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

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

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

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.

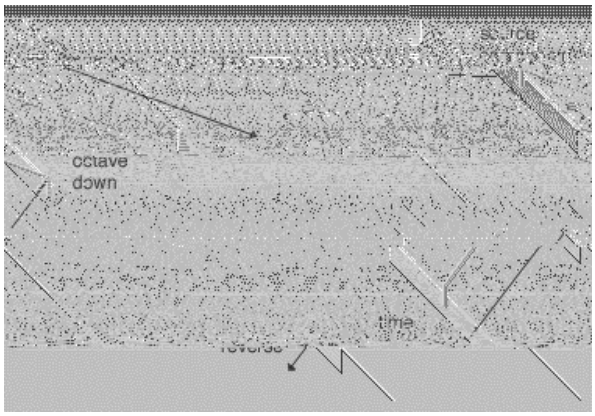

Figure 7: A simple framework

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.

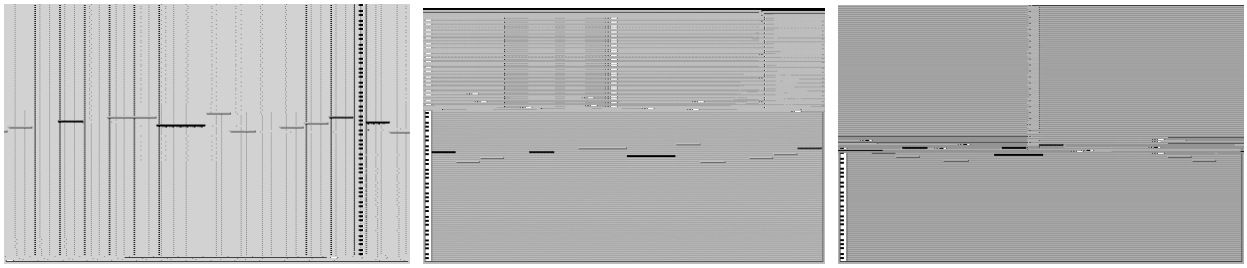

Figure 8: The components: (from left) source, octave down, (octave down and) time reversed.

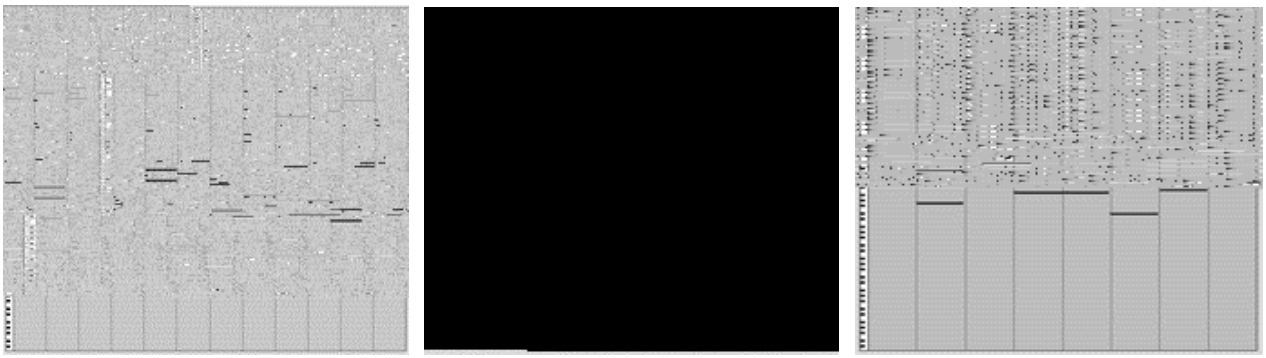

Figure 11: Three alternative source components (in 12, 16 and 8 divisions).

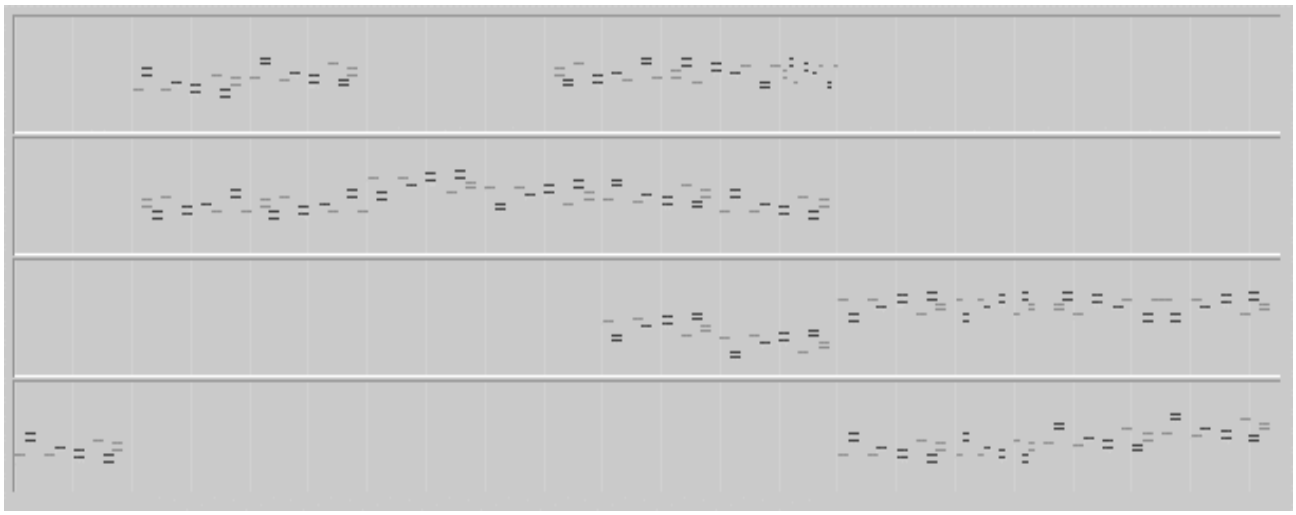

Figure 12: Sequence from source 1.

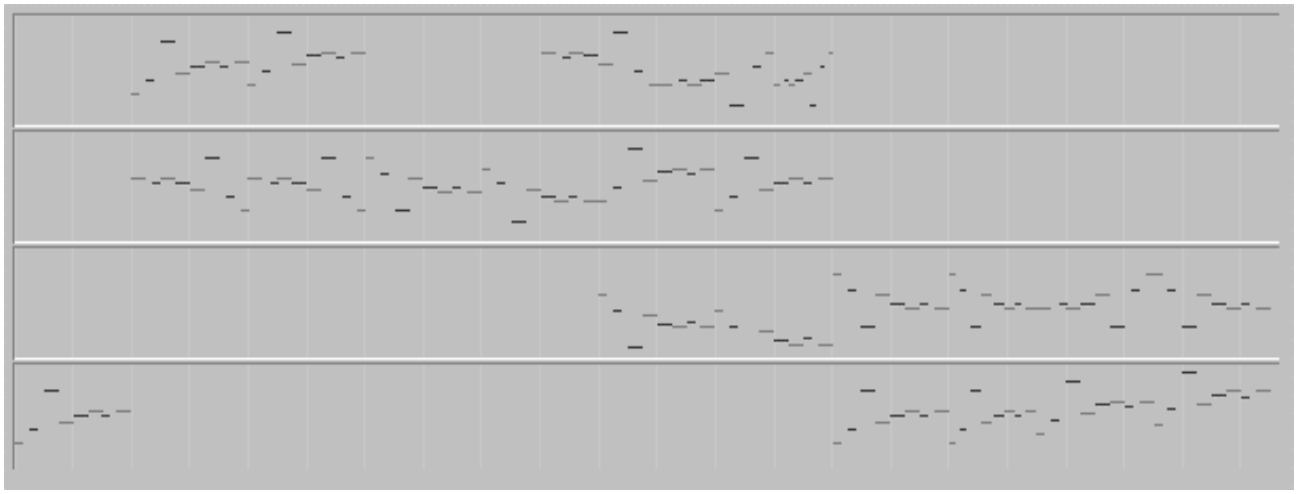

Figure 13: Sequence from source 2.

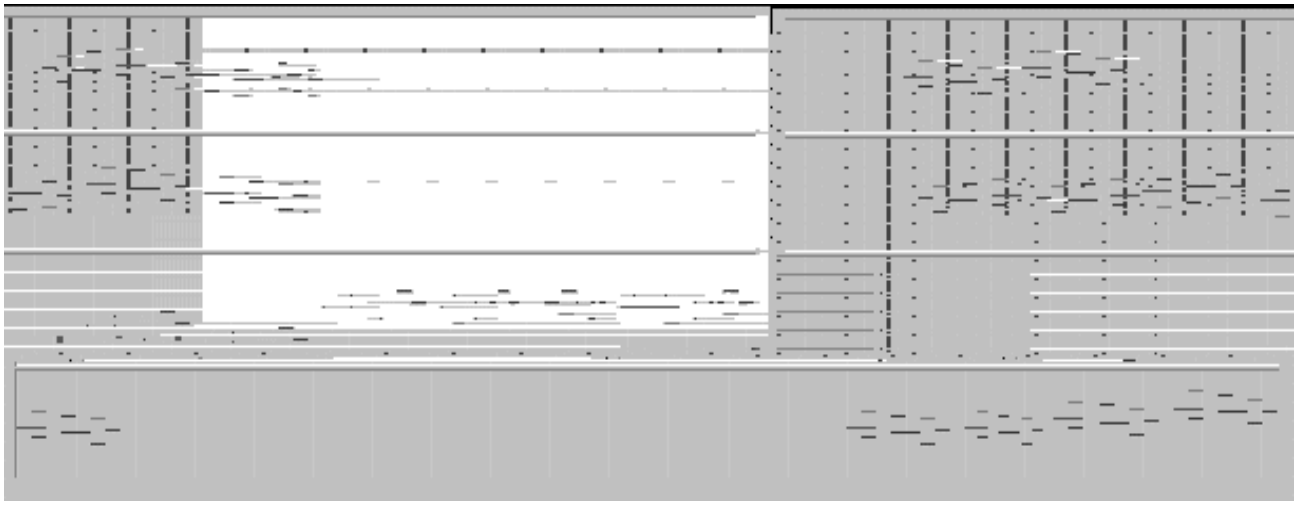

Figure 14: Sequence from source 3.

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 a

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.