal 1991 Proceedings* (Montreal: ICMC-1991, McGill University, 1991) 533-535

[28] Symbolic Sound Corporation, "What is Kyma?" <http://www.symbolicsound.com/cgi-bin/bin/view/Products/WhatIsKyma> (accessed June 13, 2004).

[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.