Return to Index

FrameWorks : a structural composition tool

Richard Polfreman
Music Department
Faculty of Engineering and Information Sciences,
University of Hertfordshire, College Lane, Hatfield, Herts. AL10 9AB.
r.p.polfreman@herts.ac.uk

Abstract

FrameWorks is a new software tool that has been developed as a proof of concept for novel ideas relating to user-interface approaches for music composition systems. FrameWorks has been developed on the basis of HCI (Human Computer Interaction) research investigating the processes of music composition and aims to support more top-down approaches to the task which are poorly supported by extant systems. This paper briefly covers the background to the project, describes the main features of the FrameWorks concept, presents the current user-interface and discusses future developments of the system. Some simple musical examples are given to illustrate the system's potential.

Keywords: User-Interface Design, Music Composition Systems, Midi Sequencers, Java Applications

1 Introduction

FrameWorks has been developed as a proof of concept for emerging ideas relating to handling musical structure effectively within a software composition environment. These ideas have been derived from Task Analysis work studying the process of music composition through observations, interviews, retrospective protocols and questionnaires involving a variety of composers working on compositions for acoustic, electronic and mixed media. The resulting Generic Task Model has been used in the development of Modalys-ER (Polfreman, 1999), a graphical physical modelling synthesis program in collaboration with IRCAM, and more recently in the development of FrameWorks (Polfreman and Loomes, 2001). While the GTM is not yet fully complete and further research is necessary to further formalise the description and increase confidence in its generality, we found the model particularly useful in setting out the structures and concepts to be used in interactive software development.

FrameWorks is a small composition environment based on three main levels: Workbench, Framework and Sequence. The Workbench is an area where the composer can organise musical and non-musical materials that are important to them for a particular composition, e.g. text, diagrams, and musical fragments. The Framework is where an individual composition is constructed using *components* that can be connected via *relations*. Components are flexible containers of musical material, while relations indicate dynamically maintained musical transformations. The Sequence level displays the resulting material from the framework without indicating directly any the structural elements, similar to standard sequencing software. FrameWorks in its current form uses MIDI, while the central concept is generic and may be applied to any time based (musical) information, e.g. raw sound, synthesis control signals, DSP parameters, etc.

A preview release, FrameWorks_1.0pr, has been developed using Java, is freely available (from April 2001), runs on Mac OS, Windows and Linux in the near future. All descriptions of the software here relate to this version.

2 Software Aims

In developing FrameWorks we had some key aims in terms of practical concerns and user features to be provided.

2.1 Cross-platform

Given the uncertainty of operating systems and their futures, in addition to the disparate use of computer platforms by composers (Mac OS and Windows are typical, but Linux, IRIX and other Unix-like systems are also used for music applications), we wanted to develop a cross platform implementation that would not be OS dependent. Java provides

such an environment and is particularly useful for Graphical User Interface development since this part tends to be very OS specific with other development tools. Unfortunately at the start of development there was no effective sound support for Java and so we adopted Grame's Midishare which is freely available for Mac OS, Windows and Linux. Midishare provides a Java interface enabling Java programs to access the Midi system. While this limits the cross-platform nature of the system somewhat, it still allowed us to develop a multi-platform application. Since that time a Java Sound API has been developed, as part of Java 2, which provides some Midi and audio support within Java.

2.2 'Viscosity', 'Premature Commitment'

These are two of Green's 'Cognitive Dimensions of Notations' (Green, 1989) which are used in assessing the usability of notation systems. Viscosity refers to the resistance to change in a notation, i.e. how easily can local changes be made and at what cost. Premature commitment refers to when users have to make decisions too early. Low viscosity helps this since early decisions can be changed more easily later on. There are other cognitive dimensions but we have focussed on these two with Frameworks.

In a recent paper (Blackwell, Green and Nunn, 2000), Green and others examined some musical systems and noted that viscosity and premature commitment were low in software sequencers. Viscosity was low since it is very easy to make a local change, e.g. by dragging or inserting notes. Premature commitment was low since work could begin at any point in the piece and be carried out in any order. However, we argue that this doesn't take into properly into account the nature of musical works and looks purely at the notation itself and not at its application.

In many musical works there is an idea of 'theme' of some sort – a musical phrase or pattern that occurs several times in different places either identically or transformed in some way (such as a transposed version). There are also dependencies across parallel events such that chords, melodies and bass parts for example have to maintain some musical relationship that the composer has in mind. Thus, changes made locally to an event or series of events can require many other changes to made in order to preserve the integrity of the composition and so there can be a high cost involved and therefore a high level of viscosity.

Premature commitment can also be seen as high since if a composer is working on a thematic piece, they have to commit to a theme early on in order to avoid the high cost of changes to the thematic material. There is also limited support for defining higher level music structures and so typically the composer must commit to some material and develop from that material rather than being able to set out some musical structure and then begin to work on the underlying material.

FrameWorks begins to solve some of these problems, although in its current form in only limited ways. In future developments we hope to tackle these issues more effectively.

2.3 Algorithms, Maths and Programming

One solution to the above problems is to use an algorithmic composition approach. This involves defining a composition in terms of mathematical processes, usually involving some form of programming environment. An example system is IRCAM's Open Music (Assayag, 1995), which is a visual programming environment based on Lisp with built in libraries for handling musical material and manipulation. The idea here is that musical events can be generated and/or manipulated using mathematical tools and so complex relationships can be set up and maintained with changes possible at many different levels within a piece. While this is a very powerful approach it requires the composer to learn sophisticated mathematical and programming ideas in order to realise their musical ideas. As such these systems perhaps only cater for a limited number of composers. With FrameWorks we aim to provide methods of expressing and maintaining musical relationships, but without recourse to mathematical expressions and hope to deliver a system that is more usable by a wide range of composers without the technical expertise of computer programmers.

2.4 Products of Research

Our task analysis work indicated that research was an important part of the composition process and that this included many topics both within music related fields and in others. While developing a system that supported all kinds of research within it would be difficult (some kind of www search and collation system would be possible, but this is

better supported by existing web browsers), we wanted to provide in FrameWorks some area where the products of a research phase could be integrated into the composition environment. So while such materials may not be musical or incorporated directly into to a composition, they could be presented with and saved along side composition documents. One key element we wished to support was the use of informal sketches/diagrams which composers often seem to use.

3 FrameWorks

3.1 The Workbench

The Workbench is designed as an area where musical and extra-musical information can be placed and saved along with a composition. Here the composer can experiment with ideas, make notes, sketches etc. Musical materials developed here can be transferred to the Framework for use in an actual composition. Currently supported element types are shown in Table 1, together with which basic operations can be carried out on them.

	Create / Edit	Export	Import
Text	•	X	X
Picture	X	X	from .gif files
Diagram	•	X	X
Component	•	to framework	from framework
Relation	•	to framework	from framework

Table 1: Workbench elements.

An element is represented on the workbench by a rectangular object with an expander button which reveals/hides a drop-down preview of the element's contents. A double-click opens a new window displaying the element contents fully and for editing where supported. Figure 1 shows a diagram element with its preview (diagrams can be in colour in FrameWorks).

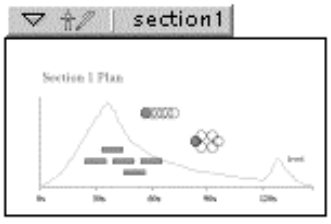

Figure 1: A diagram element (expanded).

Figure 2 shows an example workbench with relevant items labelled. At the top are mode selection buttons for switching between the workbench, framework and sequence. The central area is the workbench where various elements are placed. At the bottom are stores buttons that provide quick access to files of different types stored with the workbench.

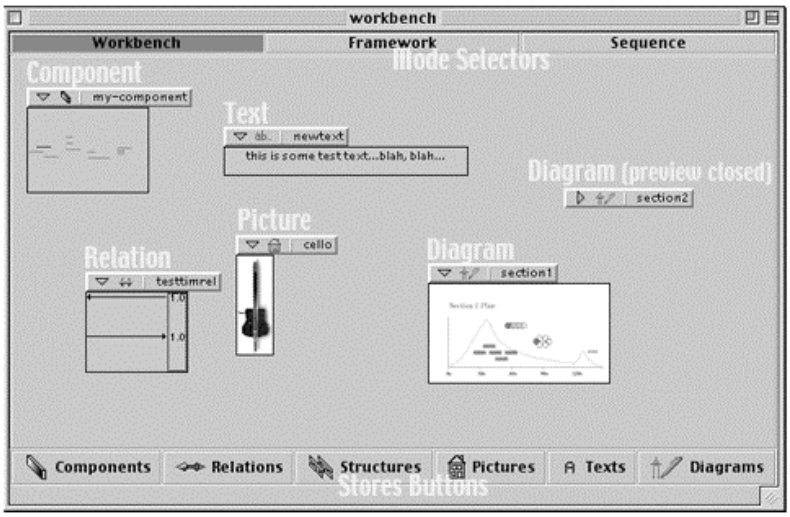

Figure 2: An example workbench.

3.2 The Framework

The framework is where a composition as actually put together. It uses a track based system as in traditional sequencers but the organisation and manipulation of material in these tracks is novel. Frameworks has no concept of bars or measures. Tracks represent (real) time lines onto which the composer places components. These are containers for musical material and are represented as parallelogram shaped objects that hang onto a track (the effect is a bit like a washing-line hanging clothes). The components can be dragged continuously to any point on a track and also dragged between tracks. They can be resized to arbitrary duration, which has the effect of stretching and compressing the musical material within and thus changing the tempo of this material. There is no global or track level concept of tempo in Frameworks (a notional 120 bpm tempo is present) and all timings are listed in seconds or milliseconds. In future a global time scaling feature should be possible in order to vary the playback speed of the whole framework.

The components can be opened to edit the musical events they contain. Component editors currently use a piano-roll style notation (with event shading indicating MIDI velocity), since a proportional notation best reflects the flexible duration of components. For example, events have no fixed duration, it is proportional to the duration of the whole component. In this way the display of a component's contents is invariant with respect to changes to the component duration. While the component editor is basic, it does have some useful features, particularly for creating complex rhythmic patterns. The editor can display a time grid from 2 to 64 divisions and events can be snapped to this grid (or quantized after insertion). By changing the grid size for different groups of events it is simple to create rhythms of say, 13 against 7 against 4, for example. Figure 3 shows a component editor.

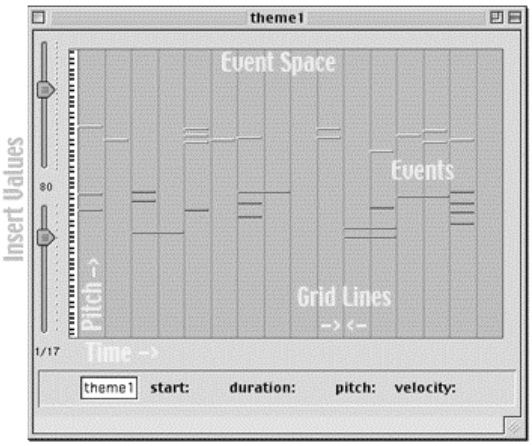

Figure 3: A component editor.

By placing components on the framework a composer can build up the structure of a composition without necessarily having written any musical material. Also, since components can be resized, the duration of specific material can be changed very easily. Both of these properties aid in reducing premature commitment as described earlier.

While components in themselves allow some structure to be defined, one of the most important elements in music composition are the relationships between materials occurring in different places, parts and/or times. In the framework this is made possible through the use of relations. A component can be connected to another component via a relation which establishes a dynamically maintained link between the material in the source component and that of the target. The most basic relation is the *identity*. With this relation the target component always contains a copy of the material in the source. Even during editing this relationship is maintained such that changing events in the source immediately changes events in the target. More interesting relationships can be established using *time*, *value*, *filter* and *multi* relations. Time relations contain an arbitrary number of *time maps*, each of which takes a time slice of the source material (start and end points expressed as percentages) and adds it to the target material either forwards or reversed and with a time scale factor. A simple graphical editor is used to construct these relations. Value relations implement time varying transformations on the values of the material from the source component before placing the result in the target component. Currently these can affect pitch and velocity values and can apply shift, scale or inversion transformations. The time dependent behaviour is expressed through an editable breakpoint envelope (a value relation editor is shown in Figure 4). Filter relations selectively pass material from the source to the target. This is according to pitch, velocity and duration ranges simultaneously. The values to pass are specified using graphical range sliders. Multi relations allow an arbitrary number of other relations to be applied successively within a single relation.

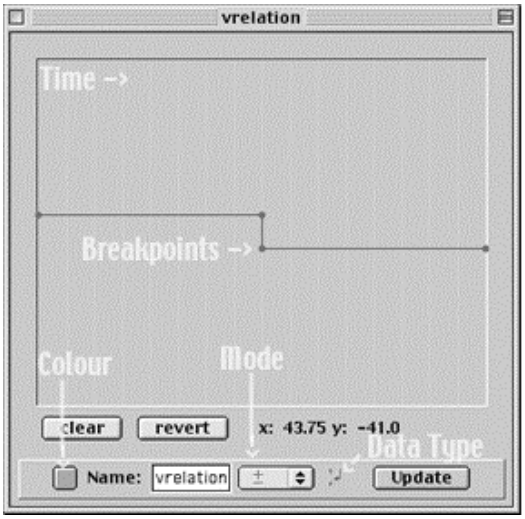

Figure 4: A value relation editor.

A single component may only have one source but many targets, while chains of components can be made by connecting one to another and on to others using various relations. Disconnecting a component from a source will leave its material in place, but it will no longer be dependent upon the source material. Since the framework is dynamically maintained such that any changes made to material, components or relations are instantaneously reflected throughout the structure, this produces a system with reduced viscosity as defined earlier. In fact it encourages the composer to experiment with different musical ideas since the cost of changes is so low – one can imagine a composition completely defined by one source component and connected dependent components where changing the source can very quickly give quite different musical results (as in some of the examples given later).

Although currently a single level framework, with only some basic relation types, this already offers composers much that is difficult to achieve within typical music applications. In future developments we hope to extend the framework system to handle hierarchical organisation and perhaps multiple overlayed structures to improve the scalability of the system. We also wish to extend the type of relations supported, in particular by adding event-based transformations. An interesting feature of compositions produced using the system is that the framework document itself contains a

useful analysis of the work since important relationships are explicitly set out in the component structure. These documents could then be studied from a musicological point of view, where no doubt other relationships and musical ideas could be elucidated. Figure 5 shows a framework under construction.

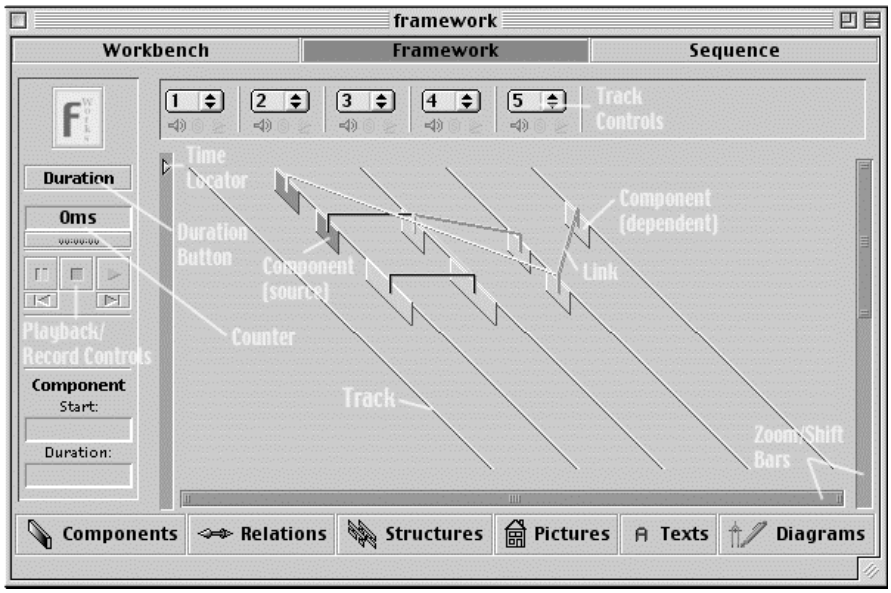

Figure 5: An example framework.

3.3 The Sequence

The sequence level allows the composer to view the resulting composition without explicitly indicating the structure in terms of components and relations. As such it shows a flattened version of the piece similar to a typical MIDI sequencer. Again this uses a kind of piano-roll style notation to indicate the musical material. Currently the sequence display is very basic and needs some enhancement to give improved notation of the piece. While some editing features remain, the sequence level does not allow the direct editing of the musical material, since this is dependent upon the framework. A possible future addition would be to include the ability to disconnect the sequence from the framework and allow such editing, although subsequent re-connecting to the framework would destroy any changes made to the sequence. The current alternative to this is to export the composition as a standard MIDI file and edit the piece in a MIDI sequencer, since event level editing is generally well supported in such programs. Also in the preview release there is only support for MIDI note events, so another program is necessary to add MIDI controller data for example. Figure 6 shows the sequence level corresponding to the framework shown in Figure 5.

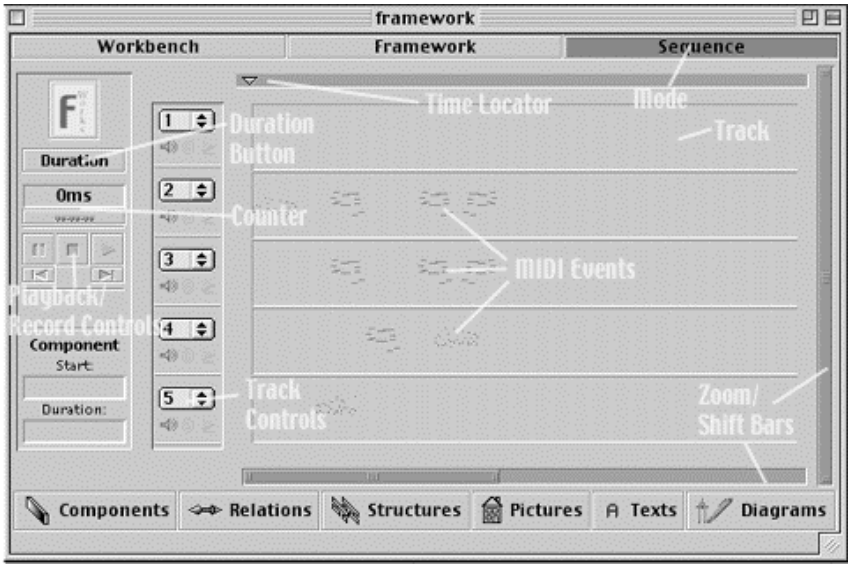

Figure 6: An example sequence.

4 Musical Examples

In this section we demonstrate, using simple examples, some of the ways in which the framework area of the software can be effective in exploring musical ideas.

4.1 A Basic Framework

This first example examines a very simple framework using just three components and two relations. Figure 7 shows the framework with annotations to indicate the precise relationships between the components. All that happens here is a simple phrase in the source component is extended by an octave transposition down and this transposed version is then time reversed. Two tracks are used to clarify the organisation of the components, but they are simply arranged successively in time.

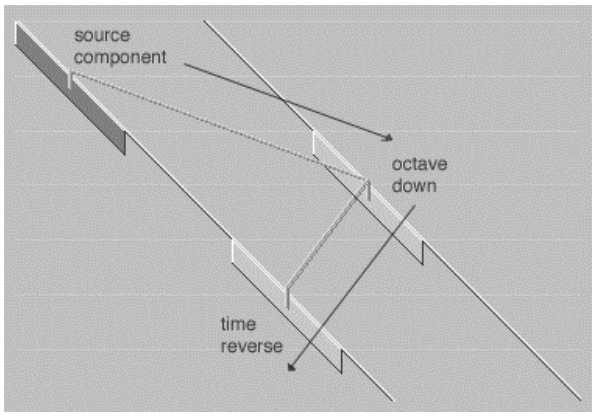

Figure 7: A simple framework

The components can be placed anywhere in the framework without affecting the relationships between the material within them. For example, the source component does not have to occur at the start but could equally well be placed at the end or in the middle. Figure 8 shows the contents of the components, the source and the two dependent ones.

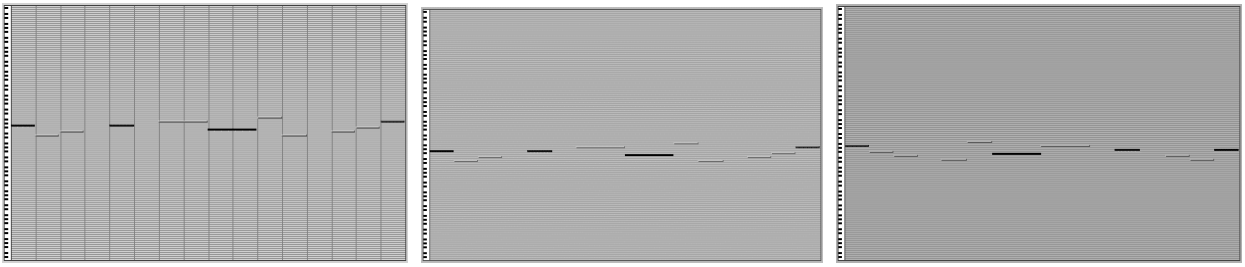

Figure 8: The components: (from left) source, octave down, (octave down and) time reversed.

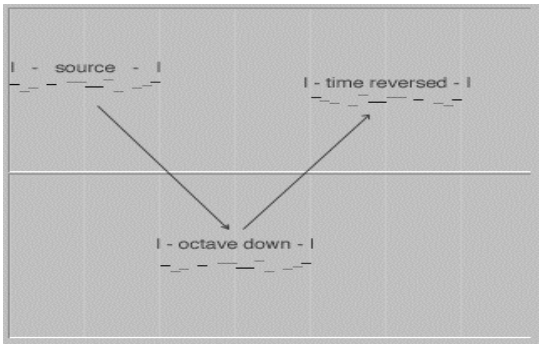

Figure 9: The sequence (annotated).

The resulting sequence is shown in Figure 9. While this framework is very basic, it illustrates the connectedness of material within different components and by simply changing either the material, the relations or the location of components the musical results can be altered very easily. In the next section we explore the effect of changing material in a more complex (but still relatively simple) framework.

4.2 Frameworks with Alternative Content

In this example we construct a framework for the possible opening of a musical work and then explore some possibilities of using different source material. Again a single component is used as the total source material and everything else is derived from it using simple relations. Here we use eight different relations: identity; time reversal; double and 4/3 speed; fifth up, fifth down and octave down transpositions; inversion about E3. Figure 10 shows the framework annotated to show the details of the relations. We then fill this framework with different types of material by simply editing the source component. Since all other components are linked in some way to the initial source, changing this source material affects the entire piece. In the examples, the first phrase uses a division of 12, the second a division of 16 and the third a division of 8. Only the content of the source component changes, the framework itself remains unaltered. While one can hear the same structure being implemented in the results, the musical effect is quite different in many ways.

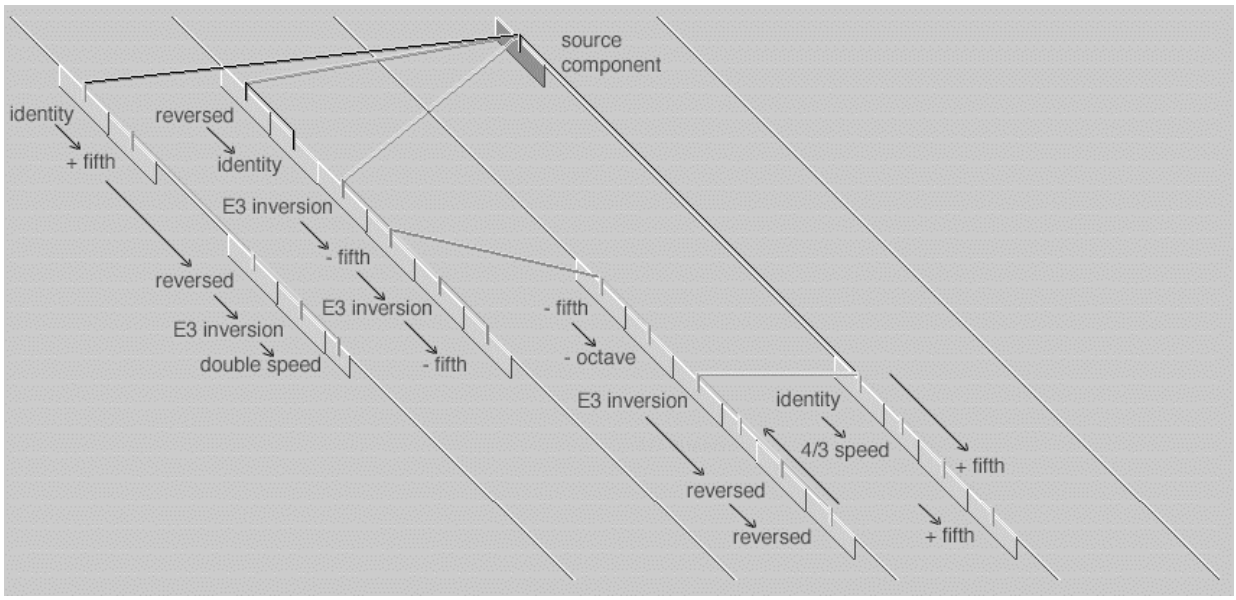

Figure 10: An opening framework.

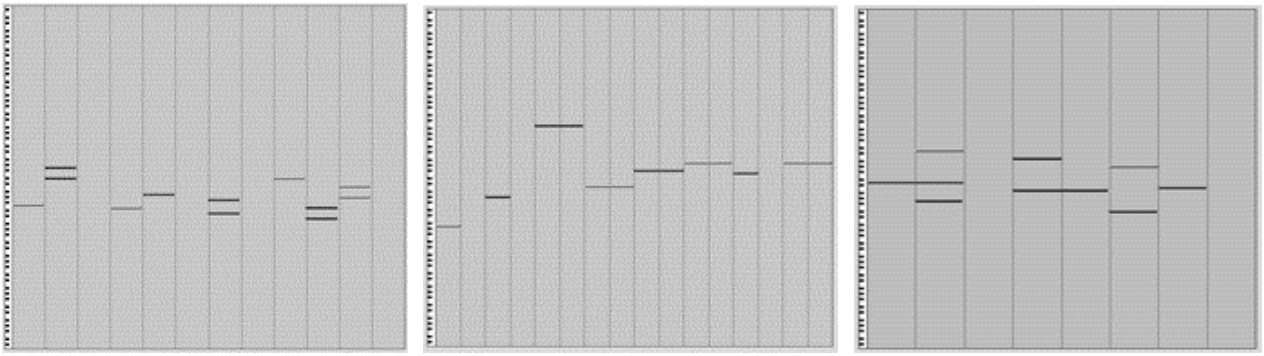

Figure 11: Three alternative source components (in 12, 16 and 8 divisions).

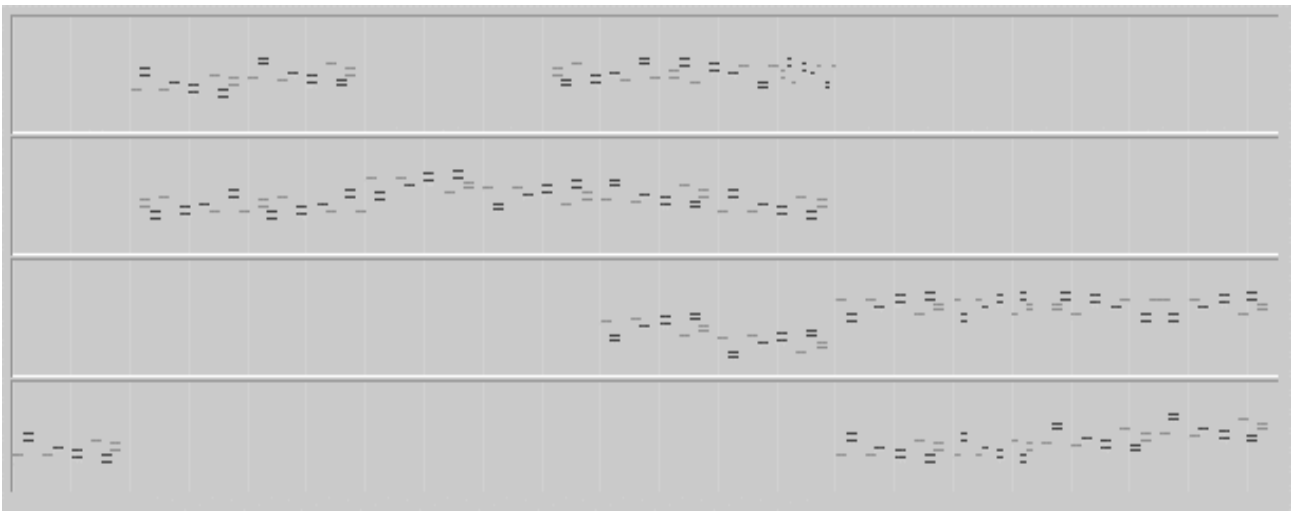

Figure 12: Sequence from source 1.

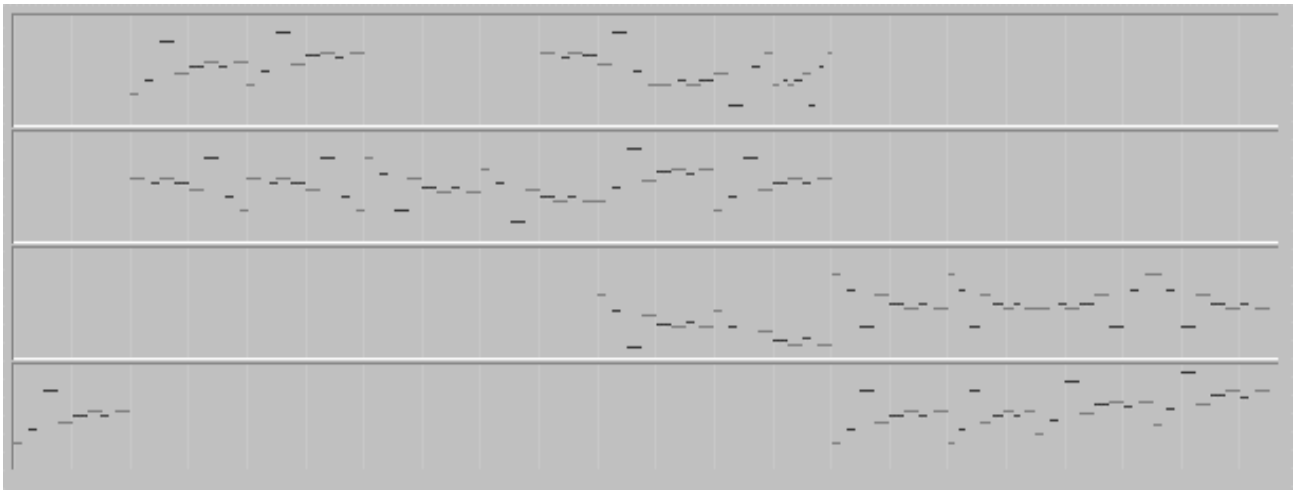

Figure 13: Sequence from source 2.

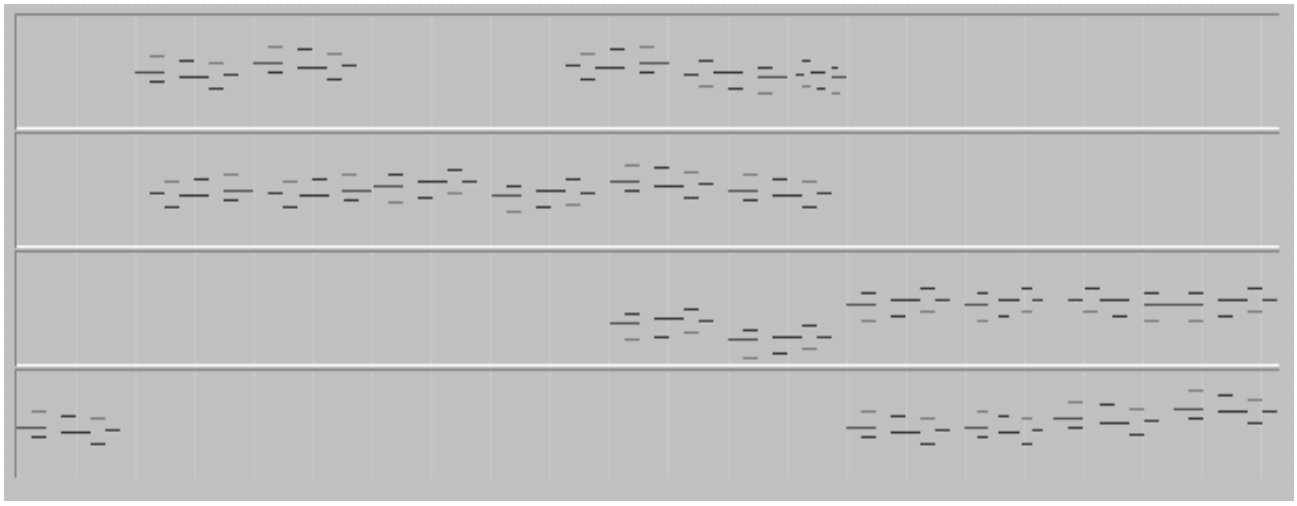

Figure 14: Sequence from source 3.

5 Limitations

Several limitation of FrameWorks in its current form have been highlighted by our preliminary experiences of using FrameWorks both as a composition tool and as an analysis tool (analysis by reworking existing pieces using the framework structure, some results of which will published in a future paper). Some of these we were already aware of since FrameWorks has yet to reach the level originally envisaged for the program. One clear hindrance is the lack of hierarchical organisation. This often leads to multiple links having to be used where one group of components are related to another group by the same relation. Because of this and for other reasons, it also leads to an over-complex visualisation of the structure that could be simplified by looking at higher level relationships and super-components. Another limitation, shown by the use of transpositions and inversions, is that FrameWorks is not aware of musical keys. This means that often it is necessary to combine additional time-dependent transpositions to these relations in order arrive at the correct musical result in tonal music. Adding specification of musical keys or modes to the FrameWorks system would allow such corrections to be made automatically, although this would need to be able to be overridden in some circumstances. Another problem is that time relations are not event aware, they simply manipulate time. Thus, relations such as retrograde, rather than time reversal, are difficult, although these can be 'fudged' by careful manipulation of event durations.

While there are such limitations with FrameWorks as it stands, it is interesting to use the software for both composition and analysis. An additional facility for adding text annotations to the framework would enhance the analysis use of the program. As the system develops in sophistication it should improve in its usefulness both as a composition and analysis environment.

5 Conclusions & Further Research

The main priorities for future development are: support for hierarchical arrangement of components; including other MIDI data such as controllers; more sophisticated relations including event based relations and some form of key/mode handling; improved notations, particularly for the sequence level; general improvements and bug fixes. Although not intended as a true algorithmic composition system, it may be interesting to develop generation systems that produce musical material for source components according to various processes and constraints.

Despite current limitations, we believe that FrameWorks provides an interesting environment for music composition that is particularly useful for experimentation with different musical ideas. There is much potential for using FrameWorks to aid analysis of extant works, while compositions created with the system provide a form of analysis in the framework used to construct them. Whether automated analysis of extant works into framework structures would be possible remains a matter for research. Using feedback from composers and our experiences of analysing musical works we hope to develop the FrameWorks concept over time to produce a highly effective and fluid system for the support of creative composition, while the analysis capabilities of the system may be useful in pedagogical circles.

References

Assayag, G. 1995. Visual Programming in Music. In *Proceedings of the 1995 ICMC.* pp. 73-6. ICMA.

Blackwell, A.F., Green, T.R.G. and Nunn, D.J.E. 2000 Cognitive Dimensions and Musical Notation Systems. *Workshop on Notation and Music Information Retrieval in the Computer Age.* At the ICMC 2000.

Green, T.R.G. 1989. Cognitive dimensions of notations. In *People and Computers V, Proceedings of the HCI '89 Conference.* pp. 443-460 Eds. A Suttcliffe. and L. Macauley, CUP.

Johnson, P 1992. *Human computer interaction: psychology, task analysis and software engineering*. McGraw-Hill.

Polfreman, R. 1999. A task analysis of music composition and its application to the development of Modalyser. *Organised Sound*, 4(1):31-43, CUP.

Polfreman, R. and Loomes, M.J. 2001. A TKS Framework for Understanding Music Composition Processes and its Application in Interactive System Design. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*. pp 75-83. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.