DKomp ose : a package for interactive composition in the Max environment, adapted to the acoustic MIDI Disklavier piano.

Simon BOLZINGER

CNRS - Laboratoire de Mécanique et d'Acoustique
31 chemin J. Aiguier, 13402 Marseille Cedex 9, FRANCE

ABSTRACT

DKomp ose is a package of Max subpatches to ease interactive composition. Starting with the original examples from J.C. Risset and S. Van Duyne, and adding a number of new possibilities, a formalisation of the exchanges of MIDI codes between the pianist and the computer make the programming of new interactive pieces easier for the composer. The solution of basic technical problems, like MIDI feedback and velocity-dependent delays, is included. Musical applications include delay, harmonizer, arpeggiator, trills, echos, sequence recording or performing, note detector, level detector, performance analysis (durations of notes, chords or silences, contrast, staccato/legato proportion, timing deviations measurements), tempo detector and follower... A specific way to edit scores, on Max graphical tables, gives some possibilities of score following. The context proposed can be extended to general interactive composition with any MIDI compatible device.

INTRODUCTION

The recent developments in real-time processing open to the contemporary composer a new way of composing, by the multiplication of the tasks that a single performer can do, when he is interactively connected to a computer. The real-time control of an acoustic instrument by a computer allows interactive composition, which has a computed part, although the sound is entirely produced by a mechanical way, and with human expressivity. The first implementation of this process were made on the piano by J.C. Risset and S. Van Duyne at M.I.T. in 1989 [1]. With the same equipments, i.e. the MIDI piano Yamaha Disklavier and the MAX software (M.Puckette 1989), and the original ideas and patches from Risset & Van Duyne, I created several interactive modules, and organised them in a friendly-user context, as a package of functions named D K O m p o s e . It looks like a library of ready-to-use tools, which lets the composer imagine any interactivity between the pianist and the computer, without thinking about the details of MIDI programming in Max.

First of all, the basic developments, and the examples from "Duel for one pianist" of J.C. Risset, are presented. Then the general context for the programming of real-time interactive patches is defined. All the DKompose functions are classified and briefly presented. Further extensions that could add to DKompose are proposed.

162
THE BASIC DEVELOPMENTS

Since this work was made with an upright Disklavier, a specific study of the limits of interactivity with this piano was done. It includes the resolution of basic technical problems like MIDI feedback and velocity-dependent delays, and the implementation of the first musical examples of interactivity.

It would be wrong to assert that with DKompose any score can be performed thanks to the interactivity: the best programming cannot pass over mechanical limitations of instrumentation, and in the case of Disklavier these limitations are musically important.

Since the fidelity in reproducing human performance is very good, while the intensity is determined by the velocity of the hammer, it is inescapable that strict simultaneity (like real-time harmonization) is impossible. The time interval between MIDI note-on messages and sound can be reduced below 50 milliseconds for loud sounds, which is often musically acceptable; but it goes up to 200 milliseconds for soft sounds, which is heard, in the case of harmonization, as an undesirable echo. Although this seems to be an important limitation, practically it is often possible to find a satisfactory way to reproduce musical effects that could have needed exact simultaneity between player and computer.

This delay between code and sound is also depending upon velocity, and it produces a bizarre “wrapping” when reproducing a MIDI sequence. This is solved by the function skew (inspired by S. Van Duyne’s “anti-skewing module”) [1], but within the following limits: the correct rhythmic of any sequence is reproduced, but the notes that are very short or very soft can not always be performed by Disklavier: such notes just do not sound, as the note-off message comes before that the hammer hits the string. A sequence prepared with synthetic or sampled sound will not sound the same on Disklavier, particularly if it uses very-short or very-soft notes, or fast repetitions of a single note. The use of clusters is also limited by the 16 notes polyphony (maximum 16 notes held down by the computer); lastly the dynamics is compressed, as the MIDI velocity actually played by the piano is limited between 30 and 100 for the upright model, so that dynamics is reduced from 128 to 70 values.

Taking in account these general limitations, great possibilities of interactivity remain possible: when the basic technical problems are solved, the original examples given by Risset and Van Duyne become very easy to implement.

The basic developments are the resolution of MIDI feedback and rhythmic skewing. Risset & Van Duyne made the first patches to solve those problems [1], and I made new versions for Max 2.2: both phenomena are now solved by using independent boxes (AFBin, AFBeat, skew), or integrated initialization (DKinit). The MIDI feedback keeps on causing problems when the programming and the score performed are not consistent (i.e. if the pianist plays notes that the computer is supposed to play). This can be solved with a allnotesoff message controlled by the pianist. I am now experimenting an active feedback filter, which will solve this little problem more elegantly.

From the first musical examples, illustrated by J.C.Risset in his "Duet for one pianist", I also made ready-to-use boxes. These examples use sequence control by keystrokes, sequence’s speed control, variable harmonization (like “mirror” and “stretch”), arpeggios control, tempo calculation [1]. Such processes are available in the following DKompose’s functions: DKseq, harmo, mirror, stretch, arpg, tempo. To specify and use them is elementary, thanks to the Max environment and the DKompose’s context.
The basic developments are done for a specific use on Disklavier: interactive composition on any other MIDI instrument would not need functions like skew, ABF or DKinIt; the delays are even dependent on which model of the piano is used. The initialisation also includes the formatting of input and output data according to the use of the piano’s sound: single MIDI channel, 88 notes, and two (or three) pedals (DKin, DKout). However, the formalisation and the general context, explained in the following, is supposed to be useful for any kind of interactive composition using the MIDI protocol.

THE DKcompose CONTEXT

The Max graphical object-oriented programming environment is specially organised for real-time processing: in short all data and events are detected and controled with bangs, toggles, numeric boxes, and functional boxes, using the right-to-left order [2-5]. Things are just the same for DKcompose, with the implicit formalisation rules that follow: any DKcompose function is set on or off by a toggle (mostly the lefthemost inlet); it receives incoming data that can be either numerical (mostly int, or bang, or toggle). The logical way to program an interactive score is the following: any specific action by the pianist will be detected as a bang, thanks to detection and measurement functions; any specific action by the computer can be set on or off by a toggle, and triggered by a bang. Functional utilities like bang/toggle or toggle/bang conversions are pre-programmed; it certainly looks like obvious for the Max programmers, but the composer can take a good advantage in using such a formalisation, that let him program his Max-patch almost like he would write the score of his interactive composition.

As an example, the programming of an harmonization by an superior octava, will be programmed by connecting the DKcompose function harmo as follows: argument 12 for the harmonization interval, pitch and velocity from DKin outlets to harmo inlets, pitch and velocity from harmo outlets to DKout inlets. As the example is very simple, the programming would not have been much different, directly using Max’s notein and noteout: however the complexity grows rapidly when using conditions or gates, which are always useful in interactivity. For example, if you want to harmonise only the loud notes, you should only connect harmo inlets to splitvel outlets, another DKcompose function that selects a range of velocity. As the note-off message is coded with velocity zero, the direct programming of this selection would use a good part of your computer’s screen, and let you spend a little time with MIDI bugs… As the splitvel function is controlled by on/off toggle, you are free to vary the harmonisation in real time; you can also change the interval using harmo’s corresponding inlet, and so on…

THE DKcompose FUNCTIONS

The functions available in DKcompose package are classified and shortly described in the following. Interactivity programming begins by measuring parameters of the performer’s playing, and detecting specific musical events: these are the measurement functions. Then, the intervention of the computer is implemented thanks to musical functions. One also needs sequencing functions, for control and editing of sequences, and score following. The general organisation is possible thanks to the utility functions.

Measurements functions give numeric data on the performance. The basic one DKin gives pitch and velocity. The other ones were written for a real-time performance analyzer on Max [3]. These functions can measure durations (durn, durch, durt), tempo (tempo), accentuation and contrast (acc, contr), overtop sign and duration (stacc, leg).
Statistical functions give count, average, cumulated sum, and histogram (stat, histo); specific values detection (bangon, bangoff, togglenote for MIDI notes, togglev, bangv for numerical values) can convert the numerical data into bangs or toggles, and control musical effects in real time.

Musical functions are active subpatches with musical purposes, like the original examples from Risset and Van Duyne: melodic symmetry (mirror), zrepggiation (argp), intervals stretching (stretch), melodic fractal distortion (frac), or basic effects like delays, harmonization or trill (delay, harmo, trill). More specific developments are directly correlated with composition, so it is left to the user.

The sequence control is formalised using the sequencer Dkseq, which uses Max’s sequencer seq. It can be controlled by toggles for changing the speed, recording or playing (sseq). This sequencer cannot perform score following, as the speed is not changeable while playing, with an other way of score editing using Max tables (buff, show in experimentation), it will be possible to program interactive score following.

Utility functions are used for gating MIDI codes easily, and connecting toggle/hang-controlled functions together (toggles, togglebang, bangtoggle, strip, splitvel, threshvel, velp...). A real-time parsing adapted to the piano converts MIDI codes into the format pitch-velocity-duration (pianoparse, pianoformat). It also includes initialisations for Disklavier (DKinit, DKin, DKout, AFBin, AFBout, skew).

FURTHER DEVELOPMENTS

The actual DKompose 1.0 was made for interactive composition, and adapted to the MIDI acoustic piano. But it is not a limitation for the composer, in the sense that he could -and should- create new kinds of interactions, between acoustics instruments, synthesizers, samplers, and any MIDI peripheral (from standard MIDI controller to Mathews’ radio drum [4]). This only requires new functions to be integrated to DKompose (one in and one out, for each peripheral, just like DKin and DKout). All the developments made in the Max environment can be integrated, like advanced score following [7], or audionumerical I/O [6]. For any kind of interactive music using the MIDI protocol, DKompose proposes a more friendly context, in order to make the Max’s patch look as clear and useful than the classical score for the contemporary composer.

ACKNOWLEDGMENTS

I acknowledge Jean-Claude Risset for having shared his experience on interactive composition, and let me investigate with his original ideas and patches I acknowledge YAMAHA for lending the Disklavier piano model MX100A.

REFERENCES


165