Brahms – Music notation and sequencing

IF MUSIC BE THE FOOD OF LOVE

Whether it be for composing, arranging, printing or playing music, Brahms can assist with all the steps involved in computer music. Dr. Jan Wuerthner shows us how, with the aid of Brahms' KDE interface, you can be a composer in no time

The Brahms project integrates several areas of computer-aided music. As a sequencer, Brahms can read, edit and play music. A piece of music is not seen as an entire piece of audio information, as with an AIFF or WAV file, but as a succession of individual sound events in MIDI format. Brahms works with these abstract notes (which are outputted over the MIDI interface) and puts them together in a less abstract form to create music.

Brahms also incorporates a comprehensive music notation system. The MIDI format is not sufficient for the score notation of today and for this reason, Brahms has its own format (written in XML), in which notes have additional attributes for accurate representation. It can for example visually represent **double alterations** (x and bb), groupings and tuplet-brackets, as well as ornaments through numerous symbols such as slurs, trills or shakes, (de)crescendos and many others. Finally Brahms can also bring the score into a printable format.

Why Brahms was developed

Before the development of Brahms began, there were only a few music notation and sequencing programs for Linux. These were in varying stages of development and some were highly specialised. They all however exhibited an important shortcoming: they lacked interfaces for expanding their functionality. As a consequence of this, numerous projects developed their own music notation systems, although their actual goal was a lot less expansive (for example, research projects with very specialised objectives). It was in 1998 that the Brahms project arose out of this state of need, with the goal of offering a music platform. Brahms takes care of all the usual requirements of a music system, as well as being arbitrarily expandable in its function range through a suitable interface.

It will not escape the trained eye that there is a parallel with a well-known product from the world of commercially available software. Although this product originally served as an inspiration, Brahms has gone its own way in the meantime. Transferring from one software package to the other is however relatively simple.

Writing music with Brahms

Brahms is put together like a multi-track recording device that works with a piece of music (song) as its largest data structure. Apart from all the magic buttons, bells and whistles, the main window can essentially be divided in two: the "tracks" are represented, line by line, on the left and the "parts" are on the right.

Any track can consist of one or more parts. The parts belonging to a track are arranged next to each other on the right-hand side. Whole tracks or individual parts can be manipulated by a range of different attributes (see Figure 1). For example, the pitch or volume of an entire track can be varied. A track can be assigned an instrument and an output, and it can also be muted.

The actual (sound) "events" can be found in the parts of a track. All events possess a start position within their part and a duration. Those who are versed in music notation will probably miss the pauses – these can however be automatically completed by Brahms from the note positions and lengths.

The different types of events (more specifically: notes, master events and audio events) are assigned to the parts of different types of tracks (i.e. score

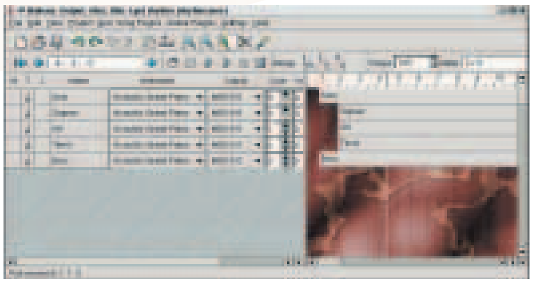
Figure 1: The tracks are displayed on the left of the main window and the parts on the right

tracks, master tracks and audio tracks). Special editors are used for the different kinds of tracks.

Brahms extensions (or add-ons) can define additional event types and new types of tracks. A good example of this is the Riemann add-on, which examines a piece of music both functionally and theoretically. To do this, it produces a harmony track and fills it with Riemann events. These new events describe the harmonious situation of the piece at different times.

A number of toolbars can be found above the tracks and parts in the main window. These include all the main functions like Play and Stop, as well as the ability to set the position, tempo and type of beat. In addition, there are multi-step Undo and Redo tools, the usual operations of cut, copy and paste, as well as two other functions for the cutting and pasting of parts. But wait, there's more: the menu allows the import and export of MIDI files, the **Mix down** of a piece of music, the reloading of Add-ons, and provides access to the global settings.

What you see is what you hear

Brahms plays each note as you enter them. The MIDI instrument, as well as the output channel, can be selected in the track settings. The current work can be played by activating either of the two Play buttons in the main window. One of the buttons plays only the marked range (the marking is located above the part field) and the other one plays the entire work.

Cold start

When starting Brahms, the fields for tracks and parts are, at first, empty. If you want to create a new piece of music, you will need at least one track with at least one part. The *Edit/Add Scoretrack* menu point lays a new track, which can receive a new part through Add Part in the right-click context menu. An even easier alternative is to double-click the large field on the right that contains the parts. Brahms then produces a part at the exact point where the mouse is located. The mouse can then be used to drag the part to any other position.

The context menu of the part offers a selection of editors, which process the contents of the respective track. The background of the right-hand field also has a context menu that accesses a number of

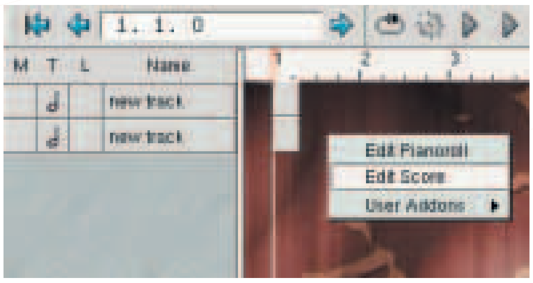

Figure 2: The context menu in the part field can start editors that work on several tracks simultaneously

editors. In this case, they process the entire piece of music (see Figure 2).

Music notation with the Score Editor

One of these editors is the Score Editor, which at first appears as an empty sheet of music. The clef and the type of tone can be selected by clicking on the violin clef (see Figure 3). At the start of the system, a red bar on the left-hand side marks the active track. The pitch (which is displayed on the appropriate toolbar) as well as the velocity (see the info. field below the system) refer to this track. A note cursor can also be used, if necessary, in this system.

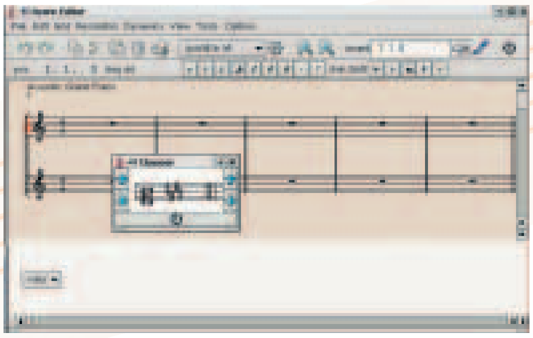

Figure 3: Notes can be entered and manipulated in the Score Editor

Most of the work can be done with the mouse. For example a single note or a group of several notes can be inserted or selected and moved with the mouse. Each note also has its own context menu, which is accessed via the right mouse button.

Toolbars for the editors

Some of the different editors use the same toolbars. The standard toolbar includes the common working functions of undo, redo, copy, cut, paste, delete and print. A selection menu makes the appropriate add-ons available and the cog to the right of this activates the selected add-on. The other icons vary the number of the displayed beats (Zoom), define the insertion position (Insert point), switch the audio play on and off when editing and ensure that the editor displays notes with explicitly defined MIDI channels in colour.

The Button bar indicates the current position (beat, strong beat and tick) and the pitch of the cursor. A left mouse click will produce the appropriate note at exactly this position. Keeping the Shift key pressed chromatically increases the tone (#), while holding the Ctrl key will decrease it (b). Remote enharmonic changes can be executed with the previous selection of the appropriate buttons on the right (bb, b, NO, #, x) or subsequently, with the help of the Note bar.

The note symbols in the Button bar determine the length of the notes to be entered – from a whole note right down to a 64th of a note. The notes are, as a rule, rounded to the nearest 16th, although this can be altered in the Resolution menu. The next two

Double alteration This changes a tone by two semi-tone steps. For example, from the point of view of the harmony, a doubly increased G is just as different to an A, as a Fis is to a Ges. These differences only escape with a moderate temperament, which is usual in computer music as well as with key instruments today. Neither the enharmonic mix up (Fis/Ges) nor the double alteration is part of the MIDI format. MIDI only describes the sound, not the representation.

Mix down Mixes the notes from several tracks into an individual track.

buttons select dotted notes and meter triplets. It is more efficient however to set the note length using the key combinations of Alt+1 to Alt+7.

The Note bar indicates the parameters of an individual note. It is deactivated when opening an editor and is displayed through the *View/Notebar* menu point or with the key combination Alt+E. The fields in the Note bar are filled with values as soon as a note is selected with the mouse or the cursor keys. The Start field describes the position in the piece (as opposed to within the part) by defining three values: beat number, strong beat (for example 1, 2 or 3 in a 3/4 beat measure) and tick within a strong beat. A quarter note consists of 384 ticks. Arpeggi (ornamental notes that are melodically connected with the following main note) can be implemented by changing this value.

The length field indicates the note duration in ticks. The pitch describes the pitch by defining the tone, followed by the octave position. The beat value of the note (0 to 127) can be found in "vel" (velocity), and the "chn" field (channel) indicates the number of the MIDI channel (0 to 15). A MIDI channel number of x signals that the setting of the associated track has been used. This value, displayed in the note sheet (Figure 4), can be coloured in with the paint brush symbol in the toolbar..

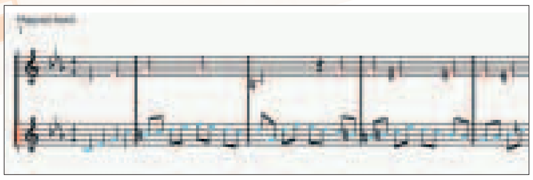
Figure 4: When mixing (main window menu, Edit/Mixdown), the individual notes of the MIDI channels are mixed into the original track

Music in word and picture

The *Score Editor* offers different tools (see table 1), which can be used to complete or supplement the score. These aids are selected in the *Tool* menu of the editor window. A tool remains active until another is selected. The symbol selection windows, which can appear with numerous different tools, disappear again when a new tool is selected.

Brahms can also print the score, although the sequencer currently requires the shareware program MUP (The *MU*sic*P*ublisher) for this. The output from Brahms serves as the input for MUP, which then supplies a very nicely formatted PostScript file. The shareware aspect of this software is a thorn in the side for many, and as such Brahms' developers are currently working on an output through the Lilypond GPL program.

After one or more parts have been produced in the editor, the main window can represent these in a variety of different ways. Using the *Edit/Preferences/Desktop* menu point the following variations can be selected: plain, show track name, show instrument, show part events, and show events and pitches (Figure 5).

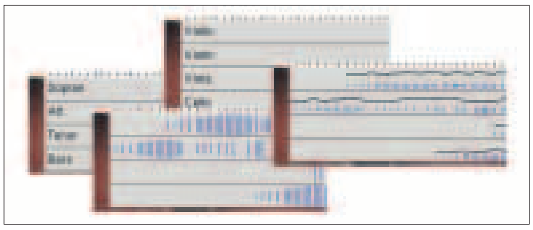
Figure 5: The parts can be viewed in a variety of different ways

More editors with more input possibilities

Different forms of music notation have been developed over the years. Today's notation has grown over the centuries, meaning that the roots come from a time when the moderately tempered C was not yet known. This will, at the latest, be seen clearly with the attempt to model your structure into an algorithm.

In this age of computer music, the representation of a Piano Roll Editor enjoys great popularity. Pianists or musicians, who are not familiar with classical notation can write and work on music using the keyboard on the left. The tones are then graphically set against time in a large diagram. Of a similar set up is the Drum Editor, with which drum tracks can be set down in a true to life form (Figure 6).

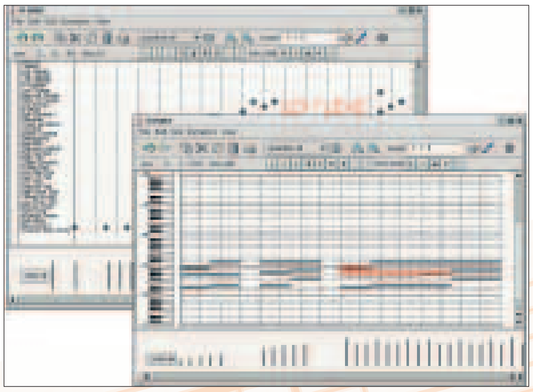
Figure 6: The Drum and Piano Roll Editors offer different representations of a track

In all editors, a selected group of notes can be moved (or dragged) with the mouse, or duplicated by simultaneously pressing the Ctrl key. If you hold the Shift key at the same time, then only the time and not the pitch of the notes are moved or copied. The note lengths can also be corrected in the Piano Roll Editor using the mouse. You can do this by clicking the end of the Note bar – the cursor changes its shape by rolling over this region.

Since Brahms encases many editing functions in abstract classes, the implementation of other editors is not a problem. Editors for the guitar and the modal notation of medieval music are both on the wish list.

Brahms supports aRts and ALSA

At present Brahms supports two architectures for the output of sound: ALSA in its version 0.5.x and aRts. Both operating modes have their pros and cons. Brahms also plans to support ALSA 0.9.x in the future, but this is dependent on how aRts is developed. In the long term, a platform is to be supported, whose interface can comply with all the sequencer's requirements. aRts is a promising candidate to supply such a platform as well as offering many other advantages besides.

aRts is not only a software synthesizer, but has also been used as the sound server of the KDE desktop for quite a while. In this mode, the many aRts capabilities can be utilised, such as sound synthesis with the **aRts-builder**, MIDI recordings from a keyboard, simultaneous playing of several sound sources in the system and the list goes on.

The recording and playing of audio data with audio tracks is in development at the moment – the first attempts have already been successful. Each Brahms track thereby receives its own mixer channel (Figure 7).

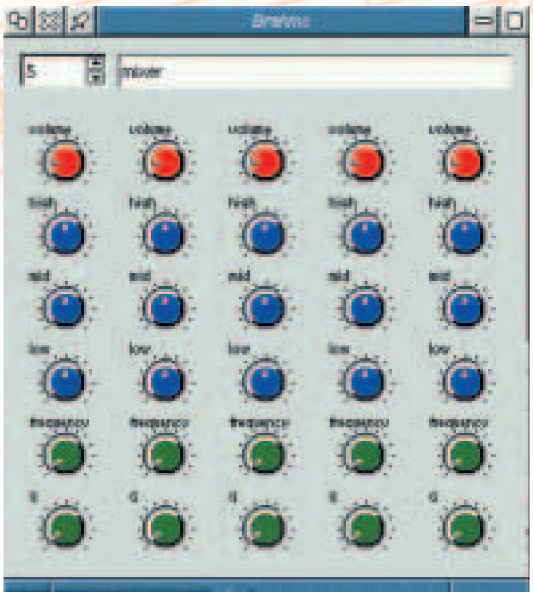

Figure 7: Each Brahms track is assigned a mixer channel

The aRts mode is used differently in KDE 2.x and KDE 3.x. If no audio can be heard, then the MIDI manager settings are usually responsible. This MIDI manager is a part of the *artscontrol* program and can be found under the *View/View MidiManager* menu point.

A synthetic MIDI instrument, for example a Slide, can be created in the manager with *Add/aRts Synthesis Midi Output*. This instrument then appears as a new output. For Brahms to be able to use this instrument, it must be connected to the sequencer software. To do this, the instrument must be selected in the output list and Brahms must be selected in the MIDI input list – they are then connected with the Connect button (Figure 8).

Table 1: Tools used in Brahms

Tools	Range of application
Insert notes	This (preset) tool is used to insert and manipulate notes.
Add note symbols	Individual notes can be decorated with this tool (Figure 9).
Add system symbols	This tool positions symbols, which are not to be assigned to a note, directly into the system (Figure 9).
Add lyrics	Even syllables and text can be assigned to the notes. Clicking a note opens a small text field, which can be closed again with the Enter key. If text is to be entered for the following note, then the Spacebar should be used to close the syllable. The reason for this is that Brahms then automatically opens a new text field for the next note (Figure 10).

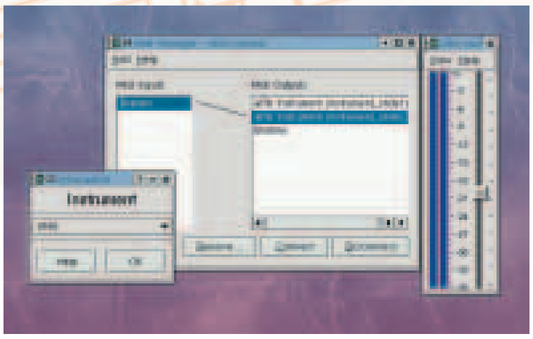

Figure 8: The output of Brahms is connected with an aRts instrument in artscontrol

ALSA, the Advanced Linux Sound Architecture

Brahms can be also operated in the ALSA mode, if ALSA version 0.5.x is installed:

```
brahms -o alsa
```

As far as configuration goes, all that is required is for the ALSA point to be defined as the output in every track. The advantage here is that the sound fonts of the soundcard can be used. In contrast, aRts only handles synthetic instruments at the moment, which is rather unsatisfactory when playing orchestral instruments. Apart from this, aRts only permits a small number of tracks on less powerful systems. Where both aRts and ALSA are supported as sound and MIDI drivers, the ALSA option is made invalid in Brahms.

Flexibility through modules

The essential components of Brahms are stored in libraries, which are dynamically loaded by the program at runtime. One could describe Brahms as merely the glue that holds these libraries together. The core library contains all the core components and

aRts-builder: The aRts-builder is a component of the aRts project and serves the production of synthetic instruments. Different modules' (or effects') inputs and outputs can be simply built up and implemented with the graphical user interface.

Hugo Riemann: Riemann (1849-1919) was a music researcher who developed a "function" theory, which states that each accordic entity can be attributed to one of three functions – Tonic, Sub dominant and Dominant – and that each of these three-sounds can be represented by one of their individual tones.

functions (not the user interface, however) as well as numerous abstract classes. These include the structural components of a piece of music (for example song, track, part, event, note), all operations that affect these elements, as well as the basic functions of the editors.

The elements of the core library require a presentation, i.e. a user interface. New graphic interfaces can be implemented with a minimum of effort due to the strict separation of function and presentation. There are currently two modes of presentation: Text Presentation and Kde Presentation. The KDE variant is loaded by default. If Brahms is started with the option:

```
brahms -p text
```

then the entire application runs on the console (text mode). Even the presentation of the notes in the *Score Editor* is abstracted in the core library. It was therefore very simple to implement an appropriate presentation in the text mode (Figure 9). Both figures illustrate the same section of music. The eighth notes are individually set in the text form: the abstraction contains the information about the groupings of notes, but leaves it up to the presentation whether to use these or not.

It is easy to see that the text form is not very simple to operate. In an emergency, it would be possible to start Brahms from the command line, i.e. without a graphic desktop. This presentation however basically serves as a proof of concept. It would be conceivable to implement a pure Qt- or Gtk-based presentation. An interesting alternative would also be a Web variation, whereby Brahms would act as a music server, which could be operated by a browser.

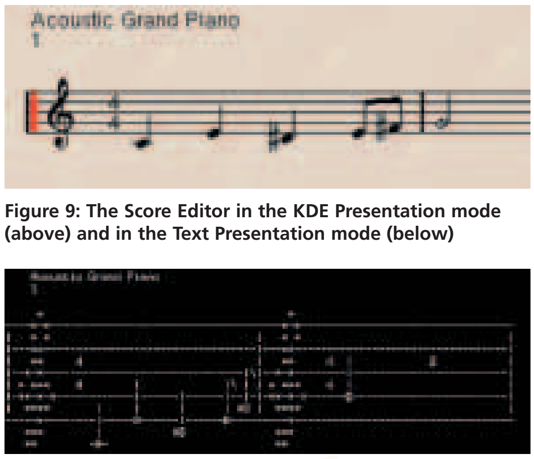

Figure 9: The Score Editor in the KDE Presentation mode (above) and in the Text Presentation mode (below)

Brahms is extensible

The big advantage of this architecture lies in the extensibility of Brahms. Extensions (Add-ons) can be written and compiled without the entire Brahms application having to be translated again – even without having to reboot. A concise API offers a simple interface, through which a piece of music with all its components can be edited.

In the category of such extensions are the three functions Quantize, Quantize Length and Fixed Length (which should be in all sequencers). It is pleasing to know that these operations also include an Undo function. The currently available extensions are summarised in Table 2.

The **Riemann** extension is not completely implemented and as such is still in its prototype stage. It's worth noting that this module, when calculating the harmonies, uses the tone material at the respective time (vertical), as well as considering the musical context (horizontal). A sound that consists of Fis and C in the context of G major, is thereby more likely to be interpreted as D major with a minor seventh than a diminished Fis.

Beyond this, the Riemann extension shows that Brahms extensions can introduce new types of tracks (Harmony, symbolised by a heart) and new types of

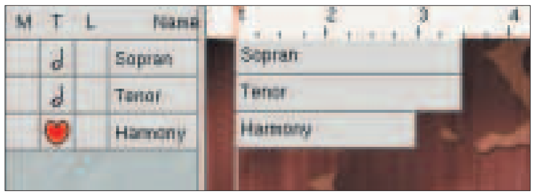

Figure 10: The Riemann extension introduces a new type of track

Table 2: Extensions for Brahms

Name	Category	Function
Quantize All	Quantize	Rounds the initial positions, the note lengths and other events.
Quantize Length	Quantize	Rounds the note lengths.
Fixed Length	Quantize	Sets the note lengths at a constant value, which is selected in the editor.
Dump	Testing	Outputs the total content of the piece of music on the standard output.
Debug	Testing	Passes on any further information to the standard output.
ExtractLyrics	Output	Outputs the lyrics (i.e. text) assigned to the notes.
Stretch	Edit	Multiplies each initial position and length by a factor of two.
Revert	Edit	Reverses the direction of a piece of music (with respect to time).
Ear Training	Harmony	Trains intervals and chords (extendable at will).
Parallels	Harmony	Searches for quint and octave parallels in a piece of music. A new track is created and filled with copies of these prohibited notes. Covered parallels are highlighted with colour by setting the MIDI channel. This is a great help when creating your own compositions and when analysing works.
Riemann	Harmony	Assigns the suitable harmonies to a piece of music at different times.

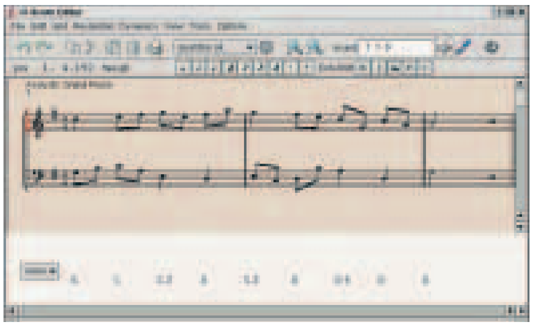
Figure 11: The Riemann events are displayed in the lower part of the editor

events (Riemann Events). The events are displayed in the lower part of the editor. The selection menu, which otherwise contains the beat values as its sole entry, now also supplies the Riemann values (Figures 10 and 11).
An extension is indicated either as an option at the Brahms command:

```
brahms -a dump -a riemann -a stretch
```

or loaded at runtime (via the menu point *File/Load Add-on...*). The libraries to be loaded are located in the *$$KDEDIR/lib* directory subsequent to installation and begin with libBrahmsAddon. If an extension has been successfully loaded, it can be found in the Editor toolbar's selection menu or in the context menus of one the following: piece of music, track or part (Figure 12).

Be your own composer

For those of you who would like to write your own extension, the well-documented dump extension can be used as a template. The extensions are located in the *brahms/addons* directory in Brahms' sources and

Installation

The brahms-1.02-kde2.tgz source package for KDE 2 is available on the Brahms homepage. The following instructions unpack, configure and install the program:

```
tar xzvf brahms-1.02-kde2.tgz
cd Brahms
./configure
make
su
make install
```

The last step must be executed as the root user (hence the *su* command). On some distributions (for example SuSE 7.3), an option is necessary in the configuration, and the third step then changes to:

```
./configure --prefix=/opt/kde2
```

Listing 1: Revert extension

```
01 void Revert::song(Song * song) {
02  Position endpos = (new SongIterator(song))->endPosition();
03  int sz = song->size();
04  for (int i=0; i<sz; i++) {
05   Track * track = (Track *) song->get(0); if (track==0) return;
06   Part * part  = (Part *) track->first(); if (part==0) return;
07   Track * newTrack = song->createTrack(track->isA(),0);
08   newTrack->setName(strdup(track->name()->getValue()));
09   if (track->isA()==SCORETRACK) {
10    ((ScoreTrack*) newTrack)->setProgram(track->program());
11    ((ScoreTrack*) newTrack)->setChannel(track->channel());
12   }
13   Part * newPart = new Part(newTrack);
14   newPart->setKey(part->key()); newPart->setClef(part->clef());
15   // ---------------------------------------------------
16   for (Iterator i = Iterator(track); !i.done(); i++) {
17    Event * event = (Event*) (*i)->copy();
18    newPart->setStart(event, endpos - i.part()->end(event));
19    newPart->add(event);
20   }
21   newTrack->add(newPart); song->add(newTrack);
22   song->remove(track); delete track;
23  }
24 }
```

the API documentation is also very helpful. The documentation is installed with Brahms, as well as being available on the Brahms homepage.

A small example shows how little work it takes to create such an extension. Aside from the formalities such as the declaration of context, category and name, the essential part of the Revert extension consists of the *song()* method (see Listing 1).

The module firstly determines the end position of the piece of music. The outer loop goes through all the tracks (line 4) and the inner loop iterates through all the events (line 16). The position of each event is subtracted from the end position, thus meaning that the time becomes a mirror image of itself. Copies of the events (not the events themselves) are finally edited and laid down in a new track with a new part (line 19). As soon as a track is edited, the original is removed in line 22.

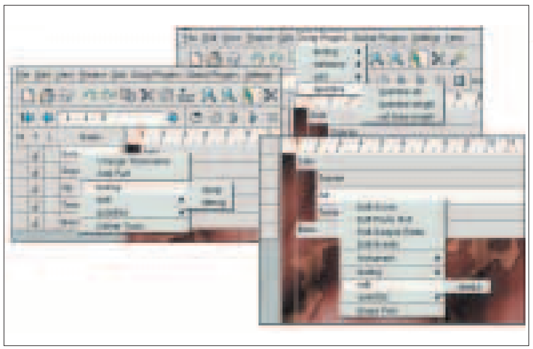
Figure 12: The loaded extensions are accessed through the context menus

Info

Brahms homepage:
http://brahms.sourceforge.net
Official site of the aRTs project: *http://www.arts-project.org*