iFPH: Wireless Control of Sound

Adrian Borza

Abstract—A wireless interactive computer music system is a versatile tool for creating and performing music onstage. It combines diverse equipments and software, besides a computer, a control surface with integrated Wi-Fi technology and a wireless router. The computer listens to and reacts in real-time to the musician performance. This paper aims to discuss several technical and musical aspects of the iFPH system, with emphasis on interactive music system concept.

I. INTRODUCTION

For more than fifty years, composers have used computers in music in two main directions of development. One of the 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 the composers of electroacoustic music have been engaged in sound synthesis and signal processing by means of computer, in order to create new sounds for their compositions. But some of them have made efforts in developing complex systems, able to interact with musician and to execute compositional algorithms in real-time. In fact, the algorithms were designed to change its actions onstage according to the musician input. The recent twenty years illustrate the trend toward the use of interactive music systems in performance and composition, in the field of music technology, due to the increase of computing power and the progress in programming.

There are several iconic programming languages, hardware and software developed over the years for interactive applications, 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).

II. INTERACTIVE COMPUTER MUSIC SYSTEMS

In concert, the synchronization between the music performed by musician and the prerecorded accompaniment on tape has always been a problem in electroacoustic music, until the rise of interactive music systems. Designed for computer music, an interactive music system reacts instantly to the performer actions, during the ongoing performance of the musical work. Therefore, the most important feature of the interactive music system is its ability to adapt itself to the changing situations all the way through the live performance.

The software for interactive music systems is built offstage, investing cognition and time for programming the computer. The goal is to successfully implement into software various composition and performance techniques. “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.” [1]

By definition, “an interactive music composition and performance system is a real-time composing and sound-producing system which employs a synthesizer, a programmable computer, and at least one performance device […]. The system is interactive in that a user can direct aspects of the system’s production of music, as he or she hears it being produced, by use of the performance device.” [2]

We preserve the key-components of the interactive computer music system: performance device, programmable computer, and sound-producing device. Nevertheless, the complexity of the system is the result of interconnection of a myriad of distinct components.

The devices working together can be conceptually bundled into sequences or stages of the whole process of music production, in which information is transferred from one device to another. “The first is the sensing stage, when data is collected from controllers reading gestural information from the human performers onstage. Second is the processing stage, in which a computer reads and interprets information coming from the sensors and prepares data for the third, or response stage, when the computer and some collection of sound-producing devices share in realizing a musical output.” [3]

Interpreting Performer Actions in Max

Max[1] programming environment supplies several objects which are capable to analyze a live music performance, i.e. Musical Instrument Digital Interface (MIDI) messages. When a stream of data is sent to a computer from a MIDI keyboard controller, Max objects provide useful information about pitch, loudness, duration, and many other MIDI messages. The MIDI protocol doesn’t offer timbre information to produce sounds by its own synthesis routines. For this reason, the sound generator or the synthesizer is commonly an external hardware or a software synthesizer driven by MIDI, software known as Virtual Studio Technology Instrument (VSTi).

For example, a convenient way to extract information about pitch and loudness is the sophisticated *borax* object. But a simpler alternative would be to make use of the basic features of *notein*. This object listens to and then it reports different integers, corresponding to the Note On and Note Off messages of the MIDI data input stream. Note Off is indicated by a Note On event with velocity equals 0. Using the very same object, an analysis of pitch and loudness occurrence can produce valuable data about duration. Duration is the time measured from the beginning of a note and the end of the same note, which are considered two separate MIDI events, Note On and

[1] “For over twenty years, Max has been used by performers, artists, and composers to make cutting-edge work by connecting basic functional blocks together into unique applications.” (Cycling74)

Note Off. To calculate the time elapsed between MIDI messages, the *timer* object can be employed to execute the task. It delivers numbers representing duration in milliseconds and ticks – values correlated to tempo.

Definitely, there is musical information that can be transferred from the musician performance to the computer accompaniment. Once the performer actions are converted to numbers, "it is quite easy to manipulate and process the numbers, using algorithms to create music." [4]

Deciphering the temporal flow of input data consists in identifying and formulating the compositional problem, among others: to isolate a particular event or data from a series of musical events during the live performance, with the aim of triggering an action or process managed by the computer. For instance, the following statements highlight the relationship between cause and effect: any musical event, in low range, activates the playback mechanism; a melodic ascending interval, between C3 and C4 notes, enables the rhythmic augmentation and diminution; a minor chord, in mezzo-piano, triggers the melodic synchronization. Most of the time, discerning clearly and rigorously the matter should lead to find a suitable compositional algorithm. "Composer objects represent musical processes, known in computer music as compositional algorithms. […] A canon (round) is a very old compositional algorithm." [5]

Developing an efficient compositional algorithm it leads to solve the problem, by judiciously selecting internal and external Max objects which are designed to execute individual and precise tasks. Some objects are better than others, in terms of action and communication. Once selected, the objects are graphically interconnected, and then the patch is tested and debugged. This is a practical stage of the composition process, and all the composer's programming skills are emphasized. The composer will design and produce countless Max compositional algorithms for a single musical problem; the ideal would be to find the optimum. If there is an improper understanding of the problem or the composer makes a clumsy decision on selecting and connecting the objects, it may even affect the efficiency of the implementation. A general rule in programming is to find simple solutions to complex problems.

III. INTEGRATING WIRELESS CONTROL

A chain of diverse electrical and electronic devices interconnected, which is defined in this paper as a wireless interactive music system, is a powerful and versatile tool for creating and performing music onstage nowadays. Our system (Fig. 1) merges several pieces of hardware, such as a microphone, mixer, sound card, loudspeakers, audio control surface, computer, wireless router, and Wi-Fi mobile phone, but not exclusively.

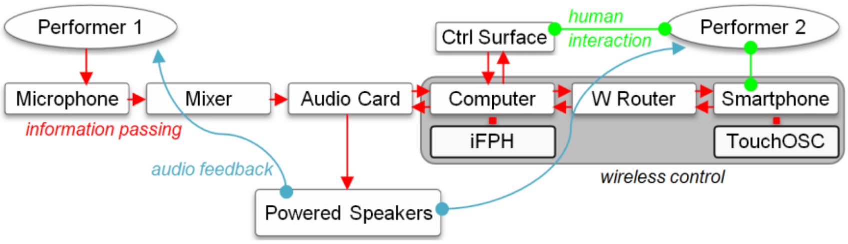

Figure 1. iFPH Wireless Interactive Music System Diagram

The system is built around a laptop driven by the iFPH[1] music software, yet it is flexible and portable. iFPH is coordinated by the Performer 2 through a smartphone or an audio control nanosurface. The Performer 1 is the actual instrument performer.

A. Computer Hardware and Software Configuration

We have effectively tested the following configuration of our system:

- Microphones type and model: cardioid and hyper-cardioid patterns, AKG C 1000 S, Sennheiser e 608

[1] The iFPH software is assembled with Max/MSP by Adrian Borza. iFPH stands for Interactive, Freezer, Player, Processor, and Harmonizer. The current version is 2.5 (2012)

- Mixer model: Behringer Eurorack MXB1002
- Microphone modeling preamplifier model: Behringer MiniMic MIC800
- Sound cards type and model: portable interfaces with FireWire and USB ports, M-Audio Audiophile, M-audio Fast Track C400
- Powered speakers type and model: studio reference, active near-field monitors, Wharfedale Pro Diamond 8.2 Pro Active, Yamaha HS50M
- Computer model and operating system: HP ProBook 4530s, Windows 7 64-bit
- Computer applications: Max 5.x, 6.0.x, Max Runtime 5.x, 6.0.x, iFPH 2.5
- Audio control nanosurface type and model: Korg Slim-line USB Control Surface, nanoKontrol2
- Wireless router type and model: TP-Link 150Mbps Wireless N Nano Router, No TL-WR702N
- Smartphone type, model and operating system: Samsung Galaxy S II, No GT-I9100, Android 4.0.3
- Virtual audio control surface: TouchOSC 1.3

The core of the iFPH wireless interactive music system includes input transducers, such as the microphone and the smartphone, processors, e.g. the computer and the iFPH software, and output transducer, such as the pair of loudspeakers.

B. Stages of Information Passing

We identify four stages of the process of transferring the information from a device to another: sensing, processing, response, and feedback.

1) Sensing stage

The sound coming from the Performer 1 is captured by the microphone, preamplified by the mixer, digitized by the sound card, and sent to the iFPH software. The Performer 2 interacts directly with the TouchOSC[1]'s user interface on the

[1] TouchOSC is a virtual audio control surface for iOS and Android operating systems, developed by Hexler

smartphone's touch screen. The device is capable of reading the tactile gestures of the performer. The TouchOSC application sends Open Sound Control[1] (OSC) messages to the iFPH software, through the wireless router. The gestural information of the Performer 2, collected and converted by the control nanosurface, is sent to the iFPH software as MIDI messages.

2) Processing stage

There is a software module built into the iFPH software – Wireless Control – which is programmed to receive and transmit OSC messages. The module has the role to bind the OSC data to the processing data sets. These sets are stored into the iFPH's Presets module. The iFPH software interprets as well the MIDI data coming from the Performer 2 and binds these data to the processing data sets, stored into the Presets module.

3) Response stage

As an immediate result, the iFPH software puts into practice its transformational abilities – successive processes and manipulations of the input sound – then sends its processed sound to the self-powered speakers, back through the sound card.

4) Feedback stage

The Wireless Control module sends back the OSC messages towards TouchOSC, through the wireless router. The messages are exposed visually on the smartphone, thus confirming the Performer 2's gestures. The iFPH software sends back MIDI messages towards the control nanosurface for the same reason. Both musicians will find audio feedback of their performance, while it being produced, by listening to music through the loudspeakers.

C. iFPH Wireless Control Module

The grid of processing data sets, called Wireless Control, is embedded into a Max patch file (Fig. 2). The patch has been

[1] OSC protocol has been developed at the Center for New Music and Audio Technologies (CNMAT) at the University of California in Berkeley

conceived with the purpose of offering more flexibility in using the iFPH software during concert. We will describe the functionality of the patch in detail.

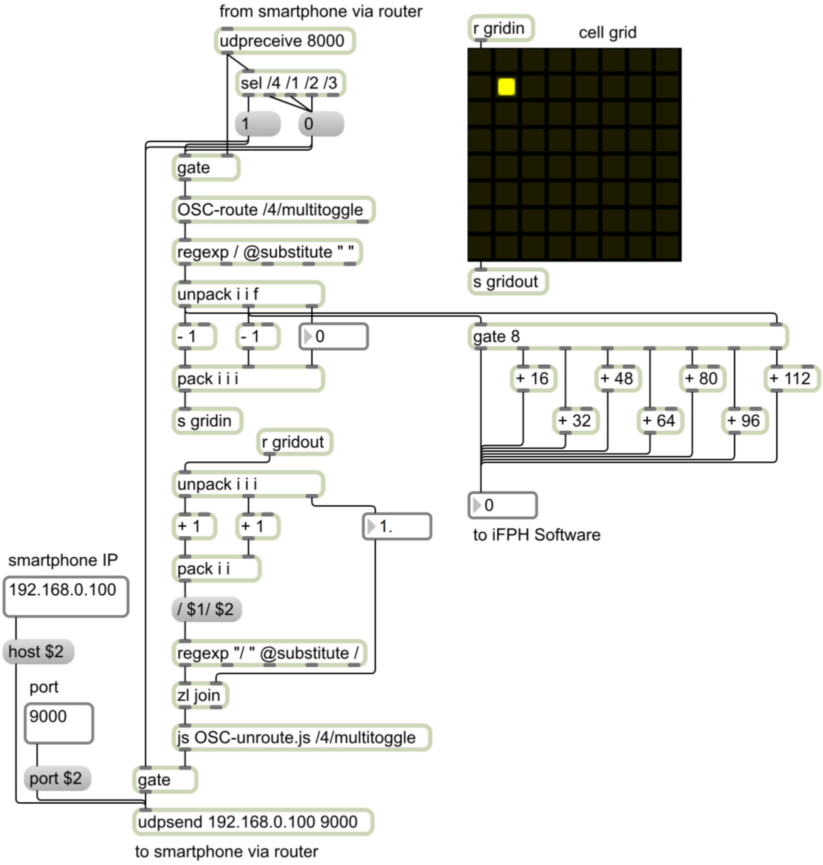

Figure 2. iFPH Wireless Control Patch

The *udpreceive* object works in conjunction with the OSC protocol, and it receives messages transmitted over a wireless network using User Datagram Protocol (UDP). The argument

8000 is the port number for receiving OSC messages from the smartphone via the wireless router.

The *sel* and *gate* objects are working together in order to bypass OSC data only when the page four is selected on the TouchOSC's user interface. Otherwise the *gate* object will stop the communication.

The *OSC-route*[1] object matches *multitoggle* element on the page four, as instructed into its argument, */4/multitoggle*, then it passes any OSC message matched before. The message represents a cell, defined by column, row, and the on/off state of the *multitoggle*.

The special *regexp* object substitutes the solidus symbol within the matched OSC message, and then the list message is divided into individual integer messages by the *unpack* object.

At this moment, the *gate* object will pass the column number only through the output associated to the specified row number, on one hand. The argument *8* is the number of gates, seen as the number of each row. The *+ (addition)* objects bind the column numbers to the preset numbers stored into the iFPH's Presets module. On the other hand, the *- (subtraction)* and *pack* objects will prepare the messages for the *matrixctrl* object. This object is a cell grid which it resembles the TouchOSC's multitoggle interface element. Any action on the *multitoggle* will cause a response on the *matrixctrl*'s interface. In other words, any musician's gestural action on the smartphone causes a visual feedback on the computer; and vice versa.

The processes are constructed then in reveres: the column, row, and state values of the *multitoggle* object are joined into an OSC message, so as to be transmitted through the *OSC-unroute.js* and *udpsend* objects. The argument *192.168.0.100* is the Internet Protocol (IP) address of the smartphone, and the argument *8000* is the port number for sending OSC messages to the smartphone via the wireless router. The arguments are changeable.

[1] *OSC-route* and *OSC-unroute.js* are Max external objects developed by Mat Wright at CNMAT

D. iFPH Music Software

The modules' name and features of the current version of the iFPH software are briefly described below.

High-pass Filter: the input audio signal is filtered according to the cutoff frequency, attenuating the amplitude of unwanted low frequencies but passing higher frequencies. The visible range of the filter on the GUI is from 20 Hz to 1 kHz.

Freezer: a sound grain is extracted from the input signal and after that is looped, layered and enriched, creating and sustaining a derivate sound. The overall amplitude of the new sound is static or dynamic, depending on the operation mode selected in the Envelope Follower for Freezer module. Up to eight different new sounds can be superimposed. The synthesis method is granular.

Envelope Follower for Freezer: the overall amplitude of the input audio signal is instantly applied to the audio signal generated by the Freezer module, personalizing the derivate sound. This means that the module reacts to music dynamics, and the input envelope is monitored constantly.

Interactive Player: pre-recorded sounds or audio files are played back when the overall amplitude of the input signal reaches or exceeds a specified threshold. The files are organized into the Playlist module.

Playlist: this is a GUI editor for organizing sound files, built with the purpose of play them back in sequential order, by using the Interactive Player module. The file types are AIFF and WAV.

Recorder: the input audio signal is recorded onto a memory buffer, and then is automatically segmented in small fragments. The segmentation process is based on the signal's overall amplitude. When the amplitude attains or decreases a specified threshold, the segment that lies between two threshold points is selected and ready to play it back. The maximum recording length is ten seconds, and the number of audio segments is limited to four.

Autonomous Player: each time an audio segment is played back, random panning and transposition is applied to it. This is especially true if Auto and Loop modes are on.

Harmonizer: performs a FFT and pitch-shifting in frequency domain of the input signal, building a chord of three sounds. The transposition range is from -24 semitones to +24.

Effects: in Wet operation mode, the input audio signal is processed by adding effects, such as distortion, chorus, phaser or flanger.

Reverb: the input/output signal is processed by adding reverb effect.

Metronome: this module is a sixty Beats per Minute (BPM) visual click track, intended to help musicians at the time they practice and perform on the stage.

Crossover: the output audio signal is split into two frequency bands and then routed to the first three channels of the sound card, achieving a simple, but productive bass management.

Presets: all the processing data sets (sets of control data over the sound) are saved into a file and called anytime later at the musician convenience. The Presets module is controlled by the smartphone or the audio control nanosurface.

Wireless Control: this is a grid of presets, a network connection setup, and a module for sending, converting and receiving OSC messages. It is a Max patch for wireless control over the processing data sets. The module is controlled through the smartphone.

The key-modules Freezer, Envelope Follower for Freezer, Interactive Player, Recorder, Autonomous Player, Harmonizer and Presets are controlled by the audio control nanosurface, in addition by using a mouse and a computer keyboard.

IV. CONCLUSION

The nonlinear access to the iFPH's processing data sets reflects the universal approach in live computer composition and performance which fits a musician, in the sense that the Performer 2 would prefer to switch anytime to any processing preset, represented visually on the smartphone's touch screen. Spontaneous creation of music onstage is a natural process. Improvisation is unpredictable, as we all know. Therefore, the flexibility in choosing one or other processing data set, with the purpose of transforming the sound produced by the Performer

1, is an advantage during spontaneous creation and music improvisation.

We have experienced this approach when we have presented publicly some of the musical works composed and programmed by ourselves: *If for Oboe and Interactive Computer* (2011), *Akedia for Oboe, nanoKontrol and iFPH* (2011), and *Les objets informatiques for Voice, Smartphone and Interactive Computer* (2012).

A wireless interactive music system appears as being a remarkable computer tool for making music, thus the role of the system is that of a musical instrument, and has to be mastered by musician. We strongly believe that the musician of the XXI century will be trained not only for composing music, but also for programming computers, as well as for performing music. The increasing tendency for programming interactive music systems reinforces the need of interaction between musician and computer. The interaction will dramatically reshape the way composers and performers will create and recreate computer music in future. But what is beyond these roles?

"If we are not capable of creating roles for the computer which go beyond our own image, perhaps we can at least attempt a synthesis of all these anthropomorphic and musical roles. This hybridization would at least offer us a glimpse of other functionalities, other conceptual frameworks, and can only serve to enrich the musical discourse and interactivity of real-time computer music." [5]

REFERENCES

[1] R. Rowe, "The aesthetics of interactive music systems", in Contemporary Music Review, vol. 18, part 3, pp. 87, 1999.

[2] J. Chadabe, "Interactive composition and performance system", US Patent No. 4,526,078, July 1985.

[3] R. Rowe, Interactive Music Systems: Machine Listening and Composing. Cambridge: The MIT Press, 1993, pp. 10.

[4] T. Winkler, Composing Interactive Music: Techniques and Ideas Using Max. Cambridge: The MIT Press, 1998, pp. 64.

[5] T. Winkler, ibid., pp. 173.

[6] C. Lippe, "A look at performer/machine interaction using real-time systems", Hong Kong ICMC Proceedings, pp. 117, 1996.