

Prolegomene la Sisteme muzicale interactive

Prolegomena to Interactive Music Systems

Adrian Borza

Academia de Muzică „Gheorghe Dima” / Academy of Music “Gheorghe Dima”

Cluj-Napoca, Cluj

aborza@gmail.com

REZUMAT

Acest studiu intenționează să discute despre conceptul de sistem muzical interactiv. Calculatorul, dotat cu un program „inteligent”, „înțelege” acțiunile interpretului și „urmărește” partitura acestuia, fiind capabil să acompanieze interpretul, să transforme și să genereze muzică, în timpul interpretării în desfășurare. Prezintă câteva exemple de algoritmi compozitionali, cu scopul de a ilustra metode și tehnici de programare în Max/MSP.

Cuvinte cheie

Sistem muzical interactiv, programare muzicală, compoziție algoritmică, muzică electronică

INTRODUCERE

De mai bine de 50 de ani, compozitorii și cercetătorii au folosit calculatorul în muzică, în două direcții importante de dezvoltare componistică. Una dintre aceste practici de a compune muzică se referă la integrarea în lucrări muzicale a sintezelor numerice a sunetului, iar cealaltă practică de compoziție se raportează la producția de algoritmi compozitionali.

Cei mai mulți compozitori de muzică electroacustică au fost angajați în generarea și prelucrarea sunetului prin intermediul calculatorului, cu scopul de a crea o nouă materie sonoră pentru compozițiile lor muzicale. Dar, unii compozitori au făcut eforturi în conceperea de programe de calculator cu totul speciale, capabile să transforme ori genereze muzică, pe scenă, să execute în timp-real algoritmi compozitionali; de fapt, algoritmi programului au fost proiectați în așa fel încât să-și modifice instantaneu acțiunile, în funcție de contribuția sonoră a muzicianului de pe scenă.

Philippe Manoury este unul dintre compozitorii care, spre sfârșitul anilor '80, a folosit un sistem muzical interactiv pentru elaborarea unor lucrări muzicale, printre care *Jupiter* pentru flaut și dispozitiv electroacustic în timp-real și *Pluton* pentru pian și dispozitiv electroacustic în timp-real. Sistemul interactiv, bazat pe 4X (pentru DSP), era coordonat de un calculator Macintosh pe care rula Patcher (pentru MIDI), primul ascendent al familiei de limbaje Max, realizat de Miller Puckette la IRCAM.

Încă de acum aproape 20 de ani, se manifestă o creștere a înglobării sistemelor muzicale interactive în interpretare și compoziție, datorită, în mare măsură, a progresului acestui domeniu al tehnologiei muzicale, în materie de programare și performanță a echipamentelor.

De curând, interacțunea în timp-real om/calculator a fost extinsă în domeniul vizual și în dans. Există câteva emblematice limbaje de programare și programe de calculator dezvoltate pentru aplicații interactive audio, video, multimedia, dans și instalații, cum ar fi 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).

ABSTRACT

This paper aims to discuss the interactive music system concept. An operational computer with “intelligent” software “understands” the performer actions and “follows” the score, being able to accompany the soloist, to transform the sound, and to generate music, during the ongoing performance. It provides the reader with compositional algorithms for the purpose of illustrating Max/MSP programming methods and techniques.

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 them 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 the musician input on the stage.

Philippe Manoury is one of the composers who have used in the late 80's an interactive music system to produce his musical works, including *Jupiter* for flute and real-time electroacoustic device, and *Pluton* for piano and real-time electroacoustic device. The interactive system, based on the 4X (for DSP), was coordinated by a Macintosh computer that ran Patcher (for MIDI), the first ascending of the Max languages family, made by Miller Puckette at IRCAM.

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 real-time 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).

(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) și OMax (2004).

SISTEME MUZICALE INTERACTIVE PE CALCULATOR

În condiții de concert, sincronizarea dintre partitura interpretului și muzica preînregistrată pe bandă a fost mereu o problemă în muzica electroacustică, până la apariția sistemelor interactive. Un sistem interactiv, conceput pentru muzica pe calculator, reacționează imediat la acțiunile interpretului, în timpul interpretării în desfășurare a lucrării muzicale. Apreciem că cel mai important aspect al sistemului interactiv este abilitatea acestuia de a se adapta situațiilor schimbătoare din actul interpretării, care apar, în mod firesc, odată cu redifuzările lucrării, dar mai ales în cazul creației și improvizării spontane.

Cu toate acestea, sistemul interactiv este construit în afara scenei, investind cunoștere și timp în programarea calculatorului. Obiectivul este de a implementa cu succes în programul de calculator diverse tehnici de compoziție și interpretare, utile elaborării muzicii. Compozitorul concepe „schite complete pentru un context interpretativ în care calculatorul poate să urmeze evoluția și articularea ideilor muzicale” (Rowe, 1993).

Pentru a pune în practică un astfel de obiectiv, compozitorul formalizează limbajul muzical prin intermediul unui limbaj de programare, asamblând algoritmi compozitionali, care produc, prin transfer de mesaje și date, muzică. Algoritmii sunt adecvați pentru analiza datelor de intrare, a informațiilor gestuale și audio captate de la interpret. Algoritmii mai sunt potriviti să îndeplinească sarcini drept răspuns, în anumite momente desfășurate în timp, la datele analizate, să transforme și să genereze materie sonoră.

În plus, „prin transferul cunoșterii muzicale către un program de calculator și a responsabilității compozitionale către interpreți de pe scenă, compozitorul de lucrări interactive explorează potențialul creativ al noii tehnologii și, în același timp, stabilește un context atractiv și fertil de colaborare între om și calculator” (Rowe, 1999).

Procesele transductive ale sistemului muzical interactiv pe calculator (Figura 1) pot fi conceptualizate în trei etape: captarea, prelucrarea și reacția (Rowe, 1993).

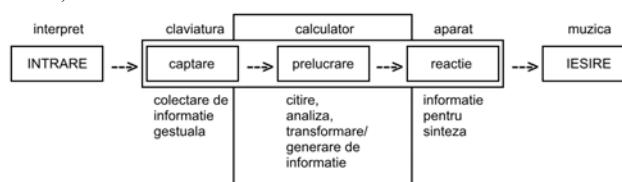


Figura 1. Procesele transductive ale sistemului muzical interactiv

Memo

Caracteristici ale sistemului muzical interactiv: interactiv (condiționat de intrare), funcționează în timp real (reacționează instantaneu), flexibil (se adaptează la situațiile schimbătoare ale actului de interpretare), analizează (datele de intrare) și răspunde (producând date de ieșire), algoritmice (folosesc algoritmi compozitionali), sistem formal (reprezintă formalizarea limbajului muzical).

INTERACTIVE COMPUTER MUSIC SYSTEMS

In a concert context, the synchronization between the performer's score and the prerecorded music on tape has always been a problem in electroacoustic music, until the rise of the 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, which occur naturally each time the work is performed, especially when the performer spontaneously improvises and composes on stage.

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”. (Rowe, 1993).

To put into practice such an objective, the composer formalizes the musical language through a programming language, assembling compositional algorithms that produce music. The algorithms are suitable to analyze data input, then to accomplish sound transformation and music generation 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”. (Rowe, 1999).

The chain transformation processes of the interactive computer music system (Figure 1) can be conceptualized in three stages: sensing, processing, and responding. (Rowe, 1993).

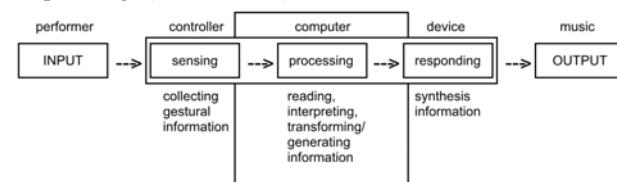


Figure 1. The chain transformation processes of interactive music system

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).

INTERPRETING PERFORMER ACTIONS

Designed at Institute de Recherche et Coordination Acoustique/Musique (IRCAM), and further developed in companies like Opcode and Cycling '74, Max/MSP/Jitter programming language, summarized Max in this document, is considered, by the software developers community, lingua franca of the XXI Century for

INTERPRETAREA ACTIUNILOR MUZICIANULUI

Concepțul la Institutul de ceretare și coordonare acustică/muzică (IRCAM) și dezvoltat ulterior la companiile Opcode și Cycling '74, limbajul Max/MSP/Jitter ori, pe scurt în acest document, Max, este considerat, de către comunitatea artiștilor programatori, lingua franca a secolului XXI pentru aplicații interdisciplinare de creație și interpretare, care implică interacțiune cu text, sunet, imagine, video și gest, în mod deosebit pentru conceperea programelor muzicale interactive și pentru compoziția muzicală interactivă.

Max oferă o colecție largă de obiecte informative MIDI, care sunt specializate în a interpreta mesajele MIDI. De exemplu, cel mai uzuial mesaj MIDI – *Note On* – este tratat în Max de obiectul *notein*, care separă datele în trei elemente – numărul tastei, valoarea presiunii de atac și numărul canalului. De altfel, toate mesajele MIDI, recepționate de la instrumentul cu claviatură prin interfața MIDI ca un flux de numere binare, sunt clasificate în Max în diferite tipuri de date și apoi sunt convertite în numere întregi (Figura 2). Altfel spus, atunci când interpretul apasă și ține apăsată o tastă, constant și indefinit, pe claviatura instrumentului MIDI, gestul este captat de instrument, iar mesajul *Note On*, produs la acțiunea interpretului, este trimis imediat spre calculator, în aproape 1 milisecundă; calculatorul „ascultă” și analizează mesajul *Note On*, apoi acesta este divizat și convertit, de obiectul *notein*, în trei mesaje numerice de tip *int* (întreg), care reprezintă valori de înălțime și intensitate ale sunetului, respectiv numărul canalului MIDI.

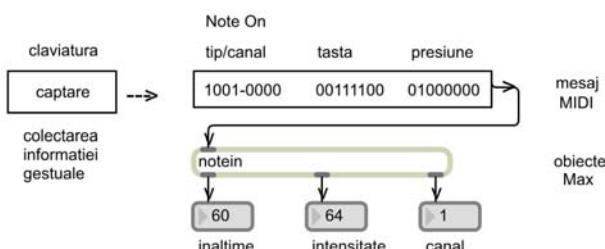


Figura 2. Flux de date numerice. Mesajul *Note On* (3 octeți)

Dacă exemplul de mai sus înfățișează anatomia trasmiterii unui singur mesaj MIDI – *Note On*, format din 3 octeți –, realitatea arată ca fluxul de date numerice, produs în timpul interpretării la un instrument MIDI, este complex, divers și transmis în mare viteză (rata de transfer în comunicarea MIDI este 31,25 Kbaud, adică 31 250 biți pe secundă). Așadar, nu doar mesajele canal de parametrare, care trec prin interfața MIDI, sunt interpretate în Max, ci întregul torrent de date seriale este analizat în această etapă de prelucrare din lanțul proceselor transductive, incluzând, pe lângă informațiile legate de nota MIDI (tastă, presiune și canal), numărul programului asociat timbrului, parametrii auxiliari (modulație, volum, panoramare, susținere etc) și datele sistem exclusive. Câteva dintre obiectele informative din Max specializate în operarea cu aceste tipuri de mesaje MIDI sunt: *pgmin*, *ctlin*, respectiv *sysexin*.

Există termeni descriși în normele MIDI care reflectă originea lor în gestica interpretativă și controlul instrumentului – *Tastă apăsată*, *Presiune individuală de apăsare* etc. Calitățile perceptive ale sunetului asociate

interdisciplinary applications, involving interaction with text, sound, video and gesture.

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 2). In other words, when the performer pushes and holds a key on instrument, constantly and indefinitely, the performer gesture is captured by instrument, producing a *Note On* message that is sent instantly to computer, in about 1 millisecond, then the message is divided and converted to three integers.

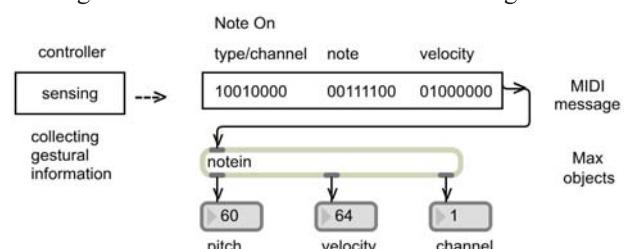


Figure 2. The data flow. *Note On* message

If the above example depicts the anatomy of a single MIDI message – *Note On* of 3 Bytes –, the reality shows that the data flow produced during performance is complex, diverse, and high-speed broadcast (the transfer rate in MIDI communication protocol is 31,25 Kbaud or 31,250 bits per second). Therefore, not only the channel voice messages are interpreted in Max, but the whole stream of serial data is analyzed, including, beside the MIDI note information (pitch, velocity, and channel), the program change data, the control change data (modulation, volume, pan, sustain etc), and the system exclusive data. Some of the Max objects specialized in interpreting these types of MIDI messages are: *pgmin*, *ctlin*, and *sysexin*.

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. (Borza, 2008).

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”. (Winkler, 1998). The difficulty is to analyze the information rather to process it.

MESSAGE PASSING AND DATA FLOW

The flow of time is essential to interactive music, as music is considered a temporal art. The question is: how the composer decodes, from a musical perspective view, the features of the message and data flow, as a result of interpreting the performer actions?

Deciphering the temporal flow consists, among others, in identifying and formulating the compositional problem: to isolate in Max a particular event from a series of musical events during the ongoing performance, with the aim of triggering an action or a process. Most of the time, discerning the matter, clearly and rigorously, should lead to find a suitable solution.

mesajelor MIDI sunt exprimate prin înălțime, intensitate, timbru și.a.m.d. (Borza, 2008)

Odată ce acțiunile interpretului (alias mesajele MIDI) sunt convertite în numere întregi, „este chiar simplu de a manipula și prelucra numerele, folosind algoritmi pentru compunerea muzicii” (Winkler, 1998). Dificultatea constă, mai degrabă, în analiza informației, decât în prelucrarea acestora.

FLUXUL DE MESAJE ȘI DATE NUMERICE

Fluxul temporal este indispensabil muzicii interactive pe calculator, după cum muzica este considerată o artă temporală. Se pune întrebarea: cum poate compozitorul să izoleze, din perspectivă muzicală, caracteristicile fluxului temporal de date numerice, rezultat, după cum am arătat, prin interpretarea în Max a acțiunilor interpretului.

Descifrarea fluxului constă, printre altele, în identificarea și formularea problemei componistice: izolare, pe calculator, a unui sau unor evenimente muzicale particulare, din cadrul unei serii de evenimente, din actul de interpretare în desfășurare, cu scopul declanșării unei acțiuni de comandă și control ori a unui proces de transformare a materialului muzical captat. De cele mai multe ori, o discernere (clară, riguroasă) a problemei, conduce la găsirea unei soluții potrivite.

Modalitatea de abordare a problemei componistice este, fără îndoială, algoritmică, prin controlul deducției logice, urmărind elaborarea algoritmului compozitional, adică a metodei ori setului de reguli, care acționează asupra unor tipuri de date cu caracter muzical ale fluxului temporal.

Elaborarea eficientă a algoritmului reprezintă rezolvarea problemei, și anume, selectarea judicioasă a celor obiecte interne sau externe din Max, care sunt proiectate să îndeplinească sarcini individuale precise. Unele obiecte sunt mai eficiente decât altele în termeni de operație și comunicare, în condițiile problemei date. Odată selectate, urmează interconectarea obiectelor, testarea algoritmului pe calculator și corectarea erorilor. Această etapă din procesul de compoziție este cu caracter practic, în care toate abilitățile de programare ale compozitorului sunt puse în valoare. Se pot concepe și realiza în Max nenumărați algoritmi compozitionali pentru problema componistică dată; ideal ar fi să fie găsit cel optim. Dacă apare o formulare nepotrivită, o decizie stângace în selectarea și conectarea obiectelor Max, acestea pot îngreuna rezolvarea problemei componistice și pot chiar afecta viteza de calcul – vitală pentru muzica interactivă. O regulă generală în programare este găsirea unor soluții simple pentru probleme complexe.

ALGORITMI COMPOZITIONALI

„Obiectele compozitorului reprezintă procesele muzicale, cunoscute în muzica pe calculator sub numele de algoritmi compozitionali. [...] Un canon este un vechi (exemplu de – n.a.) algoritm compozitional.” (Winkler, 1998)

Sistemul muzical interactiv are intrinsec algoritmi compozitionali, iar interfața cu utilizatorul și algoritmii de sinteză vin să îl împlinească. O dezbatere în detaliu a problematicii algoritmilor compozitionali depășește scopul acestui document, atâtă timp cât am încercat să ofer o introducere la sistemele muzicale interactive pe calculator. Descrierea algoritmilor exemplificați în acest

The approach to the compositional problem is algorithmic, without doubt, by logical inference control, in order to develop the compositional algorithm that is the method or set of rules, which acts on various types of data.

Developing an efficient algorithm it leads to solve the problem, i.e. 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 compositional algorithm 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 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 computer processing speed – vital for interactive music. A general rule in programming is to find simple solutions to complex problems.

COMPOSITIONAL ALGORITHMS

“Composer objects represent musical processes, known in computer music as compositional algorithms. [...] A canon (round) is a very old compositional algorithm.” (Winkler, 1998)

The interactive music system has inherent compositional algorithms; however, the graphic user interface and the sound synthesis algorithms will fulfill it. A comprehensive debate of compositional algorithms is beyond the purpose of this document, while I have tried to give an introduction to interactive computer music systems. The algorithms embodied into this study could be useful to those unfamiliar with Max programming.

The following compositional algorithms are designed to react in real-time, interactive to unpredictable events, such as music improvisation and spontaneous composition. Another feature of the compositional algorithms is mapping; the performer action is associated to the computer process, and one musical parameter transforms another parameter. For example, the pitch is captured and it changes the duration, the intensity is intercepted and it modifies the delta time, and so on. Delta time refers to the number of milliseconds elapsed since the previous Note On event (Zicarelli, 2006). In this regard, see *B. Mapping algorithm* that increases and decreases the speed of prerecorded sequence on computer, by means of the musical interval size, performed by soloist.

I chosen to discuss in detail three basic compositional algorithms implemented in Max, which speak about unpredictable temporal message and data flow. The following examples illustrate my personal programming style and experience.

Practical problems

When identifying into a MIDI data stream

- any musical event, in the low range,
- a melodic ascending interval, between C 3 and C 4,
- a minor chord, in mp,
- It must trigger
- the playback of the prerecorded musical sequence

studiu ar putea fi utilă celor nefamiliarizați cu programarea în Max.

Algoritmii compozitionali sunt proiectați să reacționeze, în timp-real și interactiv, la evenimente imprevizibile, cum este cazul improvizației și creației spontane. Lucrurile se prezintă diferit când evenimentele sunt previzibile și sunt notate în partitura interpretului, iar algoritmii răspund la o valoare de înălțime, de durată, de intensitate ori la o combinație de parametri și valori, dintr-o succesiune de date prestabilite și stocate într-o colecție de date de intrare. Algoritmii de mai jos reflectă însușirea fluxului temporal de a fi imprevizibil.

O altă caracteristică a algoritmilor compozitionali este regula de corespondență de transformare, prin care se asociază o acțiune a interpretului de acțiunea algoritmului de prelucrare, iar parametrul muzical transformă un alt parametru corespondent diferit. De exemplu, o înălțime captată transformă o durată, o nuanță interceptată transformă timpul delta și.a.m.d. Timpul delta semnifică numărul de milisecunde care a trecut de la evenimentul *Note On* precedent (Zicarelli, 2006). În acest sens, vezi mai jos algoritmul B. Mapping, care augmentea și diminuează ritmul unui fragment preînregistrat, prin mărimea intervalului muzical interpretat la instrument.

Am ales spre exemplificare câțiva algoritmi implementați în Max, care vor fi discutați pe larg în cele ce urmează și se referă la analiza și structurarea în program a fluxului temporal imprevizibil de mesaje și date numerice. Detaliile de programare ilustrează experiența și stilul de programare ale autorului, cu privire la limbajul Max/MSP/Jitter.

Probleme practice

La momentul identificării, într-un flux de date MIDI de intrare, a unui

- eveniment oarecare, în registrul grav
- interval melodic, ascendent, situat între do 3 și do 4
- acord minor, în nuanță mp
- trebuie provocată
- difuzarea unui fragment muzical preînregistrat
- augmentarea/diminuția ritmică a unei structuri ritmico-melodice preînregistrate
- sincronizarea, eveniment cu eveniment, a acordului minor cu un fragment melodic preînregistrat (sincronizare a tempo-ului, duratei și intensității sunetului)

A. Controlul difuzării

1. Un eveniment oarecare, în registrul grav,
2. declanșează difuzarea unui fragment muzical preînregistrat

Acest algoritm compozitional (Figura 3) are un grad ridicat de generalitate, prin aceea că răspunde unei categorii largi de probleme de sincronizare între interpret și calculator. Aceasta se referă la controlul difuzării oricărui fragment muzical preînregistrat, în format MIDI, prin intermediul unui eveniment oarecare – un sunet, un interval melodic sau armonic, o structură ritmico-melodică, un acord, un cluster etc –, situat în registrul grav al unui instrument MIDI, la care se interprează. Fragmentul preînregistrat poate fi o micro-structură de câteva sunete, o frază muzicală, o succesiune de acorduri ori poate avea orice structură, extensie și complexitate.

Este semnificativă una dintre operațiile sub-algoritmului de analiză (Figura 3, 1.), particularizată prin

- the rhythmic augmentation/ diminution of the prerecorded musical sequence
- the melodic synchronization, event by event, of the prerecorded musical sequence (tempo, duration, and intensity synchronization)

A. Playback Control

1. Any musical event, in the low range,
2. Triggers the playback of the prerecorded musical sequence

The *Playback Control* compositional algorithm (Figure 3) has a high degree of generality; it endeavors to solve a wide range of synchronization issues between performer and computer. This refers to the control of the playback of any prerecorded MIDI sequence, by the use of a musical event – single note, melodic or harmonic interval, melodic structure, chord, cluster etc. – performed in the low range of the MIDI instrument. The prerecorded sequence could be a musical micro-structure, a musical phrase, a series of chords, or may have any structure, extension and complexity.

There is one significant analysis sub-algorithm operation (Figure 3, 1.), since it produces the expected response at the time of the first sound of a block of sounds emerges, avoiding in this way, unexpected interruption during the playback of the prerecorded sequence (see below for a detailed description of the sub-algorithm).

By inserting additional commands and objects, the compositional algorithm may be developed so as to automate the playback of a complete series of prerecorded sequences.

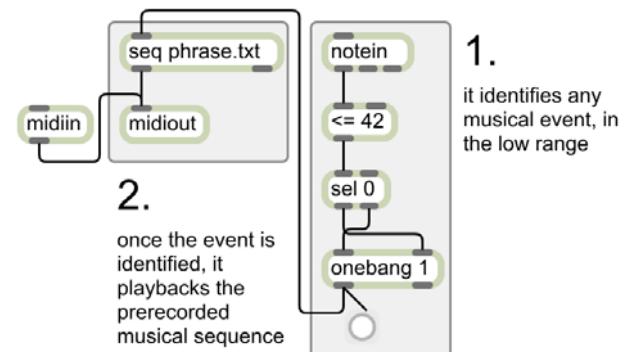


Figure 3. **Playback Control**

The analysis sub-algorithm operations are executed step by step, from top to bottom and from right to left, in a critical order of reliable functioning:

- 1.1. The *notein* object identifies and selects only the *Note On* and *Note Off* messages which constitute the temporal flow of input data. The object converts and transmits *int* messages that are integers corresponding to the pitch of the sound.
- 1.2. The values of the *int* messages received by the *<=* object are compared as they arrive with the number of his argument, 42. This value represents the upper limit of the low sound range, expanded from 0 (C-2) to 42 (F # 1); C3 in MIDI system is central C. If it meets the condition, i.e. numbers are less than or equal to 42, the object generates one *int* message with value of 1, for each low sound, but if the statement is false, the generated *int* message has the value of 0.
2. The *sel 0* object selects the first message received, which is the *int* message from the *<= 42* object. It then connects to a *onebang 1* object, which triggers a *bang* message to a small circle object.

faptul că aceasta produce răspunsul scontat la apariția primului sunet dintr-un bloc ori succesiune de sunete grave, evitându-se, în acest fel, întreuperea intempestivă, respectiv redifuzarea fragmentului preînregistrat (vezi mai jos descrierea detaliată a operațiilor sub-algoritmului).

Prin inserarea unor comenzi și obiecte informative suplimentare, algoritmul compozitional poate fi dezvoltat în aşa fel încât să fie automatizată difuzarea sincronă a unei întregi serii de fragmente preînregistrate.

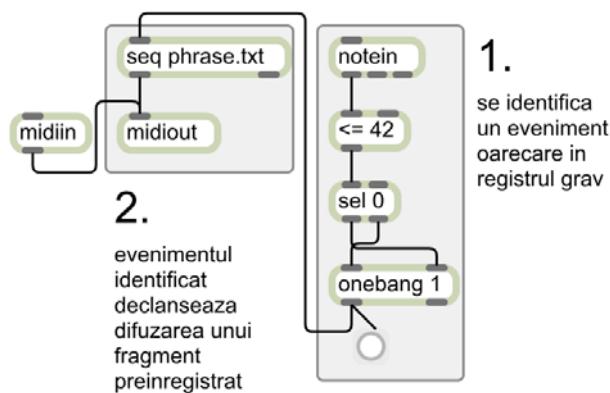


Figura 3. Control difuzării

Operațiile sub-algoritmului de analiză (Figura 3, 1.) sunt executate, pas cu pas, de sus în jos și de la dreapta spre stânga, într-o ordine critică pentru buna funcționare a acestuia:

- 1.1. Obiectul `notein` identifică și selectează, din fluxul temporal de date de intrare, doar mesajele `Note On` și `Note Off`. Obiectul convertește și transmite mesaje `int`, sub forma numerelor întregi, corespunzătoare înălțimii sunetelor.
- 1.2. Valorile mesajelor `int`, recepționate de obiectul `<=`, sunt comparate, pe măsură ce sosesc, cu cifra argumentului său, 42. Această valoare reprezintă limita superioară a registrului grav, întins de la 0 (C-2), până la 42 (F#1); C3 în sistem MIDI este sunetul do central. Dacă se întârziește condiția, și anume, numerele sunt mai mici sau egale cu 42, obiectul generează câte un mesaj `int` cu valoare 1, pentru fiecare sunet grav, însă, dacă declarația este falsă, mesajul `int` generat are valoarea 0.
- 1.3. Obiectul `sel` compară valorile 1 și 0 cu argumentul său, 0. Dacă mesajul `int` testat este 0, atunci un mesaj `bang` este comunicat prin priza de ieșire din stânga, altfel, dacă cifra mesajului `int` este 1, atunci acest mesaj `int`, respins după testare, este comunicat prin priza din dreapta, semnalând sunete în registrul grav.
- 1.4. Cele două cabluri conective sunt cuplate încruziat cu obiectul `onebang`, cu argumentul 1, astfel încât acesta permite să treacă un singur mesaj `int`, care, convertit ulterior, declanșează lectura fragmentului muzical preînregistrat. Mai precis, mesajul `bang`, recepționat prin priza din dreapta, desemnând că sunetele nu sunt grave, împreună cu argumentul 1 al obiectului `onebang`, inițializează obiectul, ca și când acesta a primit deja un mesaj `bang` prin priza din dreapta, lăsând să treacă un singur mesaj `int`, celealte fiind blocate. La rândul lui, acesta este convertit într-un mesaj `bang` și trimis obiectului `seq` (Figura 3, 2.), care declanșează lectura fisierului „phrase.txt”, unde este stocat fragmentul preînregistrat. În acest fel este posibilă semnalarea

- 1.3. The `sel` object compares the values of 1 and 0 with its argument, 0. If the tested `int` message is 0, then a `bang` message is sent through the left outlet, otherwise, if the number of the `int` message is 1, then this `int` message, rejected after testing, is sent through the right outlet, pointing to low pitches.
- 1.4. The two cables are cross coupled with the `onebang` object, with the argument 1, so that it allows passing a single `int` message, which subsequently converted, will trigger the prerecorded music sequence. More precisely, the `bang` message, received by the right outlet, designating that the pitches are not low, along with argument 1 of the `onebang` object, initializes the object as if it has already received a `bang` message through the right outlet, allowing a single pass of `int` message, others are stopped. Consequently, it is converted to a `bang` message, and is sent to the `seq` object (Figure 3, 2.), which triggers the playback of the “phrase.txt” file. Thus it is possible to isolate to the first sound of a musical event, or of a package of sounds; middle and high pitches were rejected by the action of object `<=`, described above in 1.2.

B. Mapping

1. A melodic ascending interval, between C 3 and C 4,
2. Triggers the rhythmic augmentation/ diminution of the prerecorded musical sequence

The *Mapping* compositional algorithm (Figure 4) is more selective and is addressed to a particular synchronization situations and performer/ computer interaction. The algorithm brings the solution by identifying, from the multitude of musical events produced by the instrument, an ascending melodic interval located in the first octave, between C 3 and C 4 in MIDI system, on one hand, and it calculates the interval size, on the other hand. The aim is to control the playback of the prerecorded musical sequence, and also to modify it by means of the rhythmic augmentation and diminution, proportionally to the identified size interval. The sequence can be anything in MIDI format. In addition, the algorithm selects a harmonic interval, respectively the upper interval of the chord structure of three or more sounds.

For developing purposes of the algorithm, it may be redefined the selection type of the captured musical event in a restrictive way, to a melodic interval, thus being ignored the chords and the harmonic intervals. An easier solution is to automate the on/ off function of the sub-algorithm, using `toggle` and `gate` objects, for example. Another approach, more advanced, is to filter the events perceived as instantaneous, using the information produced by the `thresh` object.

primului sunet, dintr-un eveniment muzical, dintr-un bloc ori succesiune de sunete grave; sunetele medii și acute au fost respinse prin acțiunea obiectului `<=`, descrisă mai sus la punctul 1.2.

B. Mapping

1. Un interval melodic, ascendent, situat între do 3 și do 4,
2. declanșează augmentarea/diminuția ritmică a unei structuri ritmico-melodice preînregistrate

Acest algoritm compozitional (Figura 4) este mai selectiv și se adresează unor situații particulare de sincronizare și interacțiune interpret/calculator. Algoritmul aduce o soluție, pe de-o parte, pentru identificarea, din multitudinea de evenimente muzicale produse prin intermediul instrumentului la care se interpretează, a unui interval melodic ascendent, situat în octava întâi, adică între do 3 și do 4 în sistem MIDI, iar pe de altă parte, calculează valoarea intervalului. Scopul este de a controla difuzarea unui fragment muzical preînregistrat și, totodată, de a-l modifica, prin augmentare și diminuție ritmică, în raport cu mărimea intervalului identificat. Fragmentul poate fi orice, în format MIDI. În plus, algoritmul mai selectează un interval armonic, respectiv intervalul superior al unei structuri acordice de trei sau mai multe sunete.

În sensul dezvoltării algoritmului, se poate redefini selecția evenimentului muzical captat, la una și mai restrictivă, adică la un interval melodic, fiind, astfel, ignorate acoururile și intervalele armonice. Cea mai simplă soluție este automatizarea acțiunii de scoaterea și punerea în funcționare a sub-algoritmului, prin utilizarea obiectelor `gate` și `toggle`, de exemplu. O altă soluție, mai rafinată, este filtrarea evenimentelor percepute ca instantanee, folosind informația produsă de obiectul `thresh`.

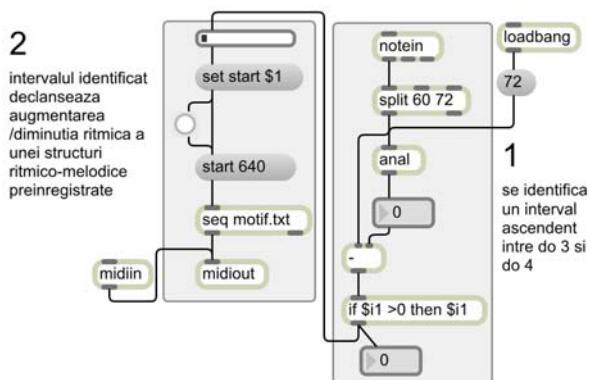


Figura 4. Mapping

Operațiile sub-algoritmului de analiză și calcul (Figura 4, 1.):

- 1.1. Obiectul `notein` permite tranzitarea, de la instrumentul MIDI conectat prin interfață la obiectul `split`, numai a mesajelor `Note On` și `Note Off`. Acestea sunt convertite în mesaje `int`, numerele reprezentând înălțimea sunetelor.
- 1.2. Obiectul `split` căută și transmite numerele situate în limitele specificate prin argumentele sale, cu valori de 60 (C3) și 72 (C4). Drept urmare, acesta ignora mesajele `int` care au valori mai mici și egale cu 59, respectiv mai mari și egale cu 73, aşadar, pe cele care nu reprezintă sunetele din octava întâi.

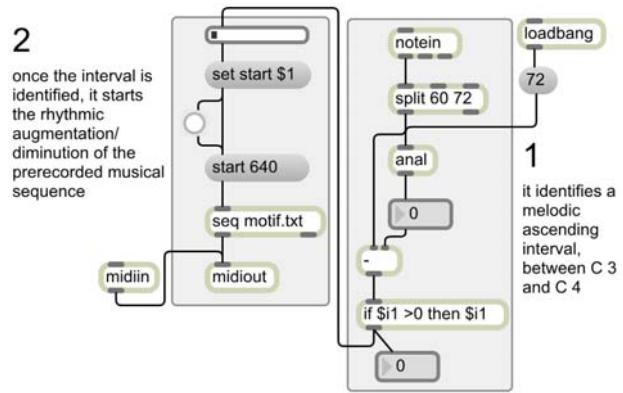


Figure 4. Mapping

The calculus operations of the analysis algorithm (Figure 4, 1.):

- 1.1. The `notein` object allows transiting only the `Note On` and `Note Off` messages which are sent by the MIDI instrument through the interface. These messages are converted in `int`, and the numbers represent the pitch.
- 1.2. The `split` object is looking and forwarding the numbers located within the limits specified by its arguments, with values of 60 (C3) and 72 (C4). As a result, the object ignores the `int` messages with values smaller and equal to 59, likewise greater and equal to 73, therefore the pitch who is not in part of the first octave.
- 1.3. In order to analyze two consecutive numbers, `anal` object saves and transfer integers to the left, from state 2 to state 1, in a `list` message, which is generated and sent to the outlet. This transfer is important for the arithmetic calculation performed in Section 1.5.
- 1.4. The `number` object displays and transmits only the first numerical value, by filtering and converting the `list` message in `int` message.
- 1.5. The `-` object (subtraction operator) performs the arithmetic computation of two `int` messages. By the right inlet, it is inserted the first operand (prior), received from the `number` object, without causing the computation. The operand represents the `Note On` message. The second operand (rear), which is a `Note Off` message, actually runs the subtraction computation and sends the result, once the message is received from the `split` object, by the left outlet.

Exemplu

The message and data flow when `Note On` (60/C3) is transmitted:

```

>> int (60) from notein to split...
>> int (60) from split to anal...
>> list (3 arguments) from anal to number...
>> int (72) from number to -...
    // comment: C4 is transmitted to the
    prior operand (right outlet) of
    "subtraction" object //
>> int (60) from split to -...
    // comment: C3 is transmitted to the
    rear operand (left outlet) of
    "subtraction" object //
>> int (-12) from - to if...
  
```

- 1.3. Pentru analiza a două numere consecutive, obiectul *anal* memorează și transferă numerele întregi spre stânga, din starea 2, în starea 1, în cadrul unui mesaj *listă*, pe care acesta îl generează prin priza sa de ieșire. Transferul este determinant pentru calculul aritmetic efectuat la punctul 1.5.
- 1.4. Obiectul *number* afișează și transmite doar prima valoare numerică, filtrând și convertind mesajul *listă* într-un mesaj *int*.
- 1.5. Obiectul - (operator aritmetic de scădere) execută un calcul între două mesaje *int*. Prin priza sa din dreapta, se introduce primul operand (anterior), recepționat de la obiectul *number*, fără să determine efectuarea calculului. Operandul reprezintă un mesaj *Note On*. Al doilea operand (posterior), care reprezintă un mesaj *Note Off*, execută efectiv operația de scădere și transmite valoarea calculată, odată ce mesajul este recepționat de la obiectul *split*, prin priza din stânga.

Exemplu

Fluxul mesajelor după transmiterea *Note On* (60/C3) de la instrument:

```
>> int (60) de la notein la split...
>> int (60) de la split la anal...
>> list (3 argumente) de la anal la number...
>> int (72) de la number la -...
// comentariu: C4 de inițializare este transmis
    operandul anterior (priza din dreapta) a obiectului „minus” //
>> int (60) de la split la -...
// comentariu: C3 este transmis operandului posterior
    (priza din stânga) a obiectului „minus” //
>> int (-12) de la - la if...
```

Fluxul mesajelor după transmiterea *Note Off* (60/C3 cu intensitate 0):

```
>> int (60) de la notein la split...
>> int (60) de la split la anal...
>> list (3 argumente) de la anal la number...
>> int (60) de la number la -...
// comentariu: C3 spre operandul anterior //
>> int (60) de la split la -...
// comentariu: C3 spre operandul posterior //
>> int (0) de la - la if...
```

- 1.6. Obiectul *if* evaluează expresia formulată în cadrul acestuia. Expresia conține o declarație condițională: dacă valoarea mesajului *int* este mai mare decât 0, atunci transmite această valoare. Dacă este adevărată condiția, sunt transmise numerele pozitive, reprezentând intervalele ascendente, altfel, valorile sunt ignorate. Mărimea intervalelor validate, de la 1 la 12, este folosită pentru determinarea proporției între augmentare și diminuție, prin intermediul sub-algoritmului de prelucrare (Figura 4, 2.). Cu cât valoare intervalului este mai mare, cu atât durata este mai mică, iar ritmul devine mai rapid; reciproc, cu cât valoare intervalului este mai mică, cu atât durata este mai mare, iar ritmul devine mai lent.

C. Acompaniament automatizat

1. Un acord minor, în nuanță mp
2. declanșează sincronizarea, eveniment cu eveniment, a acordului minor cu un fragment melodic preînregistrat (sincronizare a tempo-ului, duratei și intensității sunetului)

The message and data flow when *Note Off* (60/C3 with velocity of 0) is transmitted:

```
>> int (60) from notein to split...
>> int (60) from split to anal...
>> list (3 arguments) from anal to number...
>> int (60) from number to -...
    // comment: C3 to prior operand //
>> int (60) from split to -...
    // comment: C3 to rear operand //
>> int (0) from - to if...
```

- 1.6. The *if* object evaluates the expression it holds. The expression contains a conditional statement: if the *int* message is greater than 0, then it sends this value. If the condition is true, the positive numbers are validated, representing ascending intervals, otherwise, values are ignored. Validated size intervals from 1 to 12, are used to determine the augmentation and diminution proportion of the processing sub-algorithm (Figure 4, 2.). As the interval value is higher, the duration is less, and the rhythm gets faster; reciprocally, as the interval value is lower, the duration is greater, and the rhythm gets slower.

C. Automated accompaniment

1. A minor chord, in mp,
2. Triggers the melodic synchronization, event by event, of the prerecorded musical sequence (tempo, duration, and intensity synchronization)

This example of automated accompaniment (Figure 5) could join the compositional algorithms family, with a long history in fact, which fulfill one of the composer's dreams: the computer, equipped with "intelligent" software, "understands" the performer actions, and "follows" the performers' score, note by note, in sync with the computer's score.

The *Automated accompaniment* compositional algorithm aims to synchronize, in terms of tempo, duration and intensity, a random sequence of minor chords in mp performed on the MIDI instrument, with a prerecorded melody played back by the computer. The main purpose of this algorithm remains the analysis and configuration of the message and data temporal flow received from the performer.

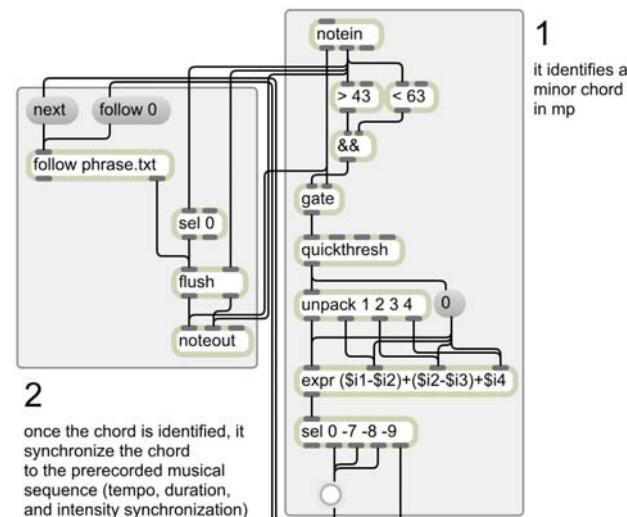


Figure 5. Automated accompaniment

Acest exemplu de acompaniament automatizat (Figura 5) s-ar putea alătura algoritmilor compozitionali, cu istorie îndelungată de altfel, care împlinesc un vis al compozitorului: calculatorul, dotat cu un program „inteligent”, „înțelege” acțiunile interpretului solist și „urmărește” partitura acestuia, notă cu notă, sincronizându-i acompaniamentul.

Algoritmul exemplificat își propune să sincronizeze, în termeni de tempo, durată și intensitate, o succesiune aleatorie de acorduri minore, în nuanță mp, interpretate la instrumentul MIDI, cu o monodie preînregistrată și redată de calculator. Scopul principal al construcției acestui algoritm compozitional rămâne analiza și structurarea fluxului temporal imprevizibil de mesaje și date receptionat de la interpret.

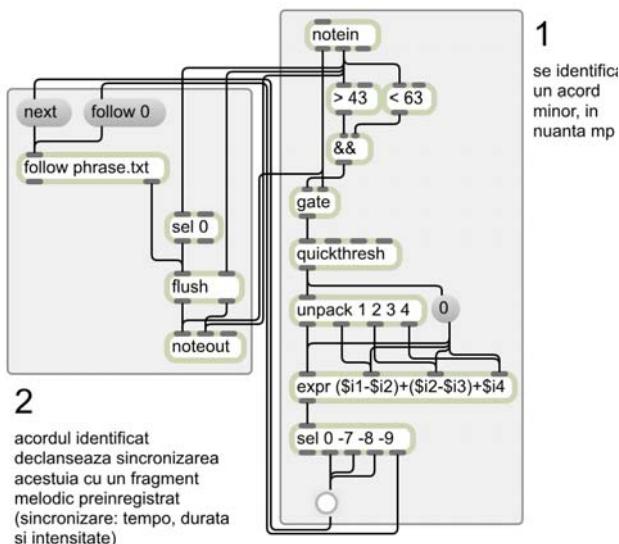


Figura 5. Acompaniament automatizat

Operațiile sub-algoritmului de analiză (Figura 5, 1.):

- 1.1. Obiectul *notein* filtrea mesajele *Note On* și *Note Off*, le convertește în mesaje numerice *int* și le transmite corespunzător instrucțiunilor, prin priza din mijloc, apoi prin cea din stânga, desemnând valori de intensitate, respectiv de înălțime.
 - 1.2.–1.3. Obiectele *<* și *>* (operatori relaționali) compară valorile cu argumentele lor specificate în obiect, și anume, 63, respectiv 43. Dacă declarațiile sunt adevărate, cele două obiecte transmit, de la dreapta spre stânga, câte un mesaj *int*, cu valoare 1. Dacă declarațiile sunt false, mesajele au valoarea 0.
 - 1.4. Numerele sunt comparate de obiectul *&&* (operator logic). Dacă ambele valori de intrare sunt egale cu 1, atunci obiectul comunică mesajul *int* 1, ceea ce înseamnă confirmarea condiției. Cifra 1 mai indică faptul că intensitatea are valori cuprinse între 43 și 63 – tronsonul relativ de valori raportate la nuanță mp. În caz contrar, dacă una dintre numere este 0, atunci obiectul *&&* comunică neîndeplinirea condiției, cifra 0.
 - 1.5. Obiectul *gate* acționează ca un „controlor de trafic”. La recepționarea valorii 1 în priza din stânga, de la obiectul *&&*, obiectul *gate* permite transmisia mesajelor *int* intrate prin priza din dreapta, primite de la obiectul *notein*, adică transferă numai sunetele care au nuanță mp. Obiectul *gate* blochează celelalte valori de înălțime, dacă recepționează valoarea 0 în priza sa din stânga, de la obiectul *&&*.
 - 1.6. Obiectul *quickthresh* este optimizat pentru detectarea acordurilor. În teoria muzicală, una dintre însușirile
- The calculus operations of the analysis algorithm (Figure 5, 1.):
- 1.1. The *notein* object filters the *Note On* and *Note Off* messages, converts them into *int* messages, and sends them to the middle outlet, then to the left outlet; numbers represent intensity and pitch values.
 - 1.2.–1.3. The *<* and *>* objects (relational operators) compare the values of their arguments, 63 and 43. If the declarations are true, both objects transmit from right to left one *int* message with value of 1. If the declarations are false, the message sent has the 0 value.
 - 1.4. The numbers are compared by the *&&* object (logical operator). If both input values are equal to 1, then the message *int* with value of 1 is sent out, which means that the condition is confirmed. The number 1 indicates also that the intensity has values between 43 and 63 – the relative values of mp. Otherwise, if one of the received numbers is 0, the condition is false, then the *&&* object sends out 0.
 - 1.5. The *gate* object acts as a “traffic controller”. On receiving the 1 value in the left outlet from *&&* object, *gate* object allows the transmission of *int* messages entering the right outlet, received from the *notein* object, that is transferring only the sounds in mp. The *gate* object stops the other pitch values, if 0 is received in the left outlet from *&&* object.
 - 1.6. The *quickthresh* object is optimized to detect chords. In music theory, one of the chord characteristics is simultaneity. But in the musical practice, there are often delays countable in milliseconds between the occurrences of sounds that make up the chord. In Max, these delays between two consecutive *Note On* messages are expressed in delta time values. The *quickthresh* object collects and sends the pitch values validated by *gate* object, in the form of *list* message, if they occur within a period of up to 40 milliseconds, calculated as time delta.
 - 1.7. The *message box* object transmits 0 each time a pitch is identified.
 - 1.8. The values of the *list* message are unpacked and sent after conversion, one by one, as *int* message, through the four outlets of the *unpack* object.
 - 1.9. The *expr* object evaluates an expression alike C language. It calculates algebraic sum of variables or random intervals that compound the chord, which may be of three or more sounds: $(x-y)+(y-z)+0$.
 - 1.10. The *sel* objects selects the result, i.e. *int* messages with values of -7, -8 and -9, which are the minor triad-chords, in root position, second inversion, respectively in first inversion, and then transmits, for each value, a *bang* message. Every message helps to playback the prerecorded sequence stored in the “phrase.txt” file, by the specialized *follow* object (Figure 5, 2.). Other results different from 0 are submitted by the right outlet.

acordului este simultaneitatea. În practica muzicală, există de cele mai multe ori diferențe de câteva milisecunde între apariția sunetelor care compun un acord. În Max, aceste diferențe de timp dintre două mesaje *Note On* consecutive, sunt exprimate în valorile timpului delta. Obiectul *quickthresh* colectează și transmite valorile de înălțime validate de obiectul *gate*, într-un mesaj *listă*, dacă acestea apar în limitele unui interval de timp de până la 40 de milisecunde, calculat pe baza timpului delta.

- 1.7. Obiectul *message box* transmite cifra 0 la fiecare identificare a înălțimii.
- 1.8. Valorile mesajului *listă* sunt izolate și transmise după convertire, una către una, în forma mesajelor *int*, prin cele patru prize de ieșire ale obiectului *unpack*.
- 1.9. Obiectul *expr* evaluează o expresie formulată asemănător limbajului C. Acesta calculează suma algebrică a variabilelor sau intervalelor care alcătuiesc acordurile aleatorii, care pot fi în fond de trei sau de mai multe sunete: $(x-y)+(y-z)+0$.
- 1.10. Obiectul *sel* selectează rezultatul, adică mesajele *int* cu valori -7, -8 și -9, care reprezintă acorduri minore de trei sunete, în starea directă, răsturnarea a II-a, respectiv răsturnarea I a acordului captat și transmite, pentru fiecare valoare, un mesaj *bang*. Fiecare mesaj contribuie la redarea secvențială a datelor preînregistrate în fisierul „phrase.txt”, cu ajutorul obiectului specializat *follow* (Figura 5, 2.). Celelalte rezultate diferite de 0 sunt transmise prin priza din dreapta.

Cum se realizează sincronizarea dintre acordurile interpretului cu melodia calculatorului?

De reținut că există o relație de subordonare cu privire la interactivitate – calculatorul supus urmează interpretul dominator –, în poftidă faptului că, la prima vedere, am fi tentați să atribuim melodia suverană doar interpretului.

Revenind la mecanismul de sincronizare, după cum am arătat la punctul 1.10., un acord minor de trei sunete, în oricare stare ar fi, în plus, în orice registru s-ar situa, dar interpretat în distribuție strânsă și în nuanță, determină transmiterea unui mesaj *bang*, după parcurgerea operațiilor sub-algoritmului.

După ce obiectul *follow* a trecut automat în mod „urmărire”, prin comanda *follow 0*, odată cu respingerea unui eveniment diferit față de acordul minor scontat, mesajul *bang* provenit de la obiectul *button*, cuplat cu comanda *next* trimisă obiectului *follow*, face ca lectura mesajelor *Note On* din fisierul „phrase.txt” să avanseze un pas. La fiecare nou mesaj *bang*, aşadar, la apariția de noi acorduri confirmate în sub-algoritmul de analiză, lectura va avansa cu câte un pas la mesajul *Note On* următor. Sub aspect muzical, acest flux de mesaje și date este descris astfel: interpretul construiește și interpretează formule ritmice și acorduri, care impun fragmentului preînregistrat un tempo și ritm distințe. În programarea cu Max, avansarea de la un mesaj *Note On* la următorul *Note On* este posibilă datorită instrucțiunii obiectului *follow* de a omite toate mesajele MIDI, cu excepția *Note On*. Durata acordurilor interpretate la instrument este transferată fragmentului redat de calculator, cu ajutorul obiectelor *sel* și *flash*, grație mesajelor *Note Off* transmise ca mesaje *int* de obiectul *notein*. Valorile de intensitate provenite de la obiectul *notein* sunt combinate în obiectul *flash* cu valorile mesajelor *Note*

How the performer's chords and computer's melody are synchronized?

Note that there is a subordination relationship of the interactivity –computer follows the artist – despite the fact that, at first sight, we would be tempted to attribute the melody only to performer.

Back to the synchronization mechanism, as mentioned in the paragraph 1.10., the minor triad-chord, in any state would be, in addition, any register would be, but performed in close distribution, is triggering to send a *bang* message, at the end of completing the operations of the sub-algorithm.

After the *follow* object is automatically switched in “follow” mode, using the *follow 0* message, in conjunction with the rejection of an event which is different from the expected minor chord, the *bang* message of the *button* object, coupled with the *next* command sent it to the *follow* object, makes the playback of *Note on* messages of the “phrase.txt” file to advance one step. At each a new *bang* message is sent, therefore, at each a new chord is confirmed by the analysis sub-algorithm, the playback will advance by one step to the next *Note On* message. From musical point of view, the message passing and data flow are described as follows: the artist creates and performs rhythmic formulas and chords, which drive a distinct tempo and rhythm to the prerecorded sequence. In Max programming, advancing from *Note On* message to the next *Note On* is possible due to the *follow* object instruction to omit all MIDI messages except *Note On*. The chord duration performed on the instrument is transferred to the sequence played by computer, using *flash* and *sel* objects, thanks to the *Note Off* messages sent as *int* messages by the *notein* object. The intensity values of the *notein* object are combined with the *Note On* values of the stored sequence, within the *flash* object, and then are transmitted to the *noteout* object. All of these operations lead to synchronize, interactively and in real-time, in terms of tempo, duration and intensity, the performer's chords, created or improvised by the musician, with the computer's melody.

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.

In this regard, the interdisciplinary-specific Music and Technology professions are considered worldwide to have a bright future. Over 110 universities and music research centers from almost 30 countries have included in their academic programs the interactive systems and the Max programming courses, with the aim of diversifying their educational offer, focusing on the recent musicians' needs.

In Romania, the introduction of such an innovative course would mean:

- ✓ attracting more students into existing specializations,
- ✓ increasing the competitiveness of graduates on the labor market,
- ✓ global educational integration (upgrading),

On ale fragmentului stocat și transmise apoi obiectului *noteout*. Toate aceste operații laolaltă conduc la sincronizarea, interactivă și în timp-real, în ceea ce priveste tempo-ul, durata și intensitatea, a acordurilor interpretate, create ori improvizate de către muzician cu melodia calculatorului.

CONCLUZIE

Fără îndoială, compozitorul secolului XXI va fi un muzician educat în programarea calculatoarelor și, totodată, în compunerea de muzică. Tendința crescândă de programare de sisteme muzicale interactive întărește nevoia de interacțiune dintre interpret și calculator. Interacțiunea va restructura dramatic modul în care compozitorii compun, în zilele noastre, muzică pe calculator.

În acest sens, profesiunile cu specific interdisciplinar din domeniile Muzică și Tehnologie sunt considerate a fi, pe plan mondial, de actualitate și de viitor. Peste 110 universități și centre de cercetare cu profil muzical din aproape 30 de țări au inclus, în programele academice, discipline care au ca obiect de studiu sisteme muzicale interactive și programarea muzicală cu limbajul Max, cu scopul diversificării ofertei educaționale, centrate pe nevoile actuale ale muzicienilor.

În țară, introducerea unui astfel de curs universitar și postuniversitar ar însemna:

- ✓ atragerea unui număr mai mare de studenți spre specializările existente,
- ✓ sporirea competitivității absolvenților pe piața forței de muncă,
- ✓ o integrare (modernizare) educațională valoroasă, la nivel mondial,
- ✓ dar mai ales o oportunitate pentru studenții români de a se familiariza cu metode și practici de programare în Max.

Iată mai jos câteva informații care se referă la locuri, instituții/departamente și denumiri ale cursurilor oferite în august 2009, având ca obiect Max/MSP și sisteme muzicale interactive: Paris, Institut de recherche et coordination acoustique/musique – *Interaction temps réel*; Londra, Middlesex University, Lansdown Centre for Electronic Arts – *Interactive and Algorithmic Systems*; Zürich, School of Music, Drama and Dance – *Live Electronics and Interactive Composition with Max/MSP*; Irvine, University of California, Department of Music – *Interactive Arts Programming*; Montreal, McGill University, Music Technology Department – *Interactive Music Systems* (MUMT 610); Zagreb, Center for Algorithmic Music – *Composition and Multimedia in MAX environment*; Seul, Hanyang University, School of Music – *MIDI and Real Time Programming with Max and MSP*.

- ✓ and especially an opportunity for Romanian students to become familiar with Max programming methods and practices.

Below I provide the reader with information regarding places, institutions/ departments and the names of the courses offered in August 2009, concerning Max/MSP and interactive music systems: Paris, Institut de recherche et coordination acoustique/musique – *Interaction temps réel*; Londra, Middlesex University, Lansdown Centre for Electronic Arts – *Interactive and Algorithmic Systems*; Zürich, School of Music, Drama and Dance – *Live Electronics and Interactive Composition with Max/MSP*; Irvine, University of California, Department of Music – *Interactive Arts Programming*; Montreal, McGill University, Music Technology Department – *Interactive Music Systems* (MUMT 610); Zagreb, Center for Algorithmic Music – *Composition and Multimedia in MAX environment*; Seul, Hanyang University, School of Music – *MIDI and Real Time Programming with Max and MSP*.

BIBLIOGRAFIE / REFERENCES

- Borza, Adrian (2008). *Muzică și calculator*. București: Editura Muzicală
- Rowe, Robert (1993). *Interactive Music Systems: Machine Listening and Composing*. Massachusetts, Cambridge: The MIT Press
- Rowe, Robert (1999). The Aesthetics of Interactive Music Systems. *Contemporary Music Review*, Vol. 18, Part 3, 83, 87
- Zicarelli, David (2006). *Max User's Manual – Reference Manual*, Version 4.6
- Winkler, Todd (1998). *Composing Interactive Music: Techniques and Ideas Using Max*. Massachusetts, Cambridge: The MIT Press