# Design Issues in Musical Composition Networks

**Peter Mcilwain**
Centre for Electronic Media Art,
School of Music,
Monash University
VIC 3800
Australia
Peter.Mcilwain@arts.monash.edu.au

**Jon McCormack**
Centre for Electronic Media Art,
School of Computer Science and Software Engineering
Monash University
VIC 3800
Australia
Jon.McCormack@infotech.monash.edu.au

## Abstract

*Alternative approaches are explored and discussed relating to the design and function of the software project "Nodal". This project aims to create a graphical environment that enables the user to configure a spatial, directed graph that generates music in real-time. We refer to such graphs as composition or nodal networks. These networks provide a programming and visualization structure that emphasizes temporal relations and event topology – the primary reasons for using networks to represent musical processes. The discussion in this paper relates to the initial design constraints that have been imposed on the project and alternatives within these constraints that give rise to different musical behaviors. Assessments are made about the effectiveness of different design approaches.*

## Introduction

Computers and algorithmic processes bring new possibilities for both compositional notation and the process of music generation. This paper describes some in-progress exploration of how we can design dynamic, graphic notation systems suitable for composing and specifying generative algorithmic processes for music composition. Conventional representation systems are ill suited to effectively notating the musical output of non-linear generative processes. Moreover, they are inadequate for programming specification of such processes. The scheme examined in this paper represents an on-going investigation into how visual programming and notation systems can be designed for generative musical composition.

*Nodal networks* are a new musical representation and generation system we have developed for computer-assisted music composition and generation. Dynamic musical systems are represented as discrete and continuous events spatially organised as a directed graph. The graph (termed a nodal network) is built interactively by the user of the system, with *nodes* (vertices) representing discrete events such as playing a note, and directed *arcs* (edges) representing continuous events such as dynamic expression. Networks are traversed in real-time by any number of software *players* who interpret the graph layout and translate it into music.

There are a number of similarities in this approach to previous generative musical specification systems, such as generative grammars, finite state automata, Petri nets and Predicate Networks (Chemillier 1992; Lyon 1995; McCormack 1996; Pope 1986). The nodal network differs principally in the mapping of geometric space to represent timing information. The arc distance between connected nodes corresponds to the time between discrete events – i.e. the longer the arc the greater the time taken for the player to travel between connected nodes. By quantising arc distances to standard musical metres, arc lengths can represent the note durations found in conventional musical notation. However, this restriction can easily be relaxed, leading to more unconventional temporal relationships.

While it is common for networks to be used in real-time composition, a precursor to the work described here is the *Snet* program (Mcilwain & Pietsch 1996), which utilized a real-time

neural network simulation. In this system, a key aspect is the evolution connections by emulating the *action potential* of a neuron.

Graph based notation and programming systems offer a number of advantages over other methods. Musical temporal structure and phrasing is given a two- or three-dimensional geometric analogue, highlighting structural and timing information that is difficult to discern in other representations. For example, linear time representation schemes, such as the "piano-roll" system currently used in most commercial music composition software cannot obviously represent cyclic, hierarchical, or feedback structures in a composition. State machines, graph grammars and Petri nets are more suited to this problem, but while they may show compositional structure and flow, this is usually abstract and is done at the expense of timing information (Pope 1986).

Dynamic graphs are capable of generating complex, emergent properties, even through relatively simple feedback mechanisms and structures. The use of real time structural change and interaction with the compositional structure gives the composer the ability to interactively work in tandem with the generative complexity of computational networks, potentially leading to new creative possibilities in composition.

The implementation of composition networks has been realised in the software project *Nodal*, which permits real time construction, editing and traversal of networks. The remainder of this paper relates to how the design process of the software influenced the development and analysis of how composition networks can be implemented as a practical tool for the musician.

## Initial Concept and Design Constraints

The initial concept behind *Nodal* was to create a method of generating music via a network in which the output of a node is a musical note (defined by MIDI note parameters). Time between note events is determined, and represented, in a graphical form by the length of arcs between the nodes. The design brief was that the software should generate music in real-time so that it could function as a module or plug-in with other software such as MIDI sequencers and programs such as Max/MSP.

Given that the number of possible configurations and functions that a composition network can have, software of this kind can potentially be too complex for a user to configure in a meaningful way. For this reason the over-riding design constraint was that the software be as simple and as intuitive to use as possible. Therefore a significant proportion of the software development process was given over to addressing user interface issues. This design constraint also determined that the software would present of a limited set of possibilities while still providing a flexible and multifaceted network environment. This raises the question of how to select design features that fall within the design constraint. Part of the process that was adopted was to look at how an existing piece of music might be created and represented as a real-time nodal network. This should then provide clues as to what features and designs may be the most useful.

The nursery rhyme *Three Blind Mice* was selected for its apparent simplicity. This selection was made with the rationale that if a nodal network was not able to generate a "simple" melody then it would be hardly likely to generated anything more sophisticated.

## Nodal Representations of a Melody

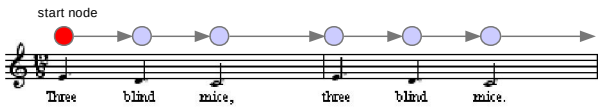



***Figure 5.*** *A nodal network with a linear structure.*

One way to represent a melody in a nodal network is to represent every note event by a singular node. In this kind of network, connections only need to be unidirectional (the first node connects to the second etc.). This results in a linear network that plays the melody once (shown in Figure 1). Clearly, this configuration is not a particularly useful or sophisticated use of a graph (in fact it is a list) because it fails to utilise the more complex structural properties that are a key feature of graphs in general. Such features enable the user to work with representations of underlying iterative, hierarchical and feedback structures that afford the possibility of creating variants from a finite set of elements. Furthermore it is time consuming and cumbersome to create melodies in this fashion.

Given that many melodies are repeated, the network described above could be configured in a loop so that the last node is connected to the first. The network would then play continuous repeats of the melody. Large-scale repetition of this kind is the lowest level of structure that can be represented in a network.

Another way to produce a continuous loop is to make all of the connections in a linear layout, bidirectional (i.e. the first node is connected with the second, which is also connected to the first, and so on). This will produce a variant of the melody that alternates between the original form and a kind of retrograde. Here the retrograde version is a literal rhythmic retrograde and a displaced pitch retrograde (see Figure 2).

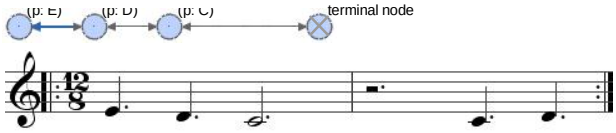



***Figure 6.*** *2 way connections in a linear network.*

## Nested Looping Structures

Melodies such as *Three Blind Mice* include motifs that are repeated. This is particularly true of rhythmic motifs (a full transcription with a motivic analysis is shown in Figure 3). Therefore, the next level of structure that can be represented is a series of nested network loops.

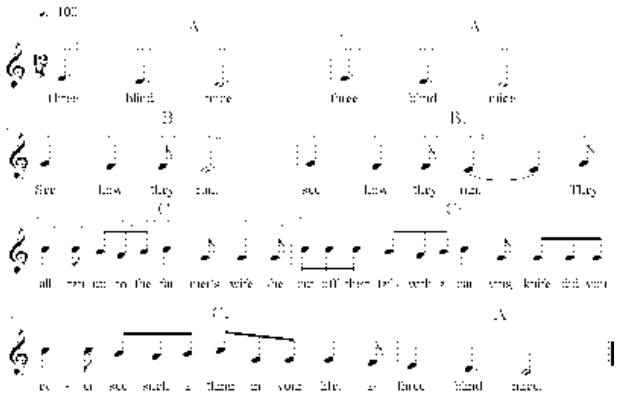


***Figure 7.*** *Transcription of Three Blind Mice with motivic analysis.*

Figure 4 illustrates a network that generates a simplified version of *Three Blind Mice*. It consists of three interconnected looped structures (sub-networks) that are part of a larger looped structure. In order for the sub-networks to be connected it is necessary for each sub-network to have one node with multiple outputs. In Figure 4, nodes that have multiple outputs are: A-3, B-4 and C-8.

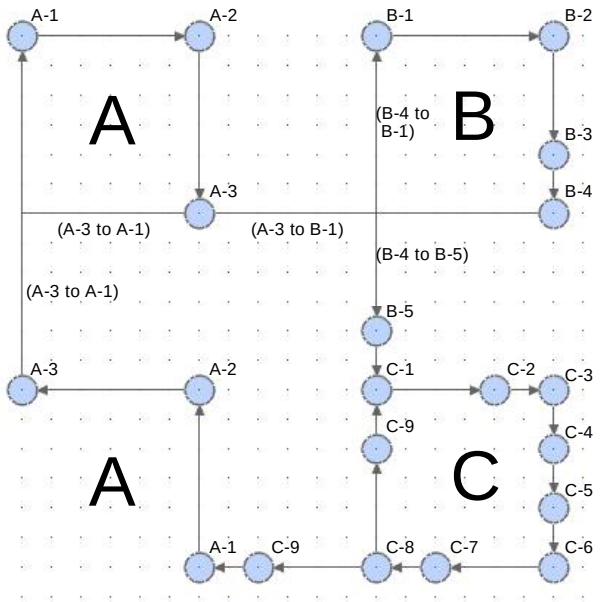



***Figure 8.*** *A nested looping structure.*

Whenever a node has multiple output connections some sort of rule must be employed to determine which of the connections will be activated[1]. The rule that is relevant in the example in Figure 4 is that the two outputs are selected alternatively. For example, on the first firing of B-4 the connection will be B-4 to B-1 and on the second firing the connection will be B-4 to B-5[2]. In the melody, motif C is heard three times. To make this possible in the network an extension to the connection rule must be applied. This is done by creating a connection list that is applied sequentially. So for node C-8 the connection list will be:

1. C-8 to C-9
2. C-8 to C-9
3. C-8 to C-10

## Multi-layered Structures

While the example in the previous section does contain some iteration the network does not display any sophisticated behavior as a result. That is, the structure of the network does not represent the evolution of the elements of motif A. Instead the three motifs must be created separately and as a result, any short-term elaboration of the melody cannot be represented. This is the case with motif C, where there are in fact three variants of the motif but only one is represented in the network shown in Figure 4.

A more sophisticated way of representing *Three Blind Mice* is to create a multi-layered network that adds information to a network rather than replacing it with a new sub-network. Essentially, this preserves the topology of the original network, but makes the spatial layout and relationships easier for the composer to understand in terms of melodic structure and progression. In order to do this it is necessary to find any elements of the melody that are retained from one motif to the next.

The rhythmic structure of *Three Blind Mice* does contain repeated elements that are elaborated as the melody progresses. As shown in Figure 5 below, the rhythmic motif in A continues through all 8 measures of the melody. Additional time points are added with successive motifs and are retained until the end of measure 7 after which there is a recapitulation of the opening motif. The exceptions to this are the time points added in measures 6 and 7. These additions only occur for one measure each and can be seen as elaborations that are necessary for text setting purposes. The analysis shows a distinct evolving pattern stemming from the initial rhythmic motif.

[1] In this discussion we assume only one input per node.

2 This rule is also applied in the example in Figure 2 where the second and third nodes in the network both have two outputs.

In rhythmic terms, the melody can be represented as a series of networks that run simultaneously (although starting and ending at different times). While rhythmic structures are retained, pitch material associated with particular time points change. In order to deal with this, a list of pitches for each node can be used. Here the list is read sequentially, the position in the list being advanced each time the associated node fires. The subnets discussed above and their related pitch lists are shown in Figure 6.

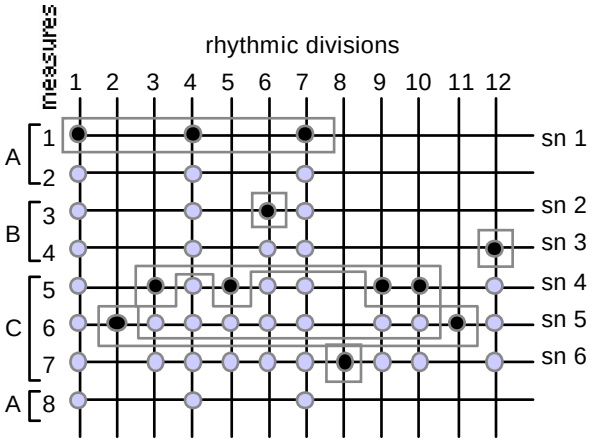



***Figure 9.*** *Overlaid rhythmic structures in Three Blind Mice. Black dots represent the addition of a new time point. Subnets derived from the analysis are enclosed in grey boxes and are annotated "sn".*

As the subnets start and stop at different times some method must be used for determining when a node is to be active. A simple convention for turning subnets on is to have an *activation threshold*. Here a node only becomes active after a certain number of messages have been passed to it. An easy way to include this is to include a non-active state in the pitch list. For example the pitch list for subnet 2 would be:

n1: Nx2, F3x2, B3x3, N

Here N = non-active state.

Another way to conceive of the subnet layer approach is to think of the layers as part of a three dimensional space. Here arcs between nodes in different layers could be used to activate the successive layers. This requires a spawning process that creates multiple activations within the network. For this to occur a special connection rule must be used where two or more connections from one node can be activated simultaneously. With multiple activations it is therefore possible to create polyphonic musical textures and, given the right configuration, complex feedback interactions between subnetworks.

The pitch list technique discussed previously represents an extension to the concept of the pitch series. The classical pitch series consists of an array of pitches that are applied to time points in a serial fashion. The technique described above has unique arrays for each time point. This has interesting, and potentially, very useful possibilities for the representation of vertical pitch structures in horizontal environments such as a nodal network (although a nodal network is not needed to implement this technique). For example the pitch lists shown in Figure 6 show chords that are voiced across time, an example of which is the pitch list for 1.1 (subnet 1, node 1). This pitch list constitutes the C Maj chord. Other examples include 1.2 - Dom7th and 3.1 Dom7th. The pitch list technique enables the exploration of horizontal harmonic structures. For example, if the chords in the pitch lists for *Three Blind Mice* where changed to those from a different mode, then the melody would be transformed with a new harmonic structure.

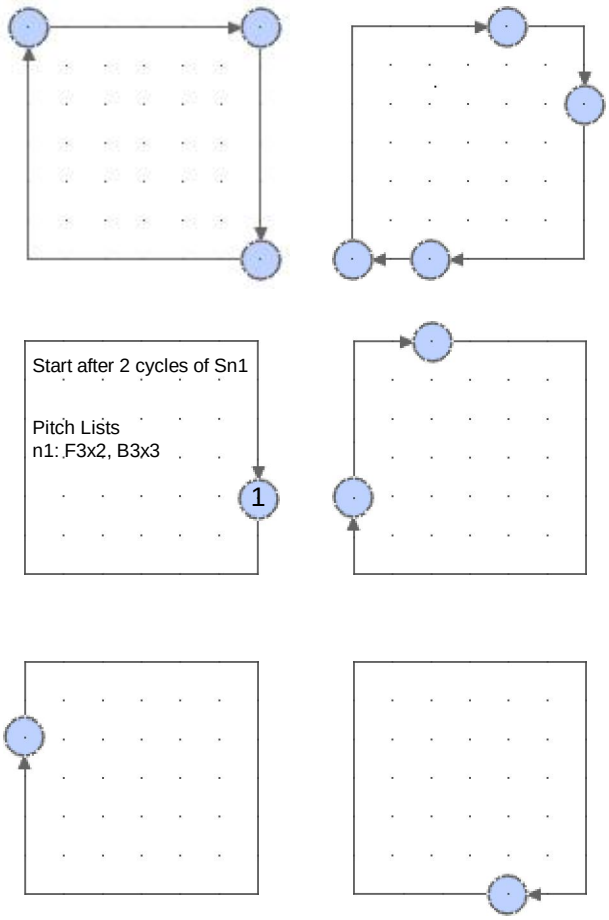



***Figure 6.*** *Subnets and related pitch lists for Three Blind Mice.*

In the rhythmic domain, the multilayered approach enables subtle yet complex evolution of rhythmic structures and has advantages over a two-dimensional structure in that the rhythmic structures can be maintained across time. Like the potential for harmonic transformation via pitch lists, base level transformation of the rhythmic structure is relatively easy to perform. As the layer approach enables the representation

of different hierarchies of structure, transformation can in fact occur an all of these levels.

This would suggest that it is desirable to create a three-dimensional environment within which to work. There may however, be a trade-off between the flexibility that a three-dimensional environment offers against the complexity of working within it.

## Emergent Music

So far we have discussed nodal networks from the perspective of taking an existing piece of music and representing it in a nodal network structure. This approach is useful in that it brings to light structures that may be musically useful. It can be argued however that this is not the best approach to take because real-time nodal networks may generate music that does not necessarily conform to the kind of structures found in music such as the *Three Blind Mice* example. They may however be of interest for other reasons.

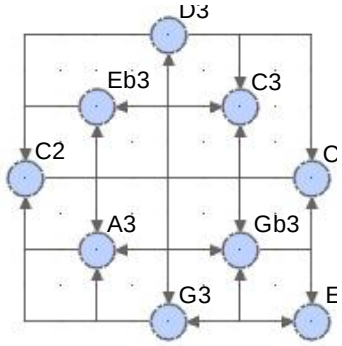
One obvious approach is to use networks in an experimental fashion. Here the network is configured without a fixed goal in mind. Instead the focus is on emergent properties of a particular network structure. It can be argued that this approach makes the best use of the structural properties of networks. The nature of real time interaction also facilitates easy experimentation and play with a complex system.

It is possible for example, to configure networks that can create musical textures that would be very difficult to calculate manually. Furthermore they can create textures with ongoing change based on a defined set of elements. The network shown in Figure 7 generates a melody that contains many motivic fragments but does not repeat for a least 80 measures. This network has a simple two-dimensional design, comprising nine nodes, each of which outputs a fixed pitch. Connection between the nodes are however relatively complex; there is an inner core of four nodes that are all interconnected and an outer layer of five nodes that are more sparsely connected. This design focuses activity on the inner core allowing for variation coming from the outer layer.

The connect rule is also simple, for every input the output connection is selected sequentially (where there is two or more outputs). While there is motivic diversity in the melody the main variation is through recombination from available pitches and durations. Therefore long term changes in register, harmonic and rhythmic structures are not present.

A more sophisticated design would allow the interaction between the nodes to influence parameters such as pitch and duration. This can be achieved via the use of relative parameters in the nodes. For example a node can be designed that has a pitch increment which outputs a pitch value either higher or lower than the value in the node that caused it to fire. This approach can be taken for all parameters that define a note. Relative values can therefore create change over time and also allow for true emergent behavior to evolve.

*a.*




*b.*

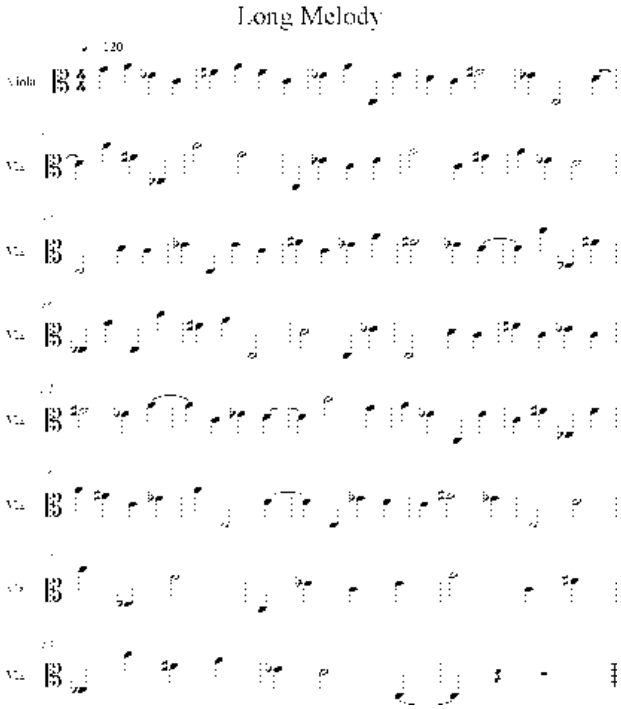



***Figure 7.*** *a) Long melody network diagram, b) transcription of the output.*

## The Nodal Software

The *Nodal* software system, currently in development, implements many of the design features discussed in this paper. Figure 8 shows a screen shot of the software in operation. Networks are created in real time and displayed in the main window. Multiple networks and players are possible, with each player starting from a possibly different node. Each node is assigned a set of *attributes* that specify player state changes car-

ried by that node (these are a generalisation of the parameters discussed in previous sections). In general, attributes are either *setting* or *modifying*. Setting attributes set a property at an absolute value. Modifying attributes modify the current value carried by the player. Common attributes include pitch, duration, volume, and note-on velocity. The system is extensible so custom attributes can easily be added to the system.

Attributes can be edited on a per-selected-node basis in specific attribute editors (right window). These editors can also be set from a MIDI input device. The software design also permits arcs to modify continuous information such as volume or expression. Attribute editors for arcs are currently under development.

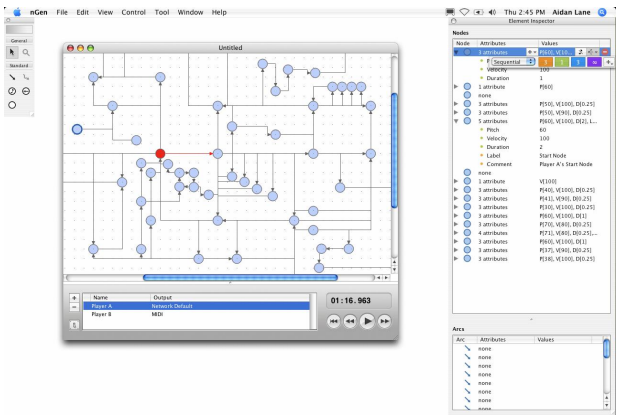


***Figure 10.*** *Screen shot of Nodal in operation.*

Outgoing arc order and sequencing can also be set in the right window. A novel colour coded arc manipulation system allows the user to change arc sequencing intuitively, accommodating arc insertion and deletion with minimal modification of existing ordering. This system allows the user to set output arc order and enumeration for a given node. Further controls allow switching between random and sequential ordering where individual arc repeat counts become probability weights.

## Conclusions

The design of *Nodal* is still evolving. In this paper we have looked at how a number of design issues and constraints can be developed using analysis of seemingly simple musical structures. Developing software with complex or unconventional user-interfaces can be a time and resource intensive task, so any design and operability issues that can be established before the software is built will be beneficial. However, in many instances these issues can only be determined by actually building and using the software, since no analogue exists in conventional systems. Intuition and heuristics play a strong role here.

Nodal networks appear to offer interesting new possibilities for the composer. It remains an on-going investigation as to how we can best use and design them in software to realize their full potential.

## Acknowledgements

The project team for *Nodal* consisted of the authors, Alan Dorin and Aidan Lane. Aidan also wrote the current version of the software. This research was supported by an Arts/IT grant from Monash University.

## References

Chemillier, M. 1992, Automata and Music, in Strange, A. (ed) *Proceedings of the 1992 International Computer Music Conference*, ICMA, San Francisco. pp. 370-371.

Lyon, D. 1995, 'Using Stochastic Petri Nets for Real-Time *N*th-Order Stochastic Composition', *Computer Music Journal*, **19**(4), pp. 13-22.

McCormack, J. 1996, 'Grammar-Based Music Composition', in Stocker, R., H. Jelinek, B. Durnota & T. Bossomaier (eds), *Complex Systems 96: From Local Interactions to Global Phenomena*, ISO Press, Amsterdam. pp. 321-336.

McIlwain, P.A., A. Pietsch (1996) 'Spatio-temporal Patterning in Computer Generated Music: A Nodal Network Approach', in *Proceedings of the International Computer Music Association Conference 1996,* ICMA, San Francisco. pp. 312-15.

Pope, S. 1986, 'Music Notations and the Representation of Musical Structure and Knowledge', *Perspectives of New Music*, **24**(2), pp. 156-189.