Earle Brown’s “25 pianos”: a web interactive implementation

Guigue, Didier[1] & **Fábio Gomes de Andrade**[2]
[1]Departamento de Música, Universidade Federal da Paraíba, Brazil
[2]Departamento de Sistemas e Computação, Universidade Federal da Paraiba, Brazil
dguigue@openline.com.br, fgandrade@ig.com.br

Abstract: *Earle Brown’s “25 pages for 1 to 25 pianos” is a typical “open form” composition, which let a range of variables up to the performer’s decision. We explain why it represents a good case study for a web interactive implementation, on a musical as well as on a computing point of view. Then we describe how this implementation is being done in Java, discuss problems, solutions and on-going developments.*

1. Introduction

1.1 Earle Brown’s “open form” music

Among the members of the so-called “New York Group” (besides Morton Feldman, Christian Wolff and with John Cage as the senior member), Earle Brown (b. 1926) is, in the middle of the 50’s, one of the most engaged composers in “open form” musical experiments. A radical break, his seminal graphical score “December 1952” appears to be « the first serious invitation to the classical musician to improvise rather than 'read' in the conventional sense » (Ryan 1999). He was strongly influenced by the painter Jackson Pollock and the sculptor Alexander Calder. The experience of Alexander Calder's mobiles, long before Brown's meeting with Cage, had implanted in his mind the idea of mobility, and also the relationship of temporal and spatial concepts in music which had encouraged his experimentations » (*Ibid.*).

These concepts are crucial for the piece “25 pages for 1 to 25 pianos” (1953), where « he develops more consistently a notion of open form rather than the 'open content' of the graphic pieces (*Ibid.*). «A temporal order can be pre-established by the performer, obtained from the composer, or arrived at spontaneously by the performer(s) » (Delaigue 1989). Brown attempts to find a path which embraces both extreme variability while maintaining an identity for the piece. « There must be a fixed (even flexible) sound content, to establish the character of the work, in order to be called 'open' or 'available' form. » (Brown, in Nyman 1999).

1.2 The “25 Pages…” variables

“25 Pages for 1 to 25 pianos” is a composition where a number of variables are up to the performer (s), in such a way the piece, as a Calder’s mobile, will look different each time it is played, although its sonic identity always remains strongly present.

The musical notation is a blend of traditional elements — the two staves of the piano system, the horizontal reading from left to right, the proportional time representation — with less usual graphic symbols for pitches, durations and articulation. These symbols are very carefully explained in the composer's foreword, and the graduation of intensities and articulations is extremely subtle.

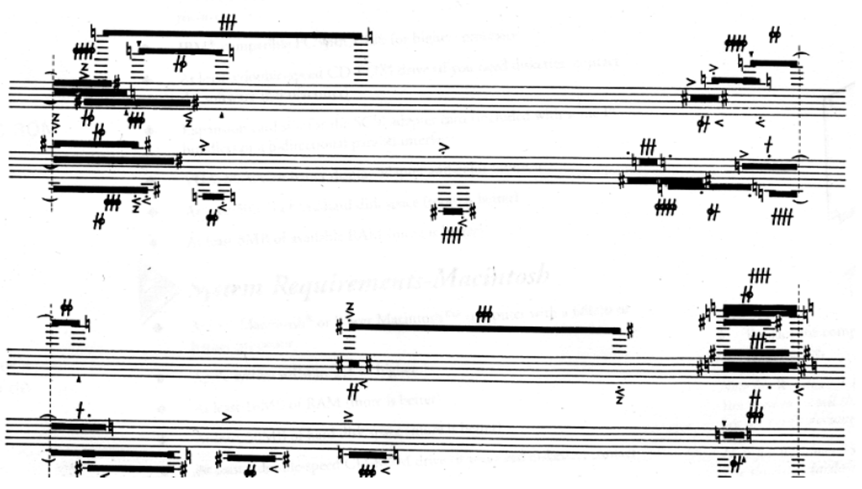


Fig. 1: An excerpt of the score: page 1, 3d & 4th systems.
© 1975 Universal Edition (Canada) Ltd, Toronto.

The variables of the piece are the following:

1. The 25 pages can be played in any order;
2. The pages can be read upside down;
3. The two line systems can be read in treble or bass clefs;
4. The total duration of the piece can range from 8 mn 20 s to 25 mn (not compulsory), for the player can choose different time resolutions;
5. The piece can be performed by 1 to 25 pianists.

Such "open form" pieces became common during that turning-point 20th century period. It is not the place here to discuss the historical context nor the conceptual background of this aesthetical option, which sounds nowadays somewhat obsolete, although it gave rise to other beautiful masterpieces such as, in the case of the piano repertoire, Boucourechliev's "Archipels", Boulez' "Third Sonata", Cage's "Chance Music", and some "Klavierstücken" by Stockhausen.

2. A web interactive simulation of "25 Pages…"

The Brown's composition represents a very interesting study case for a web interactive simulation:

- Its "full" version, with up to 25 pianists, has very few chances to be some day effectively performed on stage, for obvious reasons — what theater, what stage, what producer can afford 25 grand pianos with 25 good professional players at the same time? [1]

[1] Till the present date, we have failed in finding some evidence of live or recorded performances of this piece with more than one single piano.

Nevertheless, it suggests a very exciting and unique musical, sonic and spatial experience. Thus, there is no doubt that offering a simulation of this experience is of great musical and historical interest.

- Depending on decisions the performer (s) make for some variables, some moments of the score may turn impossible to sound or to be played: tones may fall outside the piano's range, some spreads may become excessively large for human hand, some unplayable unisons may appear, and so on.
- In some particularly dense moments, there is far more very subtle and almost microcospical indications of dynamics and attacks — generally applied independently to each tone — than the performer can securely play [2] .
- Although, in seek of realism, these shortcomings could be introduced in a simulation computer program, they can also be bypassed to obtain an absolute version of the piece where all the variables effectively and systematically affect the final result — remembering that the MIDI protocol, for instance, let simulate a virtual piano with no less than 10 octaves and a performer able to control up to 128 loudness steps.
- Unlike Stockhausen's "Klavierstück n. 11" or Boulez' "Third Sonata", there is no need to be a musical expert to understand the variable rules and control a performance of "25 pages…". This is a very important point in dealing with an "all-audiences" web production.
- Another positive aspect is that the entry points of the interactivity — number of pianos and any of the other performance variables — are very clearly audible to any non prepared subject. This means instant gratification.

These are the reasons why we choose this piece to develop a web environment where the visitor, which acts as a kind of "maestro", "artistic director" or "composer's assistent", chooses and prepares his/her own version of "25 pages…", and immediately listens to it.

We introduce another variable Brown's does not formalize, but which is implicit for any performance with more than a single piano: the way the pianos are placed on a virtual stage. In a real performance in a concert hall, and assumed that there is no electronic balance nor equalization, each piano will reach the audience from a different point in the space and with a different global loudness and timbre. Thus it is coherent with the composer's project the user can access to the performance stage and choose from where each piano will be heard.

3. The implementation

3.1 Audio vs MIDI solutions

When implementing a computer and/or web music application, the first question concerns the choice of the format. Will the "25 pages" software read, process and send forward MIDI or audio data? Assuming there is no need to describe here their respective properties, let us simply check the pros and contras of each format for each of the project's main musical constraints.

[2] The composer is fully acquainted with this problem: « There is clearly an excess of detailed information given as to the loudness, attack condition, duration and juxtaposition… excess, in the sense that all of the indications cannot be fulfilled in the more "dense" complexes » (Brown 1974). He suggests some adaptations to apply to the score in such moments.

- *Artistic performance of the 25 pages, for later digitalization*. A profissional pianist recording the pages on an acoustic grand piano is obviously the most natural solution. The recordings would then be digitalized as audio files. Using a fully weighted digital piano, perfectly simulating the behaviour of a grand piano keyboard, the MIDI recording may also be an artistic satisfying solution, from the pianist point of view (but see next topic). If needed, a very precise transcription of the subtle dynamics and attacks directions of the written score can be strictly reached, by means of a computer-assisted edition of the recorded MIDI files, a very awkward task, if not impossible, with audio data.
- *Quality of the Piano sound*. As known, MIDI does not carry any sonic data. Thus, the quality of a MIDI virtual experience of "25 pages" depends exclusively on the quality of the MIDI piano device the end-user have connected to his/her computer. Æsthetically speaking, this is a serious matter of consideration, as Earle Brown's music is essentially based on sonic acoustic qualities, rather than on abstract pitch structures.
- *Storage of the 25 pages; size of files; transfer rates on the web*. Compared to the very large size an audio file of a single page of this music will must have, even with some digital compression [3], the tiny size of a standard MIDI file is undoubtly the best choice. Small size means fast transfer and small storage needs.
- *Accurate processing of pitches and durations, in order to execute in real time the user's defined " reversions" [4], transpositions and time unit variables*. It is a well known fact that computer's "musical" understanding of an audio file is far from a trivial problem, with no satisfactory solution for the while (Leão *et al.* 2001). There is no way to ask a remote computer to make a musically correct "inversion" of a polyphonic music stored as an audio file. On another hand, good audio time-stretching technologies are available, but all are relatively slow, and thus not very suitable for our real time purpose, especially in a web environment with files as big as the "pages" would be. Moreover, longer "pages" will result in yet bigger files, causing a much slower transfer rate. No one of these shortcomings is to be expected in a MIDI-based environment: pitches and durations can be very precisely manipulated with a few simple mathematical operations which are not supposed to consume a significant amount of time nor file size. Moreover, as stated above, there is no composer's directions for pitch, duration or intensity variables that could be "impossible" to apply to a MIDI file (see however 5th constraint).
- *Polyphony of pianos (up to 25) that may eventually play identical pitch(es) at the same time*. Here is another serious strong limitation of MIDI, because the polyphonic capacity of a MIDI device may fall down to 16 simultaneous pitches [5]. And as it is not realistic to assume the lambda user may have a set of two or more chained MIDI piano modules, there is also no possibility to have the same pitch played simultaneously. However, both situations may only eventually occur when a large number of pianos are activated and play some pitches *exactly* at the same MIDI onset position. In several cases, this situation can be avoided or at least minimized (see how below). Thus, it cannot be considered to be a definitive sentence against the MIDI implementation. Anyway, the audio solution,

[3] Standard mp3 compression does not appear to be a good idea for a very sonic-dependent music as is the "25 pages".

[4] By " reversion" we mean the "upside-down" score reading variable.

[5] When an overflow of MIDI pitch data occurs (i.e. a flow of more than 16 simultaneaous events), the device uses some kind of priority algorithm, giving generally preference to the higher pitch, which is supposed to carry the melodic (i.e. main) meaning of the music, not a valid criterium for "25 pages".

which have no one of such limitations, would be a much better choice, if, again, the weight of such a number of simultaneous audio data was not prohibitive for a web interactive environment.

Due to that considerations, the MIDI option, despite its limitations, appears to be more suitable option, because of the present state of audio and web technologies, and average internet transfer rates. A secondary factor which favored this choice is that "25 pages" was yet MIDIfied, during a former student research project [6]. This constitutes a valuable shortcut for the implementation to be quickly available.

3.2 Java implementation

For several reasons, SUN's Java appeared to be the better suited programming language for this application.

- It is a multi platform language.
- It has a wide support for MIDI files, offering various classes that allow manipulation, execution and/or creation of new MIDI sequences. Besides, these classes are well documented, allowing the programmer to use the available resources in an easy and efficient way.
- It allows the application to be accessed through the web, a *sine-qua-non* condition for our project.

Our implementation consists of a homepage and a Java application. In the homepage, the visitor declares:

4. The number of active pianos.
5. How many pages each piano (pianist) will play, and in what order.
6. For each page, the two performance variables:
   - 6.1. The page position, with two options: "up" (the page is read in its common position — i.e. the MIDI file is normally played — or "down" (the page is read upside-down);
   - 6.2. The treble/bass clef rule, also with two options: "normal" (upper stave in treble clef, lower stave in bass clef — it is the way the MIDI files are recorded) and "permuted" (upper stave in bass clef, lower stave in treble clef).
7. The duration of each page.
8. The position of the piano on the virtual stage.

[6] The MIDI files have been realized by Ernesto Trajano, a pianist now post-graduating in computers, and are already available on the GMT site http://www.liaa.ch.ufpb.br/~gmt.

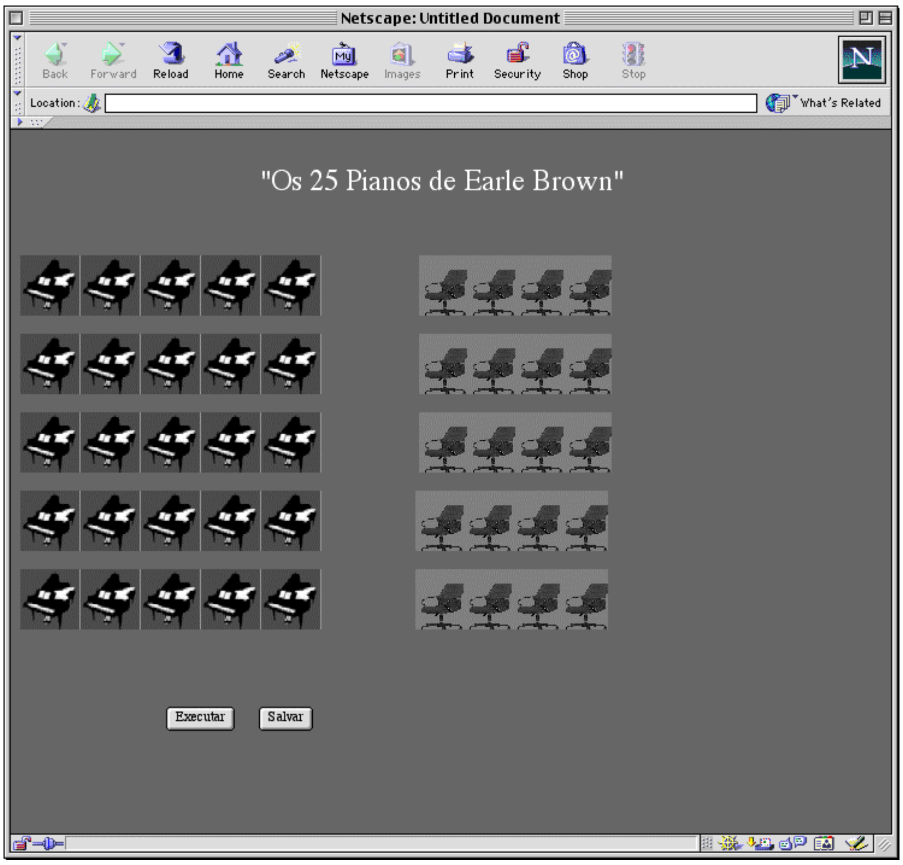
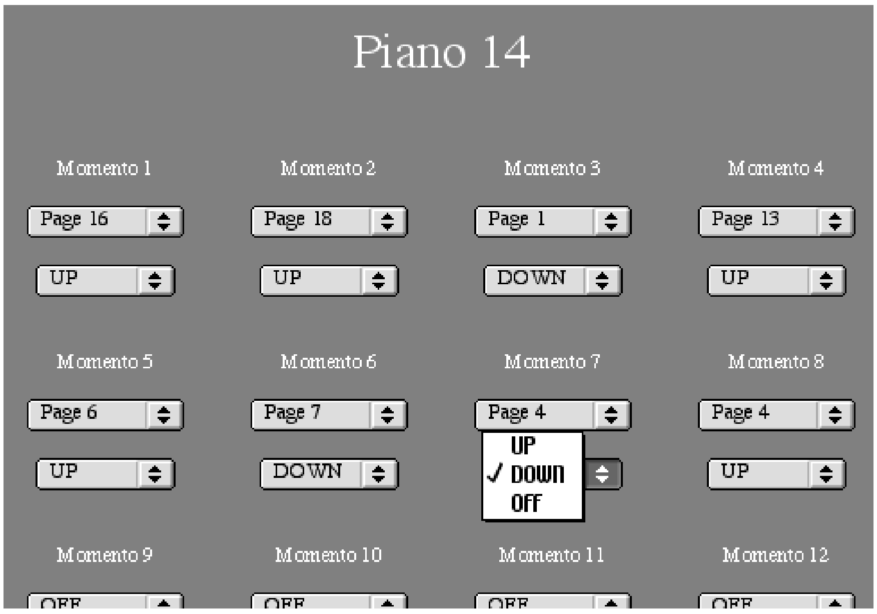


Fig. 2. : A screenshot of a first prototype of main user's interface. The final version, in english language, will represent a "true" theater. Each piano icon links to the programming page (see fig. 3).


Piano 14

Momento 1	Momento 2	Momento 3	Momento 4
Page 16	Page 18	Page 1	Page 13
UP	UP	DOWN	UP

Momento 5	Momento 6	Momento 7	Momento 8
Page 6	Page 7	Page 4	Page 4
UP	DOWN	UP [✓ DOWN] OFF	UP

Momento 9	Momento 10	Momento 11	Momento 12
OFF	OFF	OFF	OFF

Fig. 3. A prototype of each piano interface (here the 14th piano), where the user informs what and how the piano will play. This interface is available for each of the 25 pianos the user wants to activate. At the present date (submission deadline), only the variables 1, 2 and 3.1 have been implemented.

Then the Java application executes the piece according to this program. The communication between the homepage and the application is made with JSP (Java Server Pages), allowing the parameters declared by the user to be passed to the application. The application has a main class, the *Manipulador*, that does all the manipulation of the MIDI sequences. It is responsible for the concatenation of the various sequences into one single sequence, for its execution, and for ending it. This class is also responsible for the reversion function (see below).

3.3 Discussion, problems and on-going developments

While the implementation of the 1st and 2nd performance rules is rather trivial, we find serious Java performance problems when executing the 3rd variable's "reversion" algorithm. At first, we planned to have the MIDI files to be reversed on-line, according to the user's declaration. As stated above, this task would be accomplished by a set of methods available in the *Manipulador*, using the onsets and pitches MIDI data input [7]. But during the tests, we observed that the time consumed for these calculations is not adequate in a web environment [8]. This happens mainly because Java is an interpreted language [9]. Low or instable connection speeds can turn this problem even worse. Thus we resolve to build "by hand" a set of new 25 MIDI files corresponding to the "reversed" versions of the pages, so that the on-line task is limited to a pointing function to the corresponding MIDI file.

A second difficulty is due to MIDI specifications. The "clef permutation" variable is awkward to apply, at least if one want to strictly fulfill the composer's rule: « events within each 2 line system may be read as either treble or bass clef » (Brown *op. cit.*). The MIDI protocol does not have any kind of "clef data". We plan to implement this function in the following way: each page have a "pitch axis" (determined after a human analysis of the score); any pitch above this axis is said to be written/played in treble clef, while any pitch below, in bass clef. This constitutes the "normal" state of the already recorded MIDI files. If the user chooses the "permuted" version, an algorithm executes a double simultaneous transposition of pitches — the "treble clef" pitches are lowered 21 semitons, while the "bass clef" pitches are raised in the same proportion —, a relatively simple task. 21 semitons is the average interval a pitch which is written in a clef is transposed when read in the other clef [10]. We are aware we can face the same Java performance problem as for the reversion algorithm. Besides, this solution constitutes a limited interpretation of the original rule, as the composer admits both staves may also be played in the same clef. But the musical result would sound satisfactorily convincing.

The duration variable will be easily implemented by applying a multiplier to the MIDI onset and durations values. This multiplier will range from (0.5) to (2.00), up to the user's decision. Besides attending the composer's direction, this variable is also a powerful tool to avoid MIDI polyphony saturation, as a very fine tuning of the duration of each piano part can dramatically decrease the probability of several events to occur at the same onset position.

---

7 The onset reversor takes the highest value (i.e. the last event in the file) as the starting point and play back. The pitch reversor works on the central value MIDI 60 and reverses each other values (MIDI 61 becomes MIDI 59, and so on).

[8] It took several minutes on the local computer.

[9] Compiled languages such as C++ e Delphi would have developped in this case a better performance.

[10] This is an average value, because, due to the diatonic structure of the piano keyboard, such "reading" transposition (not a truly "musical" one) is not evenly distributed. Upward transpositions from E and B, and downward transpositions from C and F, count only 20 semitons.

The last variable — position of each piano on a virtual stage — involves a rather complex MIDI manipulation of three parameters: the *pan* — the standard #10 MIDI controller —, the *main volume* — #7 controller — and a distance simulation through the MIDI control of room or reverb effects (like the #91 controller). The musical result that these controllers may produce depend exclusively on the user's MIDI device capabilities. Apart of the main volume, they are only fully implemented in professional devices. We are thinking about a custom-implemented controller. Anyway, this variable may be bypassed, or limited to pan and volume controls in a first version, as a spatial control of the performance is not explicitly requested by the composer.

Finally, due to polyphonic MIDI limitations, the user should be advised to avoid to start the music with all pianos activated, and to connect more than one piano to play the same page at the same time. Nevertheless, as the exact behaviour of MIDI data flow depends on Internet conditions and on the user's local configuration, it may appear polyphonic saturations when a high number of pianos is activated. It means that some notes may drop or miss. As we have already suggested, a careful mapping of the duration variable will undoubtly help to avoid a great number of such shortcomings.

Anyway, the MIDI issues do not appear to invalidate the project as a whole, nor the musical experience it will certainly provide. Otherwise, we must thoroughly test the on-line Java performance, in order to study some kind of alternative if necessary. We will implement a "default" version of this piece, immediately executable by any new visitor. More, save/open functions should be implemented to allow, not only the user to keep recorded his own version on his hard disk, but also on the remote site, in order to other users be able to listen to and download.

This project is planned to be fully achieved and published till July 2001. It will be freely available at the GMT site: http://www.liaa.ch.ufpb.br/~gmt [11].

4. Acknowledgements

The authors thank Ernesto Trajano for his advice and help in programming.

5. References

Brown, Earle (1974). *25 pages for 1 to 25 pianos*. Toronto: Universal Edition, 1975 (originally composed in 1953, but the quoted text is dated "May 1974").

Delaigue, Oliver (1989). Les Nouvelles Musiques Americaines en France, 1945 – 85. Dissertation in Musicology, Paris: C.N.S.M.D.P., 1989.

Leão, H.B.S., Guimarães, G.F., Ramalho, G.., e Cavalcante, S.V (2001). Benchmarking Wave-to-MIDI Transcription Tools. Electronic Musicological Review, 2001, Vol. 6 N.1. http://www.cce.ufpr.br/~rem/.

Nyman, Michael (1999). Experimental Music. London, Cambridge, 1999, (originally published1974).

Ryan, David(1999). Earle Brown - A Sketch. http://www.earle-brown.org/, 11/1999.

Sun Microsystems. Application Programming Interface (API) Description, Java Sound Midi, Java Server Pages. Available at http://java.sun.com/products/jsp/.

[11] The Earle Brown's software is part of a main research project on 20th century music, coordinated by Didier Guigue and supported by the CNPQ (Brazilian Council of Research).