INTERACTION

INTRODUCTION

Currently there is considerable activity towards development of programming environments for computer music. In the IMSU environment, a control system based on a programming language, the MUSO system is a major contribution. MUSO is a modular environment, for manipulation of musical objects. The purpose of this article is to describe the technical solutions adopted for the MUSO application.

PROJECT HISTORY

The very first version of the MUSO project dates from 1983, when CRMR was equipped with an 8-bit computer. This computer became a suitable environment for electronic music. The necessary equipment was almost available, in the form of the Apple II, equipped with a simple interface. MUSO is a modular computer language, suitable for musicians to easily describe control sequences for synthesis. Since then, numerous questions have been posed. The choice of LISP as a language has not been reached yet, and it can be used by children. It should be accessible to musicians. (Paper 80. However, LISP allows every user to create his own vocabulary and to adapt the machine to his specific requirements.

The general philosophy of the project was to use the form of a computer language to describe the compositional process and tools, i.e. to provide some essential components, and to define the MUSO concepts in a complete set of primitives, efficient in the sense of formulating a musical idea.

The system is implemented in 8-bit computer, the implementation chosen for this project is a modular computer language, with the ability to deal with multiple sequencers, changing objects, discrete events, internal synchronisation etc. This version was presented at the 1984 CMC in Paris (Oliver 84).

Finally, in 1988, the version became far more stable and we decided that it was definite. Subsequently, we submitted a paper to the conference. The next step was to prepare a document for the conference, which we are preparing. In parallel, a distribution agreement for France and the MUSO system should be effective before the end of 1988.

ARCHITECTURE OF THE SYSTEM

The MUSO system is composed of three sections:
- a LISP interpreter, supplemented with a number of basic instructions permitting description of compositional processes and definition of many types of event manipulation.
- a 128-track sequencer, which, from a small reproduction and recording facilities, offers extensive sound synthesis, filtering options for the various LISP messages and a choice of concert setup to govern ensemble progression.

A musical editor working directly on MUSO sequencers on the basis either of an alphanumeric representation, a graphic display, or graphics. This is the principal output of the system.

The three modules described in greater detail, we examine its essential components, namely the MUSO drivers.

SI MUSO drivers

These serve to deal with all problems of communication, transmission and reception of MUSO messages, monitoring of coming status, filtration of messages, allowing for time and synchronisation.

Special care has been taken in the production of these drivers to ensure the maintenance of synchronisation under all circumstances and to ensure communication, simultaneous transmission and reception at full MUSO speed, irrespective of the system load, principally in the MIDI input and output device operation of all real time tasks.

Driver configurations described in a parameter table, allowing the user to access and modify the drivers.

211
function, particularly adjustment of the tick size.

Filter function for all types of message, particularly for real time, and synchronization options depend directly upon the choice of filtration of these messages.

Tick size is defined through two parameters, a value defined in beats per minute and a beat value defined in number of ticks. tick size = 60 000 (synchronizer temp) For a tick size of 1:16, for example:

- beats = 256
- tempo = 160
- beats = 100
- tempo = 80

If a measure of 00 ten is defined, time will be represented in minutes seconds and hundredths which can be particularly useful when using the graphical edit-

2) The Musical Editor

At any moment, the editor permits visualization or modification of sequence contents.

There are two possible forms of visualization:

- a full-screen alphanumeric representation in tabular form on which the exact content of a sequence may be seen and manipulated
- a graphic visualization, not in the traditional line, but with mechanical graph representation imposed by velocity as well as a sub-pixel line reference grid. Numerous mouse levels are possible for direction movement of the score.

With either form it is possible to insert, modify or cancel events, just as it is possible to call for immediate feedback from the sequencer.

3) The Sequence

The sequence can be brought to either from the editor or from the MSQ2. It can also be recorded to a sequencer simultaneously as well as processing all tip of MSQ messages, including system exclusive. The interesting point is the possibility of recording meta-events in the sequencer, namely commands passed to the sequencer itself.

The sequence permits change of tempo, reassignment of channels, starting in addition of other sequences as well as stopping these same sequences.

1) Musical Data Bank

The role of the data base is central to the architecture of the system. It is organized into 128 independent sequencers, a sequence itself being a list of sequenced time-tagged events. Moreover, all events get stamped with an MSQ channel destination number. Although this solution came up as mere memory space, it allows interchanging of sequences without loss of information and single pass recording of data originating from another sequencer and generally manipulation of channel data in the same manner as other information.

3) The musical primitives

MSQ2 offers approximately 50 musical functions and a complete command structure would be immense. However, some seem to be important because of the changes MSQ2 has undergone. A more detailed description will be given. For the others, I will give the general concept of construction.

a) the first of these concerns the creation and manipulation of elementary events, the events manipulated by MSQ2 are of three types: musical MSQ events, as used in the score, including the elements frequencies (pitched by limiting several elementary events), including also a certain extent, real time messages.

b) messages carrying a direction field and which are transmitted by the sequencer, 'key on' and 'key off', as well as events which you command to the sequencer itself.

All these events are treated with the same MSQ instruction:

```
TO SEQUENCER: Tempo NOTE 255 [TEMP] 0 0
```

TO PROGRAM: EventCH NME 192 [PROC] 0 [CHN]

TO EFFECT: EventCH MNS 128 [FITCH] 191 [CHN]

TO EVENT: EventCH MNS 128 [FITCH] 191 [CHN]

The other instructions concern the access to and modification of modules of the system constellation as events:

The instruction DATE, for example, permits modification of the date of an event.

```
DATE ALT
```

dispaces the current value by two time units,
b) the second family concerns the problem of position on the track and their displacement. To allow movement of this family is the instruction DATE which positions the insertion pointer to a given date.

To produce X at date 99 we write:

```
DATE 99 NOTE 00-99 12
```

Once I would like to mention a historical point: in the first version of MSQ2, the proposed musical notation was in this form. Since then they formed a complete set, any higher level manipulation could be...
written in this book using MUSC.

In practice, it was found that the execution time
capability was prohibitively high. Therefore we decided, at
assembler level, to implement a group of higher level
primitives. Again, the ultimate source of flexibility was manipulated: algorithms for MIDI sequence, let us say, a string
s = "MIDI LISP = LISP based

MIDI programming environment

for the Macintosh", Proceed-
sings of the International
Computer Music Conference

(Paper 80)
S. Papert
"Mindstorms: children, com-
puters, and powerful ideas.",

(Orlinsky 86)
C. Orlinsky
"MUSCUS as language de pro-
grammation musicale et com-plexnostyrislation Musique
Proceedings of the International
Computer Music Conference
1984.

Contact:
"MUSCUS 4006
Studio 86
A. Gail Jean Moody
4006" L.0. MUSCUS
79, 90 (2) (I).

ICMC 86 Proceedings

213