Do Composers Think?

Nigel Morgan

Thinking involves seeing the abstract structures that link our sensations and our feelings. In the process of thinking we look for underlying patterns and compare them. This leads to logic. [1]

Thinking is the logical manipulation of symbols. [2]

When you listen to music you gain sensations from that experience. At one level those sensations are about *distinctions*.; in register, tempo, timbre. Composers are very sensitive listeners because they imagine sensations. Your feelings for music demand a greater sense of involvement than passive listening and the experiencing of sensations. You connect to music to get a response and in turn become responsive. In composing music, sensations and feelings are drawn together by creating patterns and structures that become active symbols for the way you think.

Of course, those of us who write or create music are normally too busy with the current project to spend much time considering how we are thinking. We have acquired a technique and the basic principles of musical practice to allow us, no matter what our sensations or feelings, to produce a result, be it in score or directly in sound.

Most composition, however, involves manipulating symbols. Staff notation is, after all, intensely symbolic. What is a quarter note but a symbol on paper. Its meaning is tied to so many variables. And even those who work with sound directly can't avoid decision making and discrimination based on symbolic organisation.

The Challenge of Information Technology -

There can be little doubt that computers now challenge composers' pattern of thinking and creating. We have an emerging generation of composers who may have only experienced composition through the medium of IT; a serious concern for the music educator. Serious, because we don't yet understand the effects of computer simulation and interaction on the way we think our music through.

Computer sequencers are encouraging composers to bypass a stage of thinking and aural imagining. We can get our 'hands' directly on simulations of sounds and immediately engage in playful exploration. Our thinking becomes a response to performance detail rather than a deeper association with the elements of music. Why worry about pitch rows and rhythmic sets when your ear can lead you. This is rather like saying to an architect - why don't you just draw a picture of my new house rather than go to the trouble of making a survey and drawing up a set of plans!

---

[1]Rucker.R (1987) *Mind Tools: the Mathematics of Information.* Penquin.
[2]Marshall. G (1990) *Advanced Students' guide to Expert Systems*. Heinemann.

With music IT the relationship between thinking and doing is changing. The sequencer-user composer responds to 'an idea' by recording it. The idea immediately collects all the baggage of a performance; tempo, articulation, dynamics, timbral inflection and instrumentation.

On paper a composer's 'idea' is more likely to remain in an abstract (and symbolic) form; as a pitch row or series or as a rhythmic cell. Rarely does even a 'motif' come 'fully-fledged' to paper with every detail of its performance characteristics. Because the idea is in this more abstracted state it is easier for it to be flexible to change and development. Once recorded on a sequencer 'ideas' tend to stay put!

The MIDI sequencer is not designed to respond to the more abstract modes of composing. There is rarely much on offer beyond transposition, inversion and retrogression. Most sequencers have been designed to capture the inspirational performance; to serve an area of music-making where performance characteristics -'the feel' - is an integral building block and where interpretation is captured as part of the composition.

There should be a warning on all MIDI sequencers: 'Can Cause Dangerous Loss of Aural Imagination!'. When one creates a composition, it is realised in the neural level of the mind as an n-dimensional form field, which is then mapped on thought formalizers(language, mathematics etc). These are used to formalize it in a one-to-one way so that it can be transferred in sequence to a receiver. Instruments and sounds are the most outer level of the message. If one concentrates only at sound level, one cannot create a full-bodied composition. The sound level is still very important, since at that level the mind remaps to the music, when listening. [3]

Composers who write rather than record their music often intend their work to have more depth in its musical argument- a 'deeper' structure that has the effect of binding the music invisibly to a thread of reason. This may or may not become apparent to the performer and listener through handling and interpreting the 'symbols' of music notation.

This deeper structure is often a myriad of interlinking structures that provide a frame or map onto which the drama of creativity is recorded. Just as visual artists make sketch after sketch, experiment after experiment, in order to gain fluency in 'performing' an idea with all its variables in images or shapes, the composer has to formalise, to contain, to structure, to compose.

The interpretation of symbols comes from a lengthy acquisition of stylistic understanding and practice. A performer responds to the symbolic information of a score by making connections, decisions and relationships based on past experience and experience of the past in both aural and physical ways. When new musical concepts and symbols appear we expect the composer to provide verbal or musical instructions; the knowledge to make the musical symbols understood.

If our knowledge about music can be described and worked with at a symbolic level why are n't composers using systems that are able to manipulate the structures with which knowledge is represented; systems that use programs that are *descriptive* (I would like to see all the possibilities of this pitch series by a forward rotation of one step at a time - I don't care how it's done) rather than *prescriptive* (I shall have to write a program telling my computer exactly how to do this operation). Curiously enough, the resulting programs are freer from errors and more amenable to change.

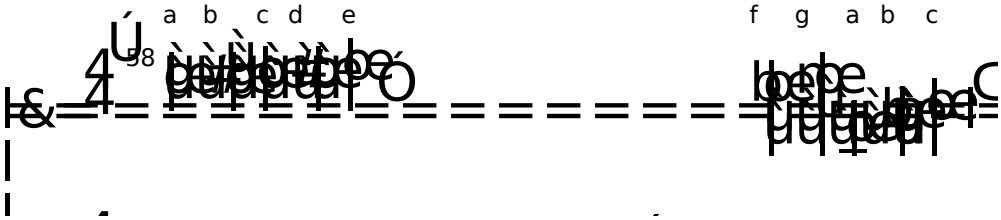
---

[3] Tolonen.P (1991) fax to the author 30/10/91

In order for a program to be descriptive it has to know a lot! It has to have a *knowledge base* which contains all the knowledge it needs, and the ability to work with that knowledge in a formal way. Of the computer languages which respond to these conditions *Smalltalk*, *Prolog*, and *LISP* have become popular choices for an increasing number of software developments for composers. Most of these, however, are concerned with the creation and manipulation of sonic material rather than providing a computer tool for symbolic composition of the abstract elements of music.

There is one 'collection of composition tools' that breaks this mould. It is written in LISP and provides the sequencer-user composer with 'tools for the mind which encourage you to develop your composing skills and horizons'; an 'expert-system' for composition. It's called Symbolic Composer.

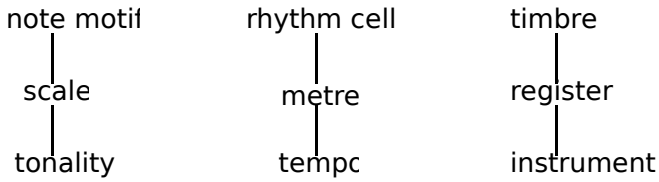
Composing with Symbols -



Let's look more closely at the relationship between music and symbols and in so doing learn about the basics of LISP and Symbolic Composer.

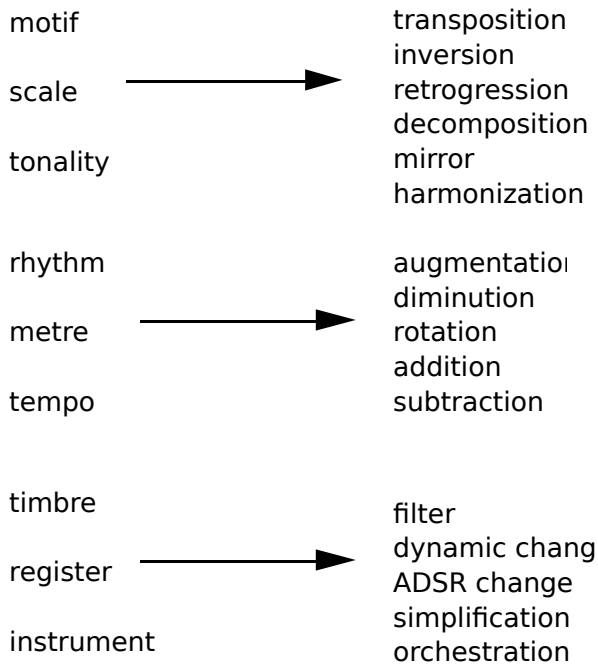
Musical textbooks on composition invariably partition technique into work with discrete musical elements; pitch, rhythm, timbre. Of course, composers don't, on the whole, work with these in isolation. As one element progresses and changes it affects another, and another. For example, a transposition in pitch affects an instrument's register and timbre. It also is rare though for composers to immediately hear a whole composition. Mozarts and Shostakovichs are rare phenomena. However, for the purposes of learning it's useful to make partitions.

Here are three common routes to composing in which an element of music data is 'mapped' to larger element assuming a deep symbolic structure.



It is not unusual for composers to turn this table on its head and work the other way around, particularly in jazz, rock and commercial music. Either way 'mapping' takes place.

In order to progress from any of these starting points composers traditionally acquire and apply certain *functions*.



These functions have been augmented in recent years by the new languages of electroacoustic music, most notably in the outline of spectro-morphology by Denis Smalley. [4]

The use of functions becomes natural and fluent to the composer who may well seek to invent special formulas and routines that go further into mathematical procedure and statistical probability. All these functions have to work within the constraints that we learn and build into our knowledge base. We know that we can't transpose a phrase for a violin below a certain pitch. Our creativity is always tempered by such constraints which very often supply and apply a frame for structure.

Now to *symbols*. We could just as easily say:

e is a and q is b

It would n't take us long to learn to think rhythms like this:

a a b a a b b b a b a rather than this:

e e q e e q q q e q e

Our computer would be much happier using a and b than the staff symbols. However, as musicians we know that our quaver or eighth note is a symbol for a rhythmic value which is not only inexact in itself (suppose I put a staccato dot over it) but depends on so many variables before its value can be defined.

[4]Emerson. S (1986) *The Aesthetics of Electroacoustic Music*. MacMillan

In MIDI, with its timing clock resolution of a quarter beat to the value 24, our eighth beat gets a value 12. If we work entirely with step-time or quantized values this can become an exact symbol. But, on paper or as we think, is an exact definition necessary? Surely the *idea* of the eighth beat value is enough.

If we retain our symbols a and b instead of the eighth and quarter beats we can , through simple algebra, develop our thinking about their possible relationships and constructs.

Writing our symbols directly into a LISP interpreter - an interpreter makes the computer respond to every line of code you type in - the response would be:

IN: a a b a a b b b a b a
OUT: no such variable as a or b (in other words the computer needs to know how to recognize a and b as something in particular)

If our computer could be taught to recognise musical symbols we could then do this:

IN: (setq a '(e)) - a is now always (e)
IN: (setq b '(q)) - b is now always (q)
As it stands a and b can be used in any tempo or metre. They are now in a very abstract but symbolic form. They have become symbols for a unit of rhythm.

In LISP brackets or parentheses are used to define a *list* of symbols or as LISP prefers to call them, *atoms*. If we now wrote them into the computer like this:

IN: a a b a a b b b a b a - the computer would respond with
OUT: (e) (e) (q) (e) (e) (q) (q) (q) (e) (q) (e)
- it would not think of the values as a rhythm. However, written like this our computer now has a word, rhythm, which contains all the data for the combination of rhythmic values.

IN: (setq rhythm '(a a b a a b b b a b a))

Now, every time we type in the word - rhythm - LISP will respond with:

OUT: (a a b a a b b b a b a)

The computer now knows what rhythm is even if it does n't know what a and b are! We could then define a and b to be literally anything we like.
a could be a rhythm in itself q. x x b could be r or h

Whatever we decided as values for a and b our *variable* word - rhythm - would have the same deep structure attached to it.

If we can do this sort of thing with rhythm, why not pitch. Let's take a short pitch motif. It could be c f# d g.

If we think motif > scale > tonality we could end up with this:

motif > c f# d g
scale > G major, (any) chromatic, C lydian, Mode III (from Messaien's modes of limited transposition) my own scale (d f# g a b c)
tonality > G major, E minor

If we convert the motif into symbols for a whole-tone scale and chromatic scale, both starting on middle C, this would be the result:

motif > c f# d g - G major mapping > (d g e a)
motif > c f# d g - C chromatic mapping > (a g c h)

In LISP we would set out our motif and its tonality like this:

IN: (setq motif '(d g e a))
IN: (setq tonal (activate-tonality (major g 5)))

As the motif is symbolic it can be applied or mapped to any scale or tonality. It can also be developed and transformed using all those functions listed earlier, and a few more besides! Here is part of the list:

• compress, inverse, mirror, repeat, scale, scroll, separate, shift, transpose, trim

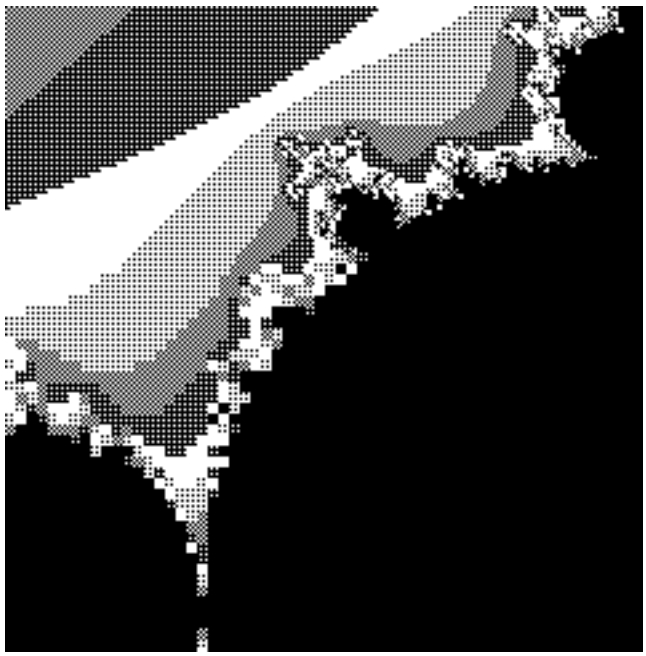
- and with a second motif:

• mix, remove, remove-pattern, find-common, transform

The most fascinating functions of all are in fact from symbol generators. These create new symbol patterns using recursive symbol definitions. *Recursion* is the ability of a piece of information or 'object' to recur by constantly going back on or 'calling' itself. If that material is defined in particular and simpler ways the recurrence does not end up being a straight repeat or loop. A pattern is generated which can explain a great deal about the nature and structure of our information or object. It can be used to solve a problem in terms of itself. [5]

[5]Friedman. D & Felleisen. M (1987) *The Little LISPer*. MIT Press

The best analogue for recursion is the fractal pattern. *Fractals* are those wonderful graphic images that appear to resemble natural phenomena. When applied to musical elements such as melody, rhythm and dynamics a similar 'natural' quality can be obtained.



Symbolic Composer has a number of symbol generators that can create the most elegant fractal patterns with musical material. Let's explore the potential of our motif with one such generator.

First, we have to make some definitions. These outline particular relationships between certain notes inside (and outside if you wish) the motif. We make associations between pitch symbols like this:

```
IN: (defsym  a' (d e))
```

This says symbol a is always associated with symbol d and e. If you were to play a in your recursive variations you would need to play d and e to follow it - always!

```
IN: (defsym  d '(c b a))
IN: (defsym  b  '(a b ))
IN: (defsym  e (d b e))
```

Now, we can ask the computer to provide us with recursive definitions of a particular symbol for any number of recursions like this:

```
IN: (listdef a 3)
OUT: a
        d
            c
            b
                a
                b
            a
                d
                e
        e
            d
                c
                b
                a
            b
                a
                b
            e
                d
                b
                e
```

```
IN: (listdef a 2)
 OUT: a
        d
            c
            b
            a
         e
            d
            b
            e
```

```
IN: (listdef a 1)
  OUT:  a
            d
            e
```

Instantly our motif sports an amazing amount of new material - and we have only defined a!

Music and Numbers-

Musical elements can be described as easily in numbers as in symbols. Just as there are special recursion generators that work with symbols, there are *vector* generators that work with real numbers.

The term *vector* is used in mathematics to distinguish between two classes of measurement. If we are measuring temperature, mass or speed we use one system of units, scalars. The second system, vectors, covers measurements involving a magnitude and a direction, such as force, acceleration, or velocity.

In Symbolic Composer there are numerous generators that produce vector patterns. These vectors can be mixed, filtered, amplified, modulated, quantized - yes, the terminology is already familiar to those of us who work with synthesisers. There is, in fact, an on-board digital synthesiser containing an unlimited number of sine, ramp, triangle, square and noise generators along with digital mixers, filters and modulators. Each oscillator has controls of volume, frequency, modulation depth and phase angle. Although it will be a while before this synthesiser is able to be used for sound itself, it can be used to express and model musical elements in mathematical and acoustical procedures. Here is an example:

I have a chord sequence; Caug7, F#aug9, Eminmaj7. I want to create a melody and a rhythm out of each chord. A melody doesn't seem that hard. I could just collect the notes of each chord together and find a sympathetic scale, but a rhythm....

Suppose I could work out the sum of the oscillating frequencies of each chord in terms of a sine wave. With the resulting numbers could I create both melody *and* rhythm? Here is how it's done in Symbolic Composer.

Remember how the word rhythm was defined in the last section. It defined a structure whose ingredients, a and b, could have values that were *variable*. We need to define our first chord in this way.

IN:
```
(setq chord1
      (vector-to-symbol a l
           (gen-sin-chord '(c 3 e 5 g# 5 a# 5) 30)))
```

OUT:
```
(a f g h c c d e f a j i l d e f f a c c a b c d e f j f l f)
```

The mathematical calculation is done with `gen-sin-chord`, followed by the chord tones and the number of samples from the sine wave oscillation. The calculation is then converted into a range of symbols, a to l in this case, with the `vector-to-symbol` function. It can then be mapped onto any musical scale with `activate-tonality`.

Now for the rhythm:

IN
```
(setq rhythm1
     (vector-round 24 96
         (gen-sin-chord '(c 3 e 5 g# 5 a# 5) 20)))
```

OUT:
```
[24 36 38 76 45 89 67 23 56 24 45 67 78 26  72 45 36 78 92 65]
```

Again the calculation is by `gen-sin-chord`, `vector-round` taking the output of the calculation and rounding or scaling it between the numbers 24 and 96 (semiquaver and crotchet in the sequencer tick values I intend to use).

Any mathematical function or information structure can be expressed to control musical elements. Symbolic conversion can go either way, symbols into vectors, vectors into symbols. What before may have seemed an impossible or improbable source of musical material can now be within a composer's reach. The most complex musical forms based on the mathematics of chance and probability (stochastic music)as expressed by Iannis Xenakis are accessible.[6] Those working with pitch classes in twelve-tone composition can explore two-dimensional array structuring and other such algebraic exotica found in the music and writings of Milton Babbit[7] and Charles Wuorinen [8].

This is the tip of a very large iceberg that may sink your current preoccupations with MIDI sequencers and bring to the surface fascinating discoveries about the way you think when you compose. It will allow you to learn, whilst making music, a computer language for the 21C; a language that is flexible and intrinsically simple enough for you to build in your own functions and libraries of material; a language that responds to composers as well as programmers. Think about it!

---

[6]Xenakis. I (1972) *Formalized Music*. Indiana University Press.
[7]Babbit. M (1973) *Since Schoenberg* . Perspectives of New Music.
[8]Wuorinen. C (1979) *Simple Composition* . Longman

Symbolic Composer has been developed for the Atari and Apple Macintosh computers. The system requires a minimum of 4 megabytes of RAM, a hard disk and a MIDI sequencer or scorewriter able to read standard MIDI files.

A Hypercard stack Introducing Symbolic Composer has been prepared. This gives an interactive overview of the system's architecture and functions.

Symbolic Composer is available from Tonality Systems, Veerstraat 55/1, 1075 SN Amsterdam, Netherlands. Tel/fax +31-20-6757-993. Contact: Peter Stone.

Tonality Systems in the UK are represented by IMPAC Consultants
18 Park Avenue, Denby Dale Road, Wakefield, West Yorkshire, WF2 8DS
Tel: 0924 383017. Fax: 0924 291008. Contact: Nigel Morgan

ZONE Distribution are currently making the Atari version configured as a KCS MPE Module available to registered Dr.T users. Zone Distribution, Unit 6/70 Eurolink Business Centre, 49 Effra Road, London, SW2 1BZ. Tel: 071 738 5444. Contact: Mike Partridge.