STUDIA UNIVERSITATIS BABEŞ-BOLYAI, MUSICA, LIV, 1, 2009 (p. 123-128)

IAC: AN INTERACTIVE MUSIC SYSTEM

ADRIAN BORZA[1]

SUMMARY. This paper aims to discuss different technical and artistic aspects of IAC software, with emphasis on interactive music system concept. Would not it be possible to create "intelligent" software that is able to "understand" the performer actions and to "follow" the score, right there on the stage? IAC facilitates the interactive composition based on programmed algorithms. The algorithms generate music structures during the ongoing performance with a MIDI keyboard connected to computer; they are designed to respond concurrently to changing pitch, intensity, and tempo.

Keywords. Interactive music system, music software programming, algorithmic composition, electronic music

Introduction

For more than 50 years composers and researchers have used computers in music in two main directions of development. One of these practices of making music refers to the integration of digital sound synthesis in musical works, and the other composition practice identifies the production of compositional algorithms.

Most of composers have been engaged in sound generation and audio processing by means of computer, in order to create new audio material for their own compositions. But some of composers have made efforts in building highly specific software, able to execute algorithms in real-time, onstage; in fact the computer algorithms were designed to change its actions according to musician input on the stage (Ph. Manoury and others). The recent 20 years illustrate the trend toward increased use of interactive music systems in performance and composition, due, in a large part, to the programming progress and the hardware power in this field of music technology.

Lately, the human/computer interaction has been extended to visual and dance applications. There are several iconic programming languages and software developed over the years for interactive audio, video, dance, multimedia and installations, such as Patcher (1986), M (1987), Jam Factory (1987), Max/ISPW (1989), Cypher (1989), Max/Opcode (1990), BioMuse (1992), Pure Data (1996), jMax (1996), Max/MSP (1997), EyesWeb (1997), vvvv (1998), Nato.0+55+3d (1999), Cyclops (2000), SoftVSN (2002), Max/MSP/Jitter (2003), and OMax (2004).

[1] Currently he is a professor at the Gheorghe Dima Academy of Music. Address: 25, I.C. Brătianu, Cluj-Napoca, Romania. E-mail: aborza@gmail.com.

1. Interactive Computer Music Systems

In a concert context, the synchronization between the performer score and the prerecorded music on tape has always been a problem in electroacoustic music, until the rise of interactive systems. An interactive system designed for computer music reacts instantly to the performer actions during the ongoing performance of the musical work. Therefore, the most important aspect of the interactive music system is its ability to adapt itself to changing situations all the way through performance. However, the interactive system is built offstage, investing cognition and time for programming the computer. The goal is to successfully implement into computer software various composition and performance techniques. The composer conceives "complete scripts for a performance situation in which the computer can follow the evolution and articulation of musical ideas".[2]

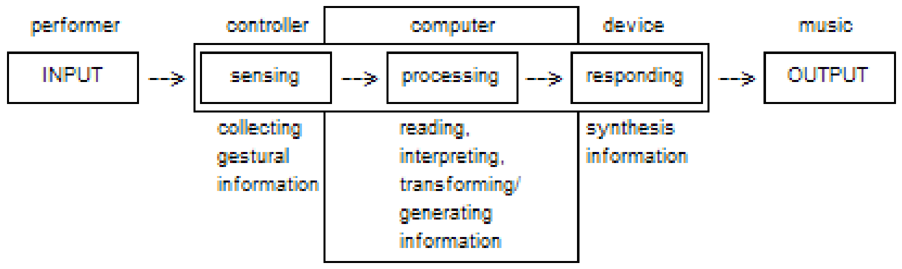
To put into practice such an objective, the composer formalizes the musical language through a programming language, assembling compositional algorithms that produce music with the computer. The algorithms are suitable to analyze data input, then to accomplish tasks as a reaction, at particular points in time, to analyzed data.

Furthermore, "by transferring musical knowledge to a computer program and compositional responsibility to performers onstage, however, the composer of interactive works explores the creative potentials of the new technology at the same time that he establishes an engaging and fruitful context for the collaboration of humans and computers".[3]

The chain transformation processes of the interactive computer music system (Figure 1) can be conceptualized in three stages: sensing, processing, and responding.[4]

Fig. 1

The chain transformation processes of interactive computer music system



[2] Rowe, Robert, *Interactive Music Systems: Machine Listening and Composing*, The MIT Press, Cambridge, Massachusetts, 1993, p. 5-6.

[3] Rowe, Robert, The Aesthetics of Interactive Music Systems, in *Contemporary Music Review*, 1999, Vol. 18, Part 3, p. 87.

[4] Rowe, Robert, *Interactive Music Systems: Machine Listening and Composing*, The MIT Press, Cambridge, Massachusetts, 1993, p. 10.

Memo

Interactive music system features: interactive (depends on input), works in real-time (reacts instantly to input), analyses (to data input) and reacts (producing data output), flexible (adapts itself to changing performance situation), algorithmic (uses compositional algorithms), formal system (represents the formalization of musical language).

2. IAC – Interactive Algorithmic Composition Music Software

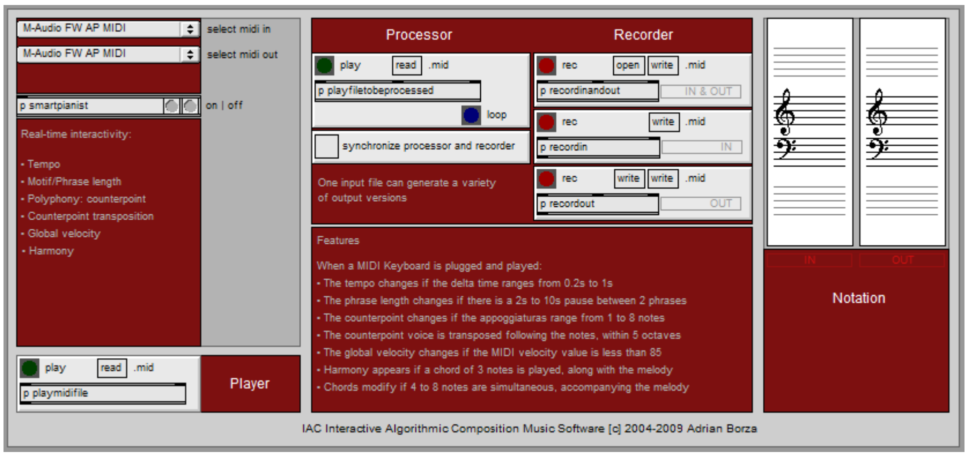
With my own software – IAC (Montreal, 2004) – built in Max language, I have composed several interactive works, ether playing and improvising on the MIDI keyboard connected to computer, or simply playing a MIDI file through the software processing module. Examples of musical works are *Fragile* (saxophone, viola, electronics, 2008), *Lay Low* (flute, violin, piano, cello, 2007), *Relocation* (saxophones, electronics, 2007), *Luxury of Loneliness* (interactive computer, 2007), *The Decipher* (interactive computer, 2007), *Today is the Day – 365 Concise Music Pieces* (interactive computer, 2007), *80 after Max* (interactive computer, 2006*), Music for MIDI Keyboard and Interactive Computer* (2005), and *Yak* (interactive computer, 2004).

2.1. Developing the user interface

The IAC graphic user interface (Figure 2) includes Max standard objects for universal commands that are familiar to commercial DAW software, commands from *play*, *stop loop*, *record*, *read*, *write* a file and music notation-based diagram to *open* editor, *reset* (panic) and *select* MIDI driver. This is exactly what the user sees on the screen, in the main window, and he interacts with it, being able to send information in a tactile manner, using the mouse, and accordingly to receive it, visually. On the other hand, the sound response to commands is instantaneous.

Fig. 2

The IAC graphic user interface



Real-time interactivity:

- Tempo
- Motif/Phrase length
- Polyphony: counterpoint
- Counterpoint transposition
- Global velocity
- Harmony

One input file can generate a variety of output versions

Features

When a MIDI Keyboard is plugged and played:
- The tempo changes if the delta time ranges from 0.2s to 1s
- The phrase length changes if there is a 2s to 10s pause between 2 phrases
- The counterpoint changes if the appoggiaturas range from 1 to 8 notes
- The counterpoint voice is transposed following the notes, within 5 octaves
- The global velocity changes if the MIDI velocity value is less than 85
- Harmony appears if a chord of 3 notes is played, along with the melody
- Chords modify if 4 to 8 notes are simultaneous, accompanying the melody

IAC Interactive Algorithmic Composition Music Software [c] 2004-2009 Adrian Borza

Developing the user interface, so as to be intuitive, easy to use, and to provide visual feedback, all the algorithms and programming details were hidden behind the graphic interface, in sub-patches. There are different control units implemented into this robust interface, designed 1) to read and play a file to be processed (*Processor*), 2) to record and save the performer data input, the software data output or to record and save both input and output data (*Recorder*), where the recording process can be in sync with the processor unit, 3) to play a MIDI file (*Player*), and 4) to illustrate musical notes events (*Notation*). By opening a special window, the user is presented with a complex editor and more (*detonate* Max object).

Memo

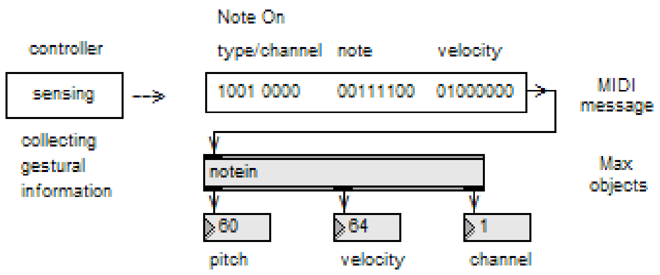
IAC user interface features: tactile (controlled by mouse), graphic (provides visual feedback), intuitive (perceived by intuition), efficient (organized and easy to use), encapsulated (algorithms are hidden in sub-patches).

3. Interpreting performer actions

Max provides a wide range of MIDI objects, which are specialized to interpret MIDI messages. For example, the most common MIDI message – *Note On* – is handled in Max by *notein* object that separates data into three elements – pitch, velocity, and channel. All the MIDI messages, received as binary digit flow from keyboard through the MIDI interface, are classified in Max in various types of data, and then are converted to integers (Figure 3). In other words, when the performer pushes and holds a key on instrument, *Note On* message is sent instantly to computer, in about 1 millisecond, the message is divided and converted to three integers.

Fig. 3

The data flow



Not only the channel voice messages, passing through the MIDI interface, are interpreted in Max, but the whole stream of serial data is analyzed, including, beside the MIDI note information (pitch, velocity, and channel)[5], the

[5] There are terms described in MIDI Specifications that reflects its origin in performance gesture and instrument control – *Note On*, *Key Pressure* etc. Moreover, the perceptive sound qualities associated to MIDI messages are described using pitch, intensity, timbre etc. For details see Borza, Adrian, *Muzică şi calculator (Music and Computer)*, Editura Muzicală, Bucharest, 2008, p. 128.

program change data, and the control change data (modulation, volume, pan, sustain etc).

Once the performer actions (alias MIDI messages) are converted to integers, *"it is quite easy to manipulate and process the numbers, using algorithms to create music"*.[6]

4. IAC features

A detailed explanation of IAC algorithms and data mapping are beyond the purpose of this paper, while I have tried to provide the reader with an introduction to interactive computer music systems. A basic description of the algorithms functionalities might be useful for those unfamiliar with the IAC software.

Performer/computer real-time interaction is noticeable in IAC on various levels when a MIDI Keyboard is plugged and played:

- *Tempo*
  The tempo is changed during performance if delta time (the number of milliseconds elapsed since the previous Note On event[7]) of the input data ranges from 0.2 second to 1 second. The tempo does not change according to the notes duration. The player has to be rhythmical accurate.
- *Phrase length, and silence*
  The length of the musical phrase, generated by computer, is changed during performance if the silence between 2 consecutive phrases played on the keyboard ranges from 2 seconds to 10 seconds
- *Polyphony: counterpoint*
  The micro-structure of the counterpoint voices, generated by computer, is changed during performance if the appoggiaturas played on the keyboard range from 1 to 8 notes (data mapping)
- *Counterpoint transposition*
  The generated counterpoint is transposed during performance if the notes played on the keyboard are within 5 pre-set octaves. The direction of transposition by a specific interval is governed by the note's pitch played at a particular time.
- *Global intensity*
  The global intensity of the audio material generated by computer is changed during performance if the input MIDI velocity value is less than 85. The global intensity is determined by the note's velocity played at a certain moment.

[6] Winkler, Todd, *Composing Interactive Music: Techniques and Ideas Using Max*, The MIT Press, Cambridge, Massachusetts, 1998, p. 64.
[7] Zicarelli, David, *Max User's Manual – Reference Manual*, Version 4.6, 2006, p. 77.

- *Harmony*
  Harmony of 2 to 3 structures is generated if a chord of minimum 3 notes is played. The chord is transformed during performance if 4 to 8 keys are simultaneously played on the keyboard.

Conclusion

Undoubtedly, the composer of the XXI century will be a musician trained for programming computers as well as for composing music. The increasing tendency for programming interactive music systems reinforces the need of interaction between performer and computer. Therefore, the interaction will dramatically reshape the way composers create computer music nowadays.

REFERENCES

Borza, Adrian, *Muzică şi calculator (Music and Computer)*, Editura Muzicală, Bucharest, 2008.

Rowe, Robert, *Interactive Music Systems: Machine Listening and Composing*, The MIT Press, Cambridge, Massachusetts, 1993.

Rowe, Robert, *The Aesthetics of Interactive Music Systems*, in: *Contemporary Music Review*, Vol. 18, Part 3, 1999.

Zicarelli, David, *Max User's Manual – Reference Manual*, Version 4.6, 2006, p. 77.

Winkler, Todd, *Composing Interactive Music: Techniques and Ideas Using Max*, The MIT Press, Cambridge, Massachusetts, 1998.