SOFTWARE TOOLS FOR ELECTRONIC WIND INSTRUMENT PERFORMANCE

Prof. Bruno Degazio
Sheridan College
School of Animation, Arts & Design

ABSTRACT

Even though alternate musical controllers such as the Electronic Wind Instrument (EWI) significantly extend the expressive range of MIDI synthesizers and software virtual instruments, the computer-based editing and manipulation of data produced by these controllers has remained in an undeveloped state. A primary problem has to do with the binding of note data to expressive information such as breath controller and pitch bend data. MIDI, originally designed as a real-time performance protocol, has very weak binding of such data into higher-level musical structures; so weak in fact that even note endings are unconnected to their beginnings. The author describes a binding paradigm and data structure which addresses this problem and which has proven very effective in the manipulation of recorded EWI MIDI performance. Four further challenges - data integrity, spurious notes, erratic velocities and inaccurate rhythm - are described along with the software solutions which have been developed to deal with them.

Keywords – MIDI, music performance, electronic wind instrument

1. INTRODUCTION

Wind controllers greatly extend the expressive dimension of MIDI. They are arguably the only application of MIDI which takes full advantage of the potentials inherent in the MIDI specification (1984). They do this by generating a rich stream of MIDI data messages including notes, breath control, pitch bend and other messages.

However, much of this potential is ignored by commercial MIDI editors such as Apple Logic, Steinberg Cubase, Digidesign ProTools, and others. The keyboard performance paradigm implicit in the MIDI specification is assumed in these commercial applications. While adequate for simulating percussive instruments such as drums and piano, most other instruments require some sort of continuous control.

MIDIForth (Degazio 1987, 1988, 1993), the author's Macintosh-based MIDI editing system, has evolved to address this lack. This paper describes recent developments which focus on the requirements of continuous performance MIDI instruments such as the Akai EWI 4000 and the Yamaha WX5 electronic wind instruments.

2. VISUAL EDITING

In order to deal effectively with notes and associated continuous data, a visual editor must be able to display such data simultaneously. This capability is notably absent from many commercial MIDI editors. MIDIForth allows the simultaneous presentation of notes together with breath controller data, pitch-bend data or both (figures 1-3).

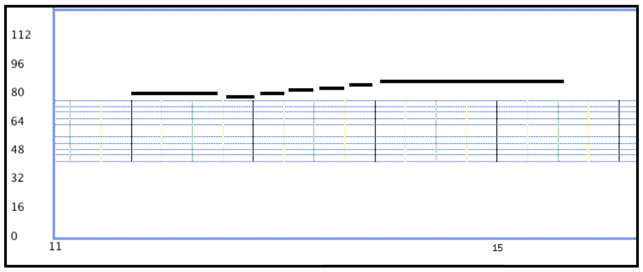

Figure 1. Visual Editor displaying notes only

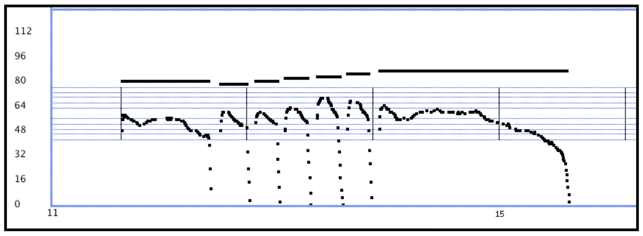

Figure 2. Notes with breath control data superimposed

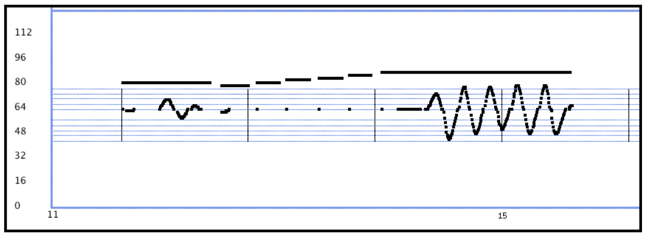

Figure 3. Notes with pitchbend data superimposed

3. PROBLEMS WITH EDITING EWI PERFORMANCE DATA

3.1 Note/Continuous Data Integrity

A fundamental problem with computer-based manipulation of EWI performance data is that the continuous control messages - usually breath control (MIDI CC#2) and pitch-bend - are not bound to the note data to which they apply. Any application intending to manipulate the complete performance event - notes plus all associated control data - must somehow logically bind the various data together.

A practical method of binding such data consists of simply including all events contained within the time boundaries of a given note-on event. For this purpose, MIDIForth employs as its basic data structure the *atom*, defined as follows:

```
AtomStructure
long:  +DurFld            \ # ticks to next event
long:  +LengthFld         \ for notes only, in ticks
long:  +AbsFld            \ absolute position from start
byte:  +StatFld           \ MIDI status code
byte:  +Data1Fld          \ pitch, cc#, PB low, program#, etc.
byte:  +Data2Fld          \ vel, cc value, PB high, etc.
byte:  +TagFld            \ editing selection tags
\ total 16 bytes
```

The start position of a MIDI event is given by its Absolute Position field (+AbsFld). For note events, the length of the note is contained in the field labeled +LengthFld . The end position is thus easily calculated by adding the LengthFld to the AbsFld. A quick search including all MIDI events contained within these limits binds the relevant data to the note for subsequent operations. Many of the operations described below automatically bind continuous data to notes in this manner. For convenient manual manipulation, a menu command, *Select Within Note* is also provided.

However, on recording actual data from an EWI an immediate complication is discovered: important data exists outside the start-end boundaries of a note. For example, the following illustration shows an actual recorded note with its breath control data superimposed (figure 4).

In this example, the first three breath data points and the final one are all outside the boundaries of the note. This is characteristic of the behavior of the Akai EWI. Other EWIs from Yamaha and other companies also produce continuous data before or after the note.

When the note actually sounds it will do so at a setting corresponding to the last received data values for breath pressure and pitch bend. A problem arises if subsequent editing of the data does not take this into account - there will be a noticeable alteration in the sound produced by the synthesizer.

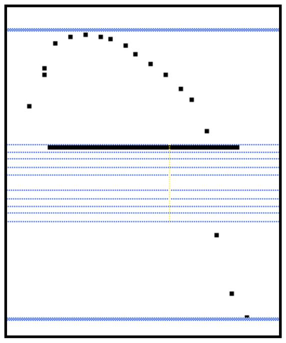
Figure 4. Extraneous breath data

For this reason MIDIForth includes a function, *EWI Cleanup*, which forces the data integrity of each note event by adding data, if necessary, to the start of the note. *EWI Cleanup* searches through a recorded MIDI performance and deletes all breath controller (BC) and pitch-bend data that exist outside the boundaries of a note. No information is lost because the routine first inserts the correct datum at the beginning of the note, which is where it would have its first audible effect anyway. Figure 5 shows the event data after application of this function. This function guarantees that every note event begins with accurate breath controller and pitch-bend data.

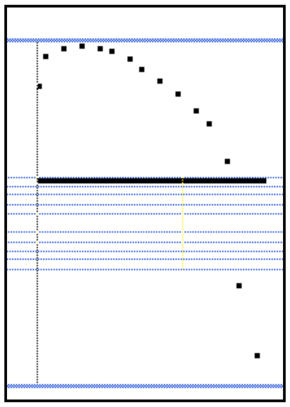
Figure 5. breath data cleaned

3.2 Spurious Notes

Another complication comes from the large number of spurious note messages produced even by expert players on the instrument. Though these notes are for the most part inaudible, they interfere with many subsequent data operations, including quantization and transcription into common music notation. Figure 6 is an example of a musical passage as recorded with an Akai

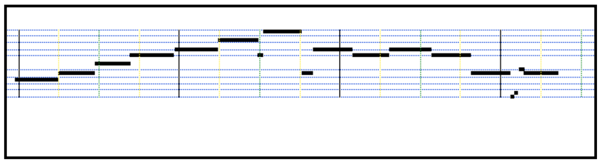
Figure 6. Spurious notes

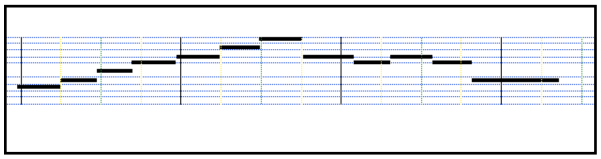
Figure 7. Spurious notes melded to adjacent notes

EWI 4000s. Note the short spurious notes that occur at note transitions.

The MIDIForth function *Meld Short Notes* automatically searches through a recorded passage and “melds” (joins) notes shorter than a specified duration into the adjacent longer note (figure 7).

3.3 Erratic Velocities

Another curious characteristic of EWIs is that the note velocities they produce are quite inaccurate. This appears to be due to the fact that the note velocity must be determined at the *beginning* of a note, before the player has had a chance to fully produce a breath envelope. This makes sense in the context of a percussive instrument such as the piano, but not with a continuously controlled instrument like an EWI. A more correct velocity value is related to the average breath pressure throughout the duration of a note. MIDIForth provides a function that computes the average breath pressure value contained within a given note and converts the note’s velocity datum to correspond. This approach, though simple, appears to work quite well

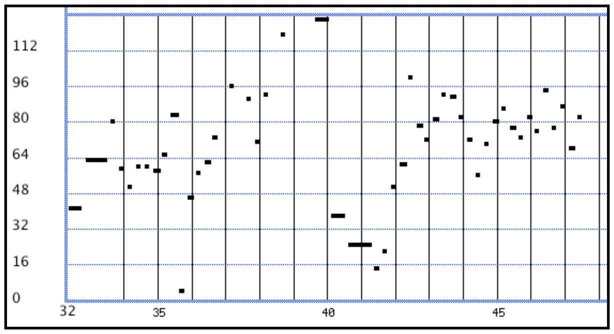

Figure 8. Erratic Note velocities produced by EWI

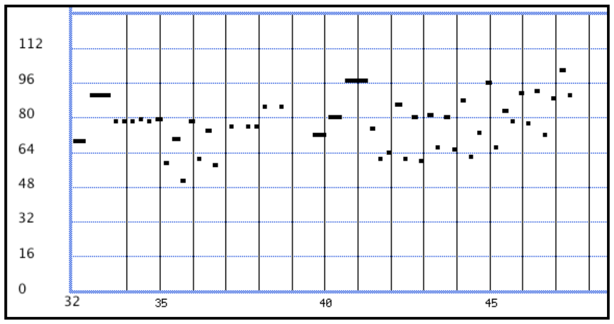

Figure 8. Note velocities adjusted according to average breath value through note.

in practice, and produces a set of velocity values which can then be applied to other, more conventional MIDI instruments, such as a sampled piano.

3.4 Inaccurate Rhythms

Sloppily played rhythms can of course be a problem with any sort of musical performance, not just with an EWI. However, in commercial applications, software methods exist, usually called *quantization,* which are used to correct rhythmic errors. Unfortunately, commercial quantization algorithms do not work with recorded EWI performances. The difficulty is related to the problem of data binding discussed in section 3.1. An EWI note event consists of an entire suite of MIDI messages, including but not limited to breath data and pitch-bend, which must be moved as an integral unit. Commercial software, ignoring this requirement, simply moves the note message to a rhythmically correct position, leaving the associated continuous data where it was. This of course results in a completely unusable mess.

In addressing this problem, the first step is of course to implement a binding protocol like the one described to ensure that notes move along with their associated data. However, this is not sufficient to produce an acceptable result. Two other characteristics of the EWI MIDI stream are of critical importance. They are, a) note overlap (tonguing) and b) the fact that the data stream is more or less continuously sampled.

3.4.1 *Note overlap* concerns the relative position of two notes in time. If the the second note begins before the first note ends, they are said to be *overlapped* or *legato* (figure 9). If the first note ends before the second begins (by as little as even one tick) the notes are *not overlapped* and are said to be *tongued, detached*, *staccato*, etc as the case may be (figure 10). It is this feature of the data stream that a skilled player controls by means of subtle changes in *tonguing.*

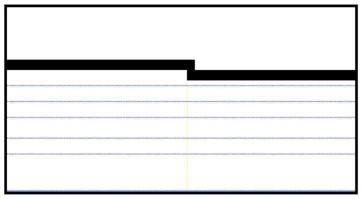

Figure 10. Overlapped (legato) notes.

A problem arises if the integrity of the overlaps is lost during the course of rhythmic manipulation or other editing. The edited performance will have a different “tonguing” than actually performed by the player. Therefore, any form of automatic rhythmic manipulation must take into account the status of overlaps between adjacent notes.

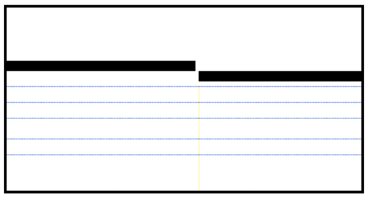

Figure 11. Tongued (detached) notes.

3.4.2 The second complication arises from the fact that an EWI note event is not a discrete data point but is a stream of related events spread through a definite time. A simple example illustrates this. Suppose we have a legato musical passage where a note has been recorded a little later than it should have been (figure 12). If we simply slide the second note event and its associated data to begin at the correct position (the light vertical line) we produce a collision with the breath control data at the end of the preceding note (figure 13).

The correct method of handling this problem was derived, after much trial and error, from the practice of cinema dialog editors, who regularly use *time-stretching software* on audio files to fit individual words to lip movements on screen. A time-stretching approach was applied to individual notes, compressing or expanding their duration by a factor exactly producing the desired position in the *following note* (figure 14). When carefully implemented, this procedure has the advantage of also preserving tonguing overlap integrity, thus solving both problems.

4. ACKNOWLEDGMENTS

The development of MIDIForth has been funded by the Canada Council for the Arts and the Ontario Arts Council.

The software described has been developed on the Macintosh computer using Carbon MacForth from MegaWolf Software Inc., New Haven, Connecticut.

5. REFERENCES

[1] Degazio, B., "The MIDIForth Computer Music System", in *Proceedings of Printemps Electroacoustic,* Montreal, 1987

[2] Degazio, B., "The Development of Context Sensitivity in MIDIForth" in *Proceedings of the International Computer Music Conference,* Koln, 1988

[3] Degazio, B., "New Software Composition Tools" in *Proceedings of the Fourth Biennial Arts and Technology Symposium,* Connecticut College, New London, Connecticut, 1993

[4] Degazio, B., "A Computer Based Editor for Rhythmic Structures" in *Proceedings of the International Computer Music Conference*, Hong Kong, 1996

[5] MacFarland, Ward, *Carbon MacForth*, MegaWolf Software, New Haven, Connecticut

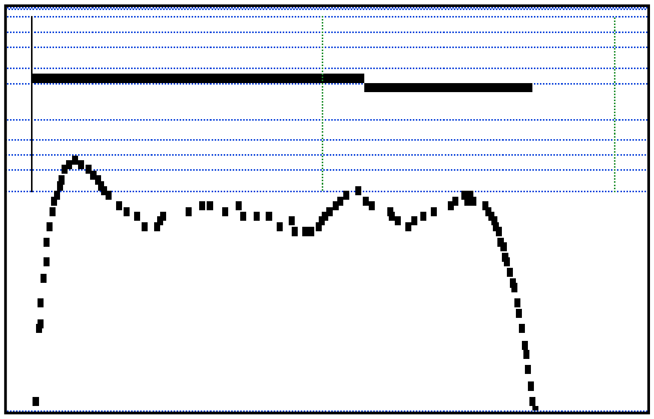
Figure 12. Legato passage, second note slightly late.

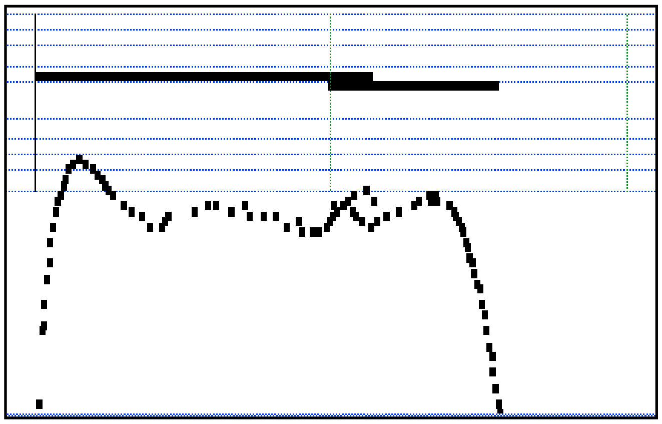
Figure 13. Second note adjusted, collision with preceding note.

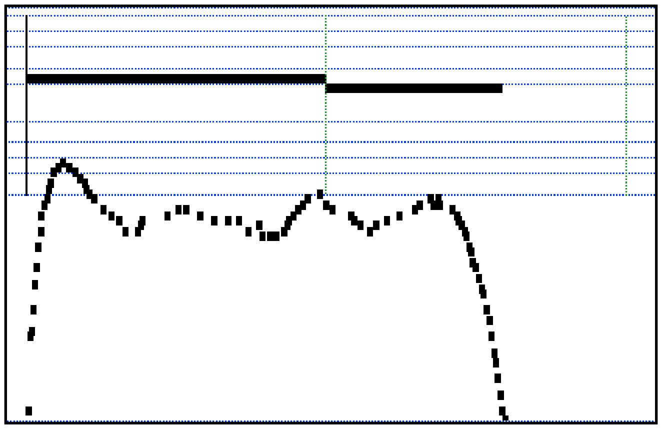
Figure 14. Second note adjusted, preceding note time-stretched.