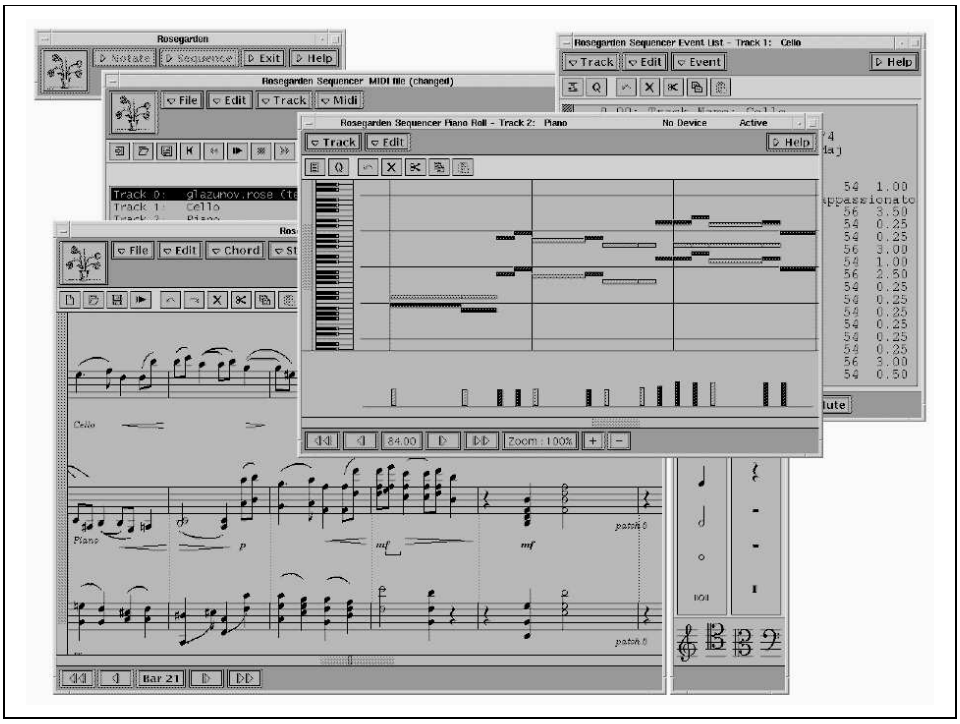
Rosegarden



User Manual

Andy Green and Chris Cannam

Contents

1. Useful Concepts

1.1 Top Box

1.1.1 The Rosegarden Top Box

The Rosegarden program suite comprises three parts: the Top Box, Sequencer (see *The Rosegarden Sequencer* [15]) and Editor (see *The Rosegarden Editor* [15]). The Top Box is a small program whose task is to coordinate the activities of the other two. You should normally run the Rosegarden suite by starting up just the Top Box; this is usually carried out by running a program called ''rosegarden''. Once the Top Box has been started, the Sequencer and Editor can be started from it.

The Top Box appears as a small window containing a single row of buttons. These are labelled 'Notate', 'Sequence', 'Exit' and 'Help'. There is also an 'About' button, the square to the left of the window containing the Rosegarden logo, which, when pressed, yields a short but flashy enumeration of the Top Box's authors. (Most of the Rosegarden programs' larger windows are equipped with 'About' buttons.)

The other buttons behave as follows:

- The 'Notate' button invokes *The Rosegarden Editor* [15]. This is a tool for transcription, display and editing of music in a traditional musical notation.
- The 'Sequence' button invokes *The Rosegarden Sequencer* [15]. This is for recording, displaying, manipulating and playing music in MIDI format.
- The 'Exit' button exits the Top Box, and also exits the Editor and Sequencer if they have been started up.

- The 'Help' button calls up the interactive hypertext help system (see *The Rosegarden Help System* [12]). The Help system contains the entire Rosegarden reference manual, with cross-references; if you're reading this on your computer screen rather than on paper, you may well be using it. Help may also be called up from the 'Help' buttons on the Editor and Sequencer, so long as the Top Box is there to provide it.

The Editor and Sequencer can also invoke and use each other, so long as the Top Box is running; details of how to start the Sequencer from the Editor are in *Sequencing a piece of music* [27] , and of how to start the Editor from the Sequencer are in *Preparing a MIDI file for the Editor* [31] .

1.2 File Boxes

1.2.1 File Boxes

Using many of the major functions within the Rosegarden editor and sequencer involves understanding the File Box. File Boxes are smallish windows which are popped up whenever a Rosegarden program requires you to select a file or enter a filename for opening or saving. Their purpose is to list the files in a given directory, allowing you select files, move to different directories and other simple operations. A File Box is divided up into five areas:

- The top label. This shows the name of the directory being inspected; if you click-and-hold upon it you get a menu of directories higher up in the file tree; selecting one of these moves to and displays that directory.
- The main file list. This shows the names of the files and subdirectories in the current directory, with subdirectories first. The list may not be complete, because by default (although you can change this using the filter button, below) the File Box filters out all files with names beginning with a dot or ending with certain special characters. If the list is long, it can be scrolled through with the scrollbar to its left. The entries in the file list can be clicked and selected, as explained below.
- The filter button. This starts out labelled 'Normal files'; by clicking on it you can choose which sorts of files to display in the file list. The choices are 'Normal files', which shows only those files with names which suggest that they are not backup or hidden files; 'Dot files', which shows hidden dot files as well as normal ones; and 'All files', which shows hidden dot files and backup files as well as normal ones.
- The current-file label. This is initially empty, but when the name of a file (as opposed to a subdirectory) is selected in the main file list, it is copied down to this label. If you're using the File Box to choose a file to save into, you can also click on

this label and type a new filename into it. (File Boxes for loading files don't allow you to do this.)

- The action buttons: 'Load' or 'Save', 'Cancel' and 'Help'. Clicking on the button labelled 'Load' or 'Save' (depending on what the file box is being used for) causes the file whose name is displayed in the current-file label to be used. The 'Cancel' button cancels the function that this File Box is being used for, and the 'Help' button calls up help on this function.

Files in the file list can be selected by clicking on them; selecting a file just copies its name to the current-file label, where, if this File Box is for a Save operation, it can be edited. Double-clicking (clicking twice in quick succession) on the name of a file selects and activates it, in the same way as selecting it and then using the 'Load' or 'Save' button. Clicking on the name of a subdirectory moves into that directory, and displays its contents in the main file list.

1.3 Toolbars

1.3.1 Toolbars

The Rosegarden Editor and Sequencer are each equipped with a toolbar, a series of buttons below the menu bar. This manual does not generally describe how to do things using the toolbar buttons, because despite their cute and appealing appearance they're considered to be more useful once you already know how the menu functions work. Instead here's a quick overview of how to use the toolbars and how they relate to the menus.

Each toolbar button corresponds to one menu function, although not every menu function has a corresponding button. The buttons are also grouped according to their menu, so the buttons which carry out functions from the File menu are all grouped at the left, then there's a bit of a gap, then functions from the Edit menu and so on. In every case, pressing the toolbar button has exactly the same effect as selecting the menu item it corresponds to.

The buttons are all labelled with a descriptive picture which hopes to give some idea (or at least a reminder) of the function of the button. However if you move the mouse pointer over a toolbar button and leave it there for a second or so, a label will appear showing the text of the menu function that that button operates.

1.4 Online Help

1.4.1 The Rosegarden Help System

Rosegarden is equipped with a powerful online hypertext help system, containing the whole of the Rosegarden Reference Manual in a hierarchical cross-referenced format.

The Help is a generic browser based upon the GNU Info format, with a structure similar to that used in several other packages such as GNU Emacs.

The structure of help pages is described in the next page. If you are using the online Help to read this, you can reach the next page by clicking on these words: *Help organisation* [12] ; a reference guide to the features and buttons of the Help window can be found under *Using help* [12] .

1.4.1 Help organisation

Help documentation is organised into a set of pages, structured in a tree hierarchy. There is one page (or 'node') at the top of the tree: this is called 'Top'. This has a menu of subpages, each of which has its own menu; finally, the pages containing actual information, like this one, are at the bottom (the 'leaves') of the tree. You can get from a page to any of the subpages on its menu simply by clicking the first mouse button on the name of the page you want to go to.

The subpages accessible from a page's menu are stored in order; each, in addition to being a 'child' of its 'parent' menu page, has up to two 'sibling' pages, 'previous' and 'next'. The previous, next and parent pages are always available through the arrow buttons immediately above the help text window. From the current page, for instance, the previous page is available through the button labelled 'The Rosegarden Help System', the next through 'Using help', and the parent page through 'Rosegarden'.

The Help system also provides some hypertext support, in the form of cross-references. These are words or phrases which appear in the text emboldened or in an italic face, and which may be clicked upon in order to transport the user directly to their page. For instance, when using the Help system you can click on the words *The Rosegarden Help System* [12] in this sentence to transport you to the page of that name, which happens to be the one before this, and on *Help organisation* [12] to redisplay this very page.

1.4.1 Using help

The Help window is split into three rough areas: the button bars, at the top; the help text window, next down; and the help menu window, at the bottom. Normally only one of the text and menu windows is in use: if there is text in the current page, there will be no menu, and if there is a menu, the only text will be the page's title.

The buttons used for motion --- those which take you to the previous, next and parent pages, plus the buttons which operate cross-reference links and menu selection --- are described in the previous page, *Help organisation* [12] . The other buttons are:

- the About button. This is the one with a picture of a pot of roses on it. If you click on it, it'll tell you some particularly uninteresting things about the author.
- the Search button. The Help system has a search system which, given one or more keywords, can supply a list of all the help pages which might be connected in some way with those keywords. It does this just by scanning the text of the help pages to see if it can find your keywords in them anywhere. (This doesn't mean that it can also search for parts of words. It can't.) To use it, click on 'Search', station the mouse pointer just in front of the words 'Search string:', type in one or more words, and without pressing *return*, hit the 'Execute Search' button at the top of the window.

After searching, if you haven't found a more interesting page as a result of the search, you can return to the help page you were previously looking at with the 'Return to Help Page' button at the top of the Search display.

- the History button. This, when pressed, switches to the History display, which shows you which pages you've read since Help was started. You can click on these to return to them, or on the 'Show Current Page' button to return to the page you were on when you pressed History.
- the Exit button. This closes down the Help window until next time it's called for.

2. Common Tasks

2.1 Introduction

2.1.1 The Rosegarden Editor

The Rosegarden Editor module allows the user to display and edit music in a traditional musical notation, transcribing from MIDI sources, saving and loading in MIDI or the editor's own internal format, or printing using the MusicTeX music typesetting system. The editor can be run independently or as part of the Rosegarden suite; this help is only available when it is run as part of the suite.

2.1.1 The Rosegarden Sequencer

The Rosegarden Sequencer module allows the user to manipulate MIDI files and data in a variety of ways, as well as providing facilities for MIDI recording and playback. The sequencer can be run either independently or as an integrated part of the Rosegarden suite. When the sequencer is run independently this help information will not be available.

2.2 Editor

2.2.1 Starting a new piece of music

To begin work on a new piece, call up the *Editor File* [33] menu and select the *Editor File - New* [33] function. A small window will then pop up asking you how many staffs you want in the piece. The default number is 1; if you only want one, just click on the Apply button in the window. Otherwise, use the keyboard to delete the 1 and type in the number of staffs you need. Then click on Apply, or Cancel to abandon the new piece.

The main editing window will then clear and the opening stubs of a new set of staffs will appear. The editor chooses a likely set of clef, key and time markings (C major, 4/4 time, and each staff in either treble, alto or bass clef depending upon its position), and gives the staffs some default (uninformative) names. All of these things can be changed: see *Changing bar properties* [23] and *Bar - Time Signature* [50] to change the time, *Editing and using the clipboard* [20] , *Using the Palette* [18] and *Inserting notes and signs* [18] to delete the clef and insert a new clef, and *Operations on staffs* [22] and *Stave - Rename Staff* [43] to change the names of the staffs.

2.2.1 Opening an editor file

Files which have previously been saved from the editor (see *Saving an editor file* [16]) can be reopened for further editing using the *Editor File - Open* [34] function on the *Editor File* [33] menu. To do this, select the *Editor File - Open* [34] menu entry, select the file to be opened on the file box (see *File Boxes* [10]), and click on the Load button. Provided the file is a properly formed Rosegarden editor file, it will be opened and its contents displayed on the main editing window.

Opening a new file does not affect the buffer which was previously being edited: its contents, and those of every other buffer which has been opened since the editor was started and has not been closed, are available by selecting the buffer's name on the list on the *Editor File* [33] menu. If you do want to finish editing the current buffer for good when loading a new one, select *Editor File - Close* [34] before opening the new file.

2.2.1 Saving an editor file

To save a buffer into its associated disc file, choose the *Editor File - Save* [34] function from the *Editor File* [33] menu.

To save a buffer into a new file, without affecting the contents of any file(s) which may previously have been associated with it, use the *Editor File - Save As* [34] function instead. This pops up a file box (see *File Boxes* [10]) to prompt for a file into which to save the buffer, and saves it in there. The new file will subsequently be that associated with the buffer, replacing any old one (so subsequent uses of *Editor File - Save* [34] will save into this new file). The name of the buffer (displayed in the title bar and in the *Editor File* [33] menu) is also changed accordingly.

If there is no disc file associated with the current buffer, for example if the buffer is being used to display a MIDI file opened straight from the sequencer, the *Editor File - Save* [34] function will not be available. To save such files you must tell the editor where they should go, using *Editor File - Save As* [34] .

2.2.1 Printing a piece of music

Although we would like to include support for printing directly from the Rosegarden editor (via PostScript, for instance), this is not yet available. Printing music requires use of, and familiarity with, the MusicTeX music typesetting language (see *Exporting MusicTeX files* [27]).

2.2.1 Moving around

When a buffer is first opened, the main window displays its contents, starting with the first bar (or 'measure,' for the more transatlantically or transcontinentally inclined) shown at the left of the display. When a piece of music becomes taller or wider than the window size, it becomes necessary to move the view around within it.

The main editing window is equipped with two scrollbars, a horizontal one at the bottom and a vertical one to the left, and with five motion buttons at the bottom.

The scrollbars work in the same way as on many other X applications, such as 'xterm'. While the pointer is over a scrollbar, the first mouse button scrolls towards the end or bottom of the piece by an amount proportional to the distance along the scrollbar the pointer is placed; the third button scrolls in the other direction by a similar amount; and the second button causes the displayed section to jump to the point in the piece corresponding to the distance along the scrollbar the pointer is placed.

The motion buttons move the display (respectively) left by a whole screenful, left by a single bar, to any bar of your choosing (upon clicking on the bar number display you are asked for a bar number to which to jump), right by a single bar and right by a while screenful. You can also use the cursor keys on the keyboard to move around a piece. Left and right will always move by one bar left or right, but the up and down buttons will only work in edit mode (see *Using the editing cursor* [19]) --- when inserting notes, they are instead used to change the length of note to be inserted.

2.2.1 Using the Palette

The Palette is the smaller window which appears to the right of the editor's main window when the editor is started up. It is used to select items to be inserted in the music in the main window.

The Palette window contains a set of buttons for selecting note or rest lengths, clefs, and accidental modifiers. If you click on a note on the palette, it will be highlighted and that length of note will be selected for insertion into the music on the main window. The cursor in the main window will change to a cross to show that insertion mode is enabled, and then clicking with the left mouse button over the main stave of music will insert notes. (See *Inserting notes and signs* [18] .)

The selection can be cleared, to return to editing mode (see *Using the editing cursor* [19]), by clicking again on the selected note in the Palette. (Clicking with the right mouse button on the main window will also return you to editing mode.) To select a different length of note or rest, or a clef, for insertion, just click on it.

At the top of the Palette there are four 'modifier' buttons. These show the sharp, flat and natural signs, and a dot. If any of these are selected, they will be used to modify inserted notes (and rests, in the case of the dot). So for example, to obtain a dotted-quaver rest, select the quaver rest and the dot modifier; to obtain a sharpened crotchet note, select the crotchet, ensure that the dot modifier is not selected, and select the sharp modifier. Double-sharps and double-flats are not available.

Finally, at the very top of the Palette is a toggle button labelled ''Follow Key''. If this is ticked, notes inserted into the musical stave will be transposed into the right key signature, so you will only need to use the sharp, flat and natural buttons for accidental notes not conforming to the signature. (Accidentals that you explicitly select from the buttons on the Palette will still be used in preference, however.) If the ''Follow Key'' button is not ticked, all notes will be entered as naturals unless you have explicitly requested an accidental.

2.2.1 Inserting notes and signs

The editor has two modes of operation: insert and edit mode. The two can be distinguished, because in insert mode, the mouse pointer becomes crosshair-shaped when it enters the main edit window; in edit mode, an approximately I-shaped cursor is used instead. To enter insert mode, one of the insertable items on the Palette must be selected (see *Using the Palette* [18]); if none are selected, the editor is in edit mode, in which various editing operations can be carried out (see *Using the editing cursor* [19] and *Editing and using the clipboard* [20]).

To insert an item into the music, select it on one of the Palette's item buttons, then click the left mouse button when the mouse pointer is somewhere within the main window. If the pointer is over a staff when the button is pressed, the item will be inserted under the pointer. If the item is a note, the position of the pointer vertically on the staff will be used to determine the pitch of the note. You can use the Palette's sharp, flat, natural and dot buttons (see *Using the Palette* [18] and *Considerations in inserting notes* [19]) to apply accidentals and dots to the note you want to insert; you can also hold the left Control or Alt key on the keyboard for a sharp or flat respectively, the right Alt key for a natural, or the left Shift for a dotted note; and the up and down cursor keys, when used in insert mode, change the selected note length. (Technical warning: some older X window managers may intercept mouse-clicks with the Shift, Control or Alt button in order to perform window selection or motion functions. If these keys don't appear to work as expected, it could be something to do with your window manager.)

When you've inserted enough notes or rests into a bar to fill the bar according to the time signature in force at that point of the staff, the notes will spill over into a new bar. If necessary, some notes or rests may be split and tied over the bar line in order to maintain a bar length as close as possible to the 'correct' length. If you then insert more notes before a tied note so as to push it entirely over into the next bar, it will be rejoined into a single note.

See also *Considerations in inserting notes* [19] .

2.2.1 Considerations in inserting notes

After a note or other item has been inserted, the stave will be reformatted and redisplayed with the new item in place. This could cause a considerable change in the layout of the stave: for instance, inserting a very short note into a bar previously containing only fairly long ones will cause the bar to become vastly wider.

Inserting an item can have an effect on other items after it in its staff. For instance, inserting a clef will change the pitch of any notes following it, and inserting a key signature may alter any accidentals displayed on subsequent notes.

In any case, the display of notes depends to some extent upon the key signature. For example: a flattened note entered into a piece whose key signature already lists a flat for a note of that pitch will be displayed without the flat; a note entered into a piece with a sharp in the key signature for a note of that pitch will be displayed with an explicit natural sign, unless the ''Follow Key'' option on the Palette is ticked; and so on. Similarly, display of notes can be affected by the presence of accidentals on notes earlier in the same bar.

2.2.1 Using the editing cursor

Editing music in any other way than simply inserting notes and signs requires that the program be in edit mode. This mode is enabled whenever insert mode isn't: i.e., when no insertable items are selected on the Palette window (see *Inserting notes and signs* [18] and *Using the Palette* [18] for further explanations). You can enter edit mode by clicking again on the selected item on the palette to clear its selection, or by clicking with the right mouse button over the main window. When the editor is in edit mode, the mouse pointer turns into an I-shaped bar upon entering the main edit window.

There are two fundamental editing selection techniques: placing the cursor, and sweeping out an area. Almost every editing function requires that one or the other of these be carried out to select which bit of music the editing should affect.

To place the cursor, simply click with the first mouse button when the mouse pointer is over some area of musical staff on the main edit window. A tall, narrow cursor sigil will be drawn beneath the cursor. Subsequent clicks elsewhere will move the cursor, and scrolling the stave (see *Moving around* [17]) or entering insert mode (see *Inserting notes and signs* [18]) will cause the cursor to be removed. For examples of functions which can be executed when the cursor has been placed, see *Editing and using the clipboard* [20], *Operations on staffs* [22], *Changing bar properties* [23], *Tempos and times* [24] and *Adding text* [24].

To sweep out an area, first click with the first mouse button over an area on a staff; and then, *without* releasing the mouse button, drag the mouse left or right along the staff. A rectangular area will be swept out to follow the pointer as you move the mouse; its edges will always 'snap' to enclose the nearest musical item on the staff. Once an area has been swept, there are various functions available which affect only the contents of that area: for examples, see *Editing and using the clipboard* [20], *Grouping* [22] and *Marking chords* [25].

You can also sweep out a whole bar, by double-clicking rapidly on a point within that bar; and you can select a whole staff by clicking three times rapidly on a point within it.

2.2.1 Editing and using the clipboard

The *Editor Edit* [37] menu provides several general functions for editing music. Several of these require that an area has been swept out or that the editing cursor is placed (see *Using the editing cursor* [19]).

There are three general editing functions: *Editor Edit - Undo* [37], which reverses the effect of the last change you made to the music buffer; *Editor Edit - Redo* [37], which undoes the last Undo; and *Editor Edit - Delete* [38], which deletes all items enclosed in the area swept out on the main window.

There are four functions for manipulating the editor's clipboard. The clipboard is a tool for temporarily storing sections of music: they can be taken from the main window and stored in the clipboard until (or in case) they are needed later. The clipboard can only store one clip of music at a time, and it can only be from one staff, although there is no limit on its length. To store something in the clipboard, sweep out an area on the main window and then use *Editor Edit - Cut* [38], if you want it also to be removed from the main window, or *Editor Edit - Copy* [38] if you want a new copy to be made for the clipboard and the swept section of music also to stay in place in the main window. Either way you will lose anything which may previously have been held in the clipboard.

The contents of the clipboard can then be inserted into the music at a different point: to do so, position the cursor on a staff somewhere, and use *Editor Edit - Paste* [38] . The clipboard's contents will then be inserted into the music at the cursor position. You can view the clipboard's contents at any time without using *Editor Edit - Paste* [38] by using the *Editor Edit - Show Clipboard* [38] function.

2.2.1 Chords

There are two basic ways of creating chords, both found on the *Chord* [38] menu. The *Chord - Create Chord* [39] function takes a number of distinct notes --- or existing chords --- and merges them into one chord. This is the most basic way of making a chord: you just place all the notes you want to use, one after the other, onto the staff; and then you select all the notes and use this function. Alternatively, if you want to place a particular chord for which you already know the name onto the staff, you can use the *Chord - Create by Name* [39] function, which allows you to specify a guitar-notation chord name, and then places the named chord onto the staff at the position of the cursor.

Having created a chord, you can then modify the arrangement of notes in it using *Chord - Edit Chord* [40] , which pops up a handy dialogue box with a range of buttons that change things about the chord.

In addition to making chords based on their notational names, the Rosegarden Editor can provide names based on the chords within the stave itself. The *Chord - Guess Chord Names* [41] function considers all the chords in the selected area, and one by one attempts to work out what each chord is. It then writes the name for each chord above that chord. *Chord - Clear Chord Names* [41] can be used to remove these names. (Of course, Rosegarden will not always get the names right, and in any case chords can be indefinitely ambiguous. Future versions will no doubt allow you to decide you know better and to provide some chord names yourself, but in this version the only way to do so is to create the chords using the *Chord - Create by Name* [39] function in the first place.)

For further control over how and where Rosegarden draws the chord names, you can use the four functions *Chord - Label No Chords* [41] , *Chord - Label Named Chords Only* [42] , *Chord - Label All Chords* [42] and *Chord - Label Chords and Notes* [42] . These declare, respectively, that you want to see no labels at all; that you want any chords on which you have used the *Chord - Guess Chord Names* [41] function to be labelled, and no others; that you want every chord to be labelled; and that you want not only chords but also individual notes to be labelled. Individual notes will be labelled in lower-case, whereas chords are always named with an initial capital letter.

Finally, note that it is *not* possible to create chords in which the individual notes are of various different lengths. Each note in a chord must be of the same length. There aren't any provisions in Rosegarden for writing staves which are contrapuntal within any one staff, and this does slightly limit the useful possibilities for entering musical staves.

2.2.1 Grouping

The Rosegarden editor supports two kinds of groups: beamed groups and triplet (and similar time-domain-perverted) groups.

To group together a series of notes, contain them in a swept area (see *Using the editing cursor* [19]) and then select the *Group - Beam* [45], *Group - Simple Tuplet* [45] or *Group - Tuplet* [46] function from the *Group* [44] menu. To ungroup a series of notes which contains some grouped notes, enclose them in a swept area and select the *Group - Break Group* [47] function. A swept-out series of items can be grouped even if they are not all chords, although in order for the *Group - Beam* [45] function to have any effect, the series must contain at least two chords.

If you have a long series of notes which you would like divided into beamed groups without having them all beamed together into one group, you can use the *Group - Auto Beam* [47] function. This finds likely-looking sequences of notes within the selected area, and forms them into beamed groups. See also *Importing MIDI files* [25] (where the same technique is used) for some discussion of how this process can go wrong.

The *Group* [44] menu also provides access to some other functions for expressing relations between notes.

Dynamic markings can be added across groups of items, using the *Group - Slur* [44], *Group - Crescendo* [45] and *Group - Decrescendo* [45] functions, which decorate a selected series of items with a slur (or tie, or phrase) marking, or with opening or closing hairpin dynamic markings respectively. The latter three functions have yet to be implemented.

2.2.1 Operations on staffs

The *Stave* [42] menu provides several functions for manipulating the display and content of the staffs in a piece, and how they relate to each other.

To add staffs to the score or remove existing staffs, use the *Stave - Add Staff* [42] and *Stave - Delete Staff* [43] functions.

Adding a staff requires that you choose where the new staff is to go in the piece: a box is popped up to offer you a choice of at top (first), at bottom (last), or, if a staff is selected by having the cursor positioned in it, after the selected staff. The empty stub of the new staff will be placed where requested, with default clef and time signature markings already in place.

Deleting a staff requires that there is a staff selected with the cursor. As deleting a whole staff causes its entire contents to be lost, a box will be popped up to ask you to confirm the deletion before it is carried out.

You can also change the displayed name of a staff with the *Stave - Rename Staff* [43] function.

Often one staff within a piece of music may end before the others. If you object to having a gaping hole in the latter part of the score where a staff has ended, you can use the *Stave - Fill to End* [44] function, which fills in the selected staff with rests until it is the same length as the longest staff in the piece.

2.2.1 Changing bar properties

The *Bar* [48] menu contains several functions for changing the meaning of bar lines, and any instructional indications within the bar (for which see *Tempos and times* [24]). To use any of these functions, there must be a bar selected by having the cursor placed within it.

The *Bar - Plain* [48], *Bar - Double Bar* [48], *Bar - Repeat at Start* [49] and *Bar - Repeat at End* [49] functions are used to change the type of bar line the selected bar has. *Bar - Plain* [48] resets it to having single bar lines at both beginning and end; the others give it a double bar line at the end, a dotted double bar line at the beginning and a dotted double bar line at the end, respectively.

Normally, the Rosegarden editor reformats the music it contains as it is entered; it is free to move bar line about as it wishes to ensure that all the bars are of the same length. It will however usually leave double-bars and repeat bars alone (see *Bar - Double Bar* [48] for details of exactly how it handles them), so if you want to make sure later sections of music aren't moved around all over the place just because you're editing an earlier bit, try putting a double bar line in somewhere.

2.2.1 Tempos and times

The *Bar* [48] menu provides some functions for adding, removing and changing meta musical indications such as tempo and time markings.

The *Bar - Label* [49] , *Bar - Metronome* [50] and *Bar - Time Signature* [50] functions change the label, metronome marking for the bar and time signature. *Bar - Label* [49] prompts you for a textual label, and then if there is no label already present on the bar, it adds the new label above the opening bar line; otherwise it replaces the old label with the new one. Similarly, *Bar - Metronome* [50] allows you to enter a new tempo marking and replaces any existing metronome marking with the it --- there can only be one tempo marking in a given bar, in whichever staff of the piece • , and *Bar - Time Signature* [50] allows you to replace any existing time signature marking in this bar, or if you wish in the bar at this point in every staff, with a new one.

To remove a label or metronome marking from a bar, use the *Bar - Remove Label* [49] and *Bar - Remove Metronome* [50] functions.

See *Changing bar properties* [23] for other uses of the *Bar* [48] menu.

2.2.1 Adding text

The Rosegarden editor allows descriptive and decorative text to be added to the music in four different styles (in addition to bar labels, for which see *Tempos and times* [24]). To add text, place the cursor on a staff and then use the *Text* [51] menu.

The four functions at the top of the *Text* [51] menu, the *Text - Add Text options* [51] , all work in the same way: by prompting you for some text and then adding it to the music above or below the position of the cursor. The difference is in the placement and style of the text, and the menu entries contain small icons to try to remind you of the purpose of each one.

The text styles are as follows:

- above the staff, in a smallish font;
- a considerable distance above the staff, in a larger font;
- below the staff, in a smallish font; and
- below the staff, in a smallish font and in italics.

You can also add ‘dynamic’ textual markings, using the *Text - Dynamic* [51] function. These are simple indications of force, from ‘pppp’ to ‘ffff’. They are shown below the staff, but they differ from the usual below-staff markings in that they may be taken into account when creating MIDI output (either by exporting to a file with *Editor File - Export MIDI* [35] or to the Sequencer with *Editor File - Sequence* [36]) to calculate likely amplitude values. The calculation of MIDI dynamics is rather experimental and is switched off by default, but can be activated by setting the 'midiDynamics' resource to True in the supplied Rosegarden resource file. But even if you don't use the dynamic calculations themselves, it probably makes sense for any simple markings of this sort to be written using the 'dynamic' text-style.

Text can also be removed using the *Text* [51] menu; the *Text - Clear Group* [51] , *Text - Clear Bar* [51] and *Text - Clear Staff* [52] functions remove all text found within a swept-out area, a bar selected by cursor placement or a staff selected by cursor placement, respectively.

2.2.1 Marking chords

The Rosegarden editor supports several types of dynamic markings and decorations for chords. Some of these, such as slurs and (de)crescendo marks, may extend over series of musical items; these are described in *Grouping* [22] . Others, such as accents, affect only single chords (although they can be added to several chords at one go). This sort of marking is found on the *Marks* [52] menu.

To add marks to chords, first sweep out an area on the stave (see *Using the editing cursor* [19]). (A mark-adding operation will affect every chord in the swept area, but won't affect any non-chord items found in the area.) Then call the *Marks* [52] menu and select the relevant function for the type of mark you need, and that mark will be placed over your selected chords.

To remove marks, sweep out an area and then use the 'Unmark' function on the *Marks* [52] menu. Any marks present on any chords within the swept area will be deleted.

2.2.1 Importing MIDI files

The Rosegarden editor includes powerful (but by no means infallible) facilities for reading and generating MIDI files. These are available from the *Editor File* [33] menu.

To read in a MIDI file and display it in musical notation for subsequent editing, use the *Editor File - Import MIDI* [34] function. This prompts you for the name of a MIDI file to open (see *File Boxes* [10]), prompts for quantization levels in the same way as the Sequencer's *Track - Quantize* [61] function, and reads it in as a new buffer. There is no disc file associated with the new buffer, so in order to save it as a new editor file you must use the *Editor File - Save As* [34] function, not *Editor File - Save* [34] .

The *Editor File - Import MIDI* [34] function can only deal with MIDI Type 0 (single track) and Type 1 (simultaneous track) files; if you have a Type 2 (sequential track) file, use *The Rosegarden Sequencer* [15] to divide it into separate files before editing. (See *Organising recorded data* [29] for help in managing different types of files in the sequencer.) MIDI Type 1 files often reserve the first track for file meta-events such as key and tempo events; the editor checks for this, and if it finds no actual notes in the first track, that track will not be displayed (although it will still be processed to obtain meta-event information for the other tracks).

The editor attempts to maintain the correct clef, key and tempo markings for the MIDI file, to arrange its note pitches and lengths sensibly and to beam together groups of quavers and shorter notes in a likely way. When it gets something wrong, it is likely to be for one of these two reasons:

- The MIDI file is quantized to too short a note length. In this case, a note which should fill a certain length (say for example a quaver) may be interpreted a a shorter note plus a rest (such as a dotted semiquaver and a demisemiquaver rest) because it has been released too soon. This can ruin the layout of the piece more effectively than you might imagine. Quantizing can be a tricky process: if the file does actually contain some genuinely short notes, quantizing may lose them; and forms such as triplets may be lost.

Whilst the Rosegarden Editor can quantize MIDI files on import, there are more possibilities available if you use *The Rosegarden Sequencer* [15] to inspect and quantize files before trying to import them into the editor: see *Organising recorded data* [29] and *Preparing a MIDI file for the Editor* [31] for help on quantizing in the sequencer and transferring files directly from the sequencer to the editor.

- The MIDI file contains the wrong time signature for the piece. As compound times (6/8, 9/16 and so on) are normally beamed in groups of multiples of three notes, and simple times in groups of multiples of two, a file which gives the wrong time signature can be wrongly beamed. This often happens where the file has the default signature of 4/4, but should in fact be in 6/8 or some other such time.
- The MIDI file contains triplets or other stretched or squashed series of notes. There is no portable way to represent such forms in MIDI, and the editor isn't clever enough to know when the note lengths should suggest that triplets were intended.

• The beaming algorithm employed by the editor has failed. This can often happen in conjunction with quantizing problems, or when the piece is in an unusual time signature; usually it means that the beamer has become out of phase with the beat of the music, and can no longer find any likely-looking groups to beam. There is very little that you can do about this, except ensure that the file is properly quantized and be prepared to do some editing after it has been imported.

2.2.1 Exporting MIDI files

The editor is capable of producing MIDI Type 1 (simultaneous track) files from the music in its buffers. MIDI files can be used as a standard musical interchange format for sequencers and synthesisers; they are also accepted by some other notation programs. Using MIDI files to store music intended for further editing using the Rosegarden editor later is not recommended because, although they take considerably less disc space than Rosegarden editor files, they cannot store all the necessary information on grouping, marking, dynamics, text and so on.

To write the current buffer out to a file in MIDI format, use the *Editor File - Export MIDI* [35] function on the *Editor File* [33] menu. You will be prompted for a file in which to put the MIDI output (see *File Boxes* [10]), and the buffer will be immediately written out to that file as MIDI. The contents of the buffer in the editor will be unaffected.

For details of how to send MIDI data directly to *The Rosegarden Sequencer* [15] without having to write them to a file in between, see *Sequencing a piece of music* [27].

2.2.1 Sequencing a piece of music

The *Editor File - Sequence* [36] function on the *Editor File* [33] menu allows you to sequence the MIDI representation of a musical buffer immediately, without having to export it to a file and load it into a sequencer later. Selecting the function causes the contents of the current buffer to be sent to *The Rosegarden Sequencer* [15] in MIDI Type 1 format. If the sequencer is already running, it will be sent the data and will raise its window to the front of the display when ready; otherwise, it will be started up automatically.

This function requires that the complete Rosegarden suite is running: the editor must have been started up from *The Rosegarden Top Box* [9], and the sequencer must be either already running, or available for execution.

2.2.1 Exporting MusicTeX files

MusicTeX is a music typesetting language which consists of a set of macros working on top of the TeX typesetting language. In order to make use of MusicTeX files such as those produced by the Rosegarden editor, your system must be equipped with the TeX program, the MusicTeX package and a suitable set of fonts for printing music and drivers for sending the output to your printer.

To produce MusicTeX output, select the *Editor File - Export MusicTeX* [35] function and choose a file in which to save the output (see *File Boxes* [10]). The current buffer will be processed and a MusicTeX transcription of it placed in the selected file.

Rosegarden can also produce OpusTeX output (see *Editor File - Export OpusTeX* [36]), and the PMX preprocessor format for MusiXTeX (see *Editor File - Export PMX* [36]); and MusicTeX files can also be read by the MusiXTeX system.

2.2.1 Converting Rosegarden 2.0 files

Version 2.1 of the Rosegarden Editor can load files that were saved from version 2.0, although the converse is not true (files saved with 2.1 will not load into 2.0).

When loading a file from version 2.0 into this version of the Editor, you may be asked something like, ‘‘there are 34 textual markings in this file that look as though they could be safely made into style ‘dynamic’. Should I convert them?’’

This question arises because version 2.1 introduces a new text style -- ‘dynamic’ -- which was not available in 2.0. (See *Adding text* [24] for details of how to enter text with this style.) Text entered in style ‘dynamic’ may be taken into consideration by the Editor when it produces MIDI output, either to a file or to the Sequencer, depending on your setting of the ‘midiDynamics’ resource in the resource file.

If you choose to convert textual markings into dynamics where appropriate, then the Editor will simply scan the newly loaded buffer for textual marks that appear below the staff (the ‘Below Staff’ and ‘Below Staff, Italic’ styles) and that appear to contain only simple volume labels such as ‘pp’ or ‘mf’; any such marks found will just have their style changed to ‘Dynamic’. (The disc file will be untouched until next time you save it.)

Incidentally, if you really want to load Rosegarden 2.1 files into version 2.0, it is possible to change them by hand, as the files are written in ASCII. The only differences are the version tag at the top of the file; the addition of ‘connection’ markers for each Stave at the end of the file; and possibly the use of the new ‘chord name’ and ‘dynamic’ text styles (positions 5 and 6 in the file format).

2.3 Sequencer

2.3.1 Loading a MIDI file

To load a new MIDI file select *Sequencer File - Open* [53]. You will then be presented with a file dialogue box. Select the required file from the list presented and press the *Load* button. The file will now load.

2.3.1 Saving a MIDI file

To save the current MIDI file, select *Sequencer File - Save* [53]. If the file has not yet been saved you will be prompted to supply a name for the new file. If the file has been saved before (or was loaded in via the *Sequencer File - Open* [53] option) then the new file will simply be saved over the top of the old file. To save the file under a different name, use the *Sequencer File - Save As* [53] option.

2.3.1 Recording MIDI data

The Rosegarden Sequencer [15] allows you to record data from attached MIDI devices in real-time. To record data select the *Sequencer File - Record* [53] option. If there is a MIDI file in the sequencer, it will begin to play, and at the same time the sequencer will listen for MIDI events on the record device (see *Midi - MIDI Setup* [62]). Any MIDI events received over the MIDI link will now be stored in memory and returned to you as a new un-named track appended to the list of tracks in the main window after you have stopped recording by clicking on the *Sequencer File - Stop* button. This data can then be manipulated and viewed using the standard track editing functions. Any unwanted events can be filtered out using the *Track - Filter By Channel* [60] and *Track - Filter By Event* [60] options. See *Organising recorded data* [29] for more details on the editing facilities that are available.

2.3.1 Organising recorded data

One of the main functions of the sequencer is to allow the user to reformat MIDI data in a more organised manner than its raw recorded form. The sequencer allows MIDI data to be separated into tracks, each of which contains information for a sub-section of the music. Tracks in the sequencer are played back simultaneously, and can be viewed as equivalent to the tracks on a multi-track recording device. Although the MIDI file format defines a form for sequential track files this format is not currently supported by the sequencer.

Usually tracks contain data for a single instrument, (although when transcribing music for an instrument such as the piano it may be desirable to separate the events into left- and right-hand parts). Also the MIDI file specification states that in multi-track files, Track 0 should contain the tempo map and key and time signature events, and that these events should not appear in other tracks in the file. The Rosegarden sequencer does not enforce this organisation, but in order to ensure that MIDI files produced by Rosegarden are readable by other applications the user should attempt to conform to this format.

To enable the user to re-organise data in a MIDI file the Rosegarden sequencer provides a number of track editing functions. The principle concept is the use of *filters* to strip out unwanted events in the MIDI track, along with *cloning*, which is used to create an exact copy of a track, and *merging*, which can be used to combine two separate tracks.

Filter functions are available from the *Track* menus on the main windows and piano roll and event list sub-windows. The filter are as follows:

Track - Filter By Channel [60]. This filter allows you to strip out all the events from a MIDI track, retaining only those events on the specified channel. This filter is particularly useful for separating tracks by instrument.

Track - Filter By Event [60]. This filter strips out all the events in a track which are not of the type or types specified. Use this filter to help with creating Track 0, or to remove unwanted events, such as text meta-events which are no longer meaningful in the edited track.

Track - Filter By Pitch [60]. This allows you to strip out any events from a track which fall above or below a specified pitch. This filter can be used (for instance) to crudely split keyboard lines into left- and right-hand parts before passing to the Notation editor to be turned into music score.

In addition to the event filter tracks can be transposed to a different key using the *Track - Transpose* [61] function, and quantized to make the playing more regular, using the *Track - Quantize* [61] option. Events may also be moved from one channel to another for a given track by use of the *Track - Change Channel* [60] facility.

2.3.1 Displaying a track

Tracks can be displayed in two formats. The simplest form is the event list. This is simply a list of all the MIDI events in the selected track, with some textual description of their parameters, and is useful for obtained detailed information about the events in a MIDI track. All the events in the track are displayed in the list. To display a track as an event list select the track from the track list in the main window and the select the *Track - Show Event List* [58] menu option.

Alternatively tracks may be displayed using the 'Piano Roll' format. This provides a graphical representation of the MIDI track, displaying note events as rectangular boxes over a vertical keyboard staff. Only note events are displayed on the piano roll. To display a track as a piano roll select the track from the track list in the main window and then select the *Track - Show Piano Roll* [58] menu option.

Both piano roll and event list windows have menu options to call up the complementary format for displaying the track in the window. To call up the piano roll window from an event list window simply select *Track - Show Piano Roll* [58], and similarly to call up an event list window from a piano roll, select *Track - Show Event List* [58].

Double clicking on the Track in the Track List window will pop-up the Event List if it is not shown, a second double click will bring up the Piano Roll view for that track and subsequent double clicks will bring the exiting Event List to the front.

See *Viewing with the Piano Roll* [31] and *Editing with the Event List* [30].

2.3.1 Editing with the Event List

The event list sub-windows provide you with a detailed list of all the events present on a MIDI track. For a description of the display format for event list windows, see *Track - Show Event List* [58].

From an event list window you have access to all the track manipulation functions, plus the ability to use the clipboard to cut and paste events. To cut, copy or delete events outline the area to be edited by clicking at the start event and dragging the cursor down to the end event. The event list can be scrolled while doing this by moving the cursor outside of the window bounds. Once you have selected the required events select the *Cut*, *Copy* or *Delete* options from the *Edit* menu as required.

To paste events into a track using the event list window click on the event in the event list before which you wish to have the events inserted and select *Paste*. If more than one event is selected then the events from the clipboard will be pasted before the start event in the selection.

See *Sequencer Edit - Cut* [56] , *Sequencer Edit - Copy* [56] , *Sequencer Edit - Paste* [56] and *Sequencer Edit - Delete* [56] for further information on the clipboard functions.

To modify an event in the event list by changing one or more of its parameters select *Event - Modify Event* [64] . A dialogue box will be called up containing the parameters for the chosen event and allowing you to alter them. If a number of events are selected then a dialogue box will only be produced for the first event in the selection. *Event - Modify Event* [64] can also be performed by double clicking on a single event in the list.

In addition to manipulating existing functions new events can be created using the options on the *Event* [64] menu. If you have selected an event in the event list before selecting one of these options then the dialogue box that is called up will contain the time of the selected event in the time field, and the event will be inserted just before the selected event after you have clicked on *OK*. You are still free to change this time value if you wish. For more information on the specific dialogue boxes for each event type see the following sections,

2.3.1 Viewing with the Piano Roll

The Piano Roll does not currently offer any editing facilites, it is purely a graphical display of the events on the track. The display can be zoomed using the '+' and '-' buttons on the bottom of the window.

2.3.1 Preparing a MIDI file for the Editor

In order to get the best results when using the Rosegarden Editor to produce a transcription of a MIDI file it is necessary to first tidy up the MIDI data in a number of ways.

Firstly the file should be quantized to a reasonable resolution with respect to both note timing and duration.In general a fairly coarse granularity should be used to produce a tidy transcription, however the coarser the resolution the more information is lost when transcribing. Optimum values will of course vary from piece to piece, but as a rough guide most pieces of music transcribe fairly well using semi-quaver resolution for both note position and duration. See *Track - Quantize* [61] for details on how to quantize MIDI data.

Secondly each track in the MIDI file is mapped to a separate staff when transcribing. It should be ensured that the MIDI file only contains a single instrument in each track. Since each MIDI instrument must reside on a separate MIDI channel this can be achieved by using the *Track - Filter By Channel* [60] function.

Although splitting the file into tracks solely by instrument works fine for some pieces, often music for keyboard instruments is best transcribed as separate staves for the left hand and right hand parts of the music. Splitting MIDI files in this manner is a more difficult task. The basic procedure that can be used is to first split the track at some designated pitch by cloning it using *Track - Clone* [59] and then filtering the two resulting tracks using *Track - Filter By Pitch* [60], so that one track contains notes above the designated pitch and the other contains the notes below this value. This provides a rough first approximation of the left and right hand parts, which can then be hand edited by using the cut and paste options to transfer anomalous notes between the two tracks.

Finally a number of useful meta-events can be inserted into Track 0 of the file, which will then be used by the Rosegarden Editor to aid with the transcription process. Tempo events and time and key signature events will all be interpreted correctly and placed at the relevant point in the transcription, and will affect the format of the transcribed music. If these events are not inserted into the MIDI file they can always be added in using the editor after transcription, but the resulting beaming may differ.

See the Rosegarden Editor page for *Importing MIDI files* [25] for a discussion on the Editor's approach to automatic beaming. See *Organising recorded data* [29] for more detailed descriptions of how to set about re-organising raw MIDI data.

2.3.1 Playing a MIDI file

To play back a MIDI file in real-time over a MIDI link select the *Sequencer File - Play* [54] menu option (although you should probably first check that the sequencer is correctly configured for your system: see *Midi - MIDI Setup* [62].

Playback will commence immediately and may be stopped by clicking on the displayed *STOP* button. If you stop and start the playback, it will continue from the point it was stopped at rather than going back to the beginning: you can use the fast-forward and rewind buttons (see *Sequencer File - Return to Start* [53], *Sequencer File - Rewind* [54], *Sequencer File - Fast Forward* [54] and *Sequencer File - Skip to End* [54]) to reposition the point of playback, both during playback and while it is stopped.

If playback sounds garbled then check that your MIDI devices are configured correctly. If necessary you can modify the events in the MIDI file to enable correct playback - most commonly this involves using *Track - Change Channel* [60] to alter the channel that these events are transmitted on.

2.3.1 Generating Csound

Rosegarden can translate MIDI files to part of a Csound score file. The format is straightforward, with each staff treated as a separate instrument, and MIDI notes translated into the standard equal-tempered Csound pitches, and the MIDI velocity given as the fifth parameter.

The translation if achieved via the *Sequencer File - Export Csound* [55] option.

3. Reference Manual

3.1 Editor

3.1.1 Editor File

The File menu on the Rosegarden Editor contains functions for editing new buffers, loading, saving and printing buffers and exiting the application. The File menu also lists the names of all buffers which have been opened since the editor was started and which have not been closed since. Selecting one of these names switches to editing that buffer. Notice that the entire contents of every file you open in the editor are held in memory until its closure (see *Editor File - Close* [34]); opening many large files may affect the performance of the editor or of the computer. This menu also contains functions for less basic file operations such as manipulating MIDI or MusicTeX files. See also *Starting a new piece of music* [15] , *Opening an editor file* [16] and *Saving an editor file* [16] for details of using the basic file operations; *Importing MIDI files* [25] , *Exporting MIDI files* [26] , *Sequencing a piece of music* [27] and *The Rosegarden Sequencer* [15] for more description of the MIDI-handling functions; and *Exporting MusicTeX files* [27] for a comment about producing a first draft of output to prepare for typesetting.

3.1.1 Editor File - New

This function is part of the *Editor File* [33] menu.

Creates an empty piece of music for editing, first asking you how many staffs you need in the score. Initialises the staff names, clefs and time signatures to a default set, all of which may be changed afterwards.

See *Starting a new piece of music* [15] .

3.1.1 Editor File - Open

This function is part of the *Editor File* [33] menu.

Opens a previously saved file for editing. See also *Opening an editor file* [16].

For a description of how the file selection box works, see *File Boxes* [10].

Note that the file format used by Rosegarden changed between the 2.0 beta and 2.0.1 releases. This version no longer supports the 'old' format. If you want to convert files saved with 2.0-beta or earlier, get a copy of 2.0.1, load the files using the "Import Old Rosegarden File" function, and save them as new files. The file-format changes were quite severe and important, and I hope the format won't be changing in an incompatible way again at any time soon.

3.1.1 Editor File - Save

This function is part of the *Editor File* [33] menu.

Saves the buffer currently being edited into its associated disc file. This is only possible if it was originally loaded using *Editor File - Open* [34], or if it has previously been saved by use of *Editor File - Save As* [34]). See also *Saving an editor file* [16].

3.1.1 Editor File - Save As

This function is part of the *Editor File* [33] menu.

Saves the buffer currently being edited into a new file, prompting you for a file name. See *Saving an editor file* [16].

For a description of how the file selection box works, see *File Boxes* [10].

3.1.1 Editor File - Close

This function is part of the *Editor File* [33] menu.

Closes and removes the buffer currently being edited, first giving you the option of saving any changes you may have made to it.

3.1.1 Editor File - Import MIDI

This function is part of the *Editor File* [33] menu.

Prompts for a MIDI file to be imported. The editor will read the file, check whether you'd like it to be quantized, convert it to Rosegarden Editor format with one staff for each MIDI track (excepting track zero in the case where it contains no actual notes), beam chords into groups automatically, choose apparently suitable clefs for each staff and display the music including any textual, key or tempo information found in the file.

Once imported, the buffer acts as a normal editor buffer: it can be edited as usual, and saved, exported or sequenced. As the editor has no filename for the converted file, it cannot be saved using the *Editor File - Save* [34] option, but only using *Editor File - Save As* [34] .

See also *Importing MIDI files* [25] .

3.1.1 Editor File - Export MIDI

This function is part of the *Editor File* [33] menu.

Prompts for a filename and saves the current buffer into that file in MIDI format. MIDI files thus produced should be compatible with any reputable sequencing software, including *The Rosegarden Sequencer* [15] .

MIDI files are exported including all information pertinent to their playback. However, they should not be used when storing score files which you have created using the Rosegarden editor, because the MIDI format cannot record much of the information present in a Rosegarden editor file; use *Editor File - Save* [34] to create suitable files instead.

By default, this version of Rosegarden does not write dynamic information (i.e. adjustments to MIDI amplitude values) out to the MIDI file. There is some experimental code for computing dynamics, though, and you can elect to use it by setting the 'midiDynamics' X resource to True in the Rosegarden resource file before you start up the Editor. If you do so, the Editor will attempt to take into account both textual markings (see *Text - Dynamic* [51]) and hairpin marks (see *Group - Crescendo* [45] and *Group - Decrescendo* [45]) when writing MIDI to a file or to the sequencer. If you have some staffs connected together as a single instrument (see *Stave - Bracket with Next* [44]), the dynamics for the topmost staff will be applied to all staffs with which it is bracketed, and any other dynamics on the other staffs will be discarded. (This situation will probably change in future releases.)

See also *Exporting MIDI files* [26] .

3.1.1 Editor File - Export MusicTeX

This function is part of the *Editor File* [33] menu.

Prompts for a filename and saves the current buffer into that file in MusicTeX format. Files thus produced can be processed using the TeX typesetting language with the MusicTeX music typesetting macro package to produce high-quality printed output.

If you intend to make use of this function, you should make sure you're familiar with TeX and MusicTeX first. The output Rosegarden produces is likely to be incomplete and imperfect and will probably need editing by hand before it will process and print well. MusicTeX can be very challenging to use, and documenting it is considerably outside the scope of this help file.

(A brief note, however: Rosegarden includes an ''autolines'' declaration in the file, but with arbitrary values, and by default it leaves the declaration commented out. No other attempt is made to fix correct line breaks, so you will probably want to uncomment this declaration and insert some better values. If you find that TeX runs out of memory or gets gradually slower and slower, you'll probably need to uncomment the autolines declaration.)

See also *Exporting MusicTeX files* [27] , and also *Editor File - Export OpusTeX* [36] and *Editor File - Export PMX* [36] .

3.1.1 Editor File - Export OpusTeX

This function is part of the *Editor File* [33] menu.

Prompts for a filename and saves the current buffer into that file in OpusTeX format. Files thus produced can be processed using the TeX typesetting language with the OpusTeX music typesetting macro package to produce high-quality printed output. See also *Editor File - Export MusicTeX* [35] and *Editor File - Export PMX* [36] .

OpusTeX output is even more experimental, and probably even more unreliable, than MusicTeX output, not least because OpusTeX has (at the time of writing) not yet reached its first official release and so is liable to change. Rosegarden's OpusTeX output code is based on the MusicTeX output, with modifications supplied by Erwin Achermann, and like the MusicTeX output module it could really do with a complete rewrite. This notwithstanding, it is likely to get more active development work than the MusicTeX output option.

3.1.1 Editor File - Export PMX

This function is part of the *Editor File* [33] menu.

Prompts for a filename and saves the current buffer into that file in PMX format. Files thus produced can be processed using the PMX preprocessor, followed by the TeX typesetting language with the MusiXTeX music typesetting macro package to produce high-quality printed output. See also *Editor File - Export MusicTeX* [35] and *Editor File - Export OpusTeX* [36] .

Of the three TeX output formats, this currently produces the best output in most cases because the PMX input format is the easiest for Rosegarden to write and PMX does a lot of the layout work. It's not ideal, because of some limitations of the Rosegarden code and a few limitations of PMX, but at the moment it's normally the best option. Of course, it requires that you have PMX available as well as MusiXTeX.

3.1.1 Editor File - Sequence

This function is part of the *Editor File* [33] menu.

Sends a MIDI representation of the music in the current buffer to *The Rosegarden Sequencer* [15] for sequencing. If the sequencer is not yet running, it will be started. This function is only available when the full Rosegarden Suite is in operation, including *The Rosegarden Top Box* [9] .

See also *Sequencing a piece of music* [27] .

3.1.1 Editor File - Exit

This function is part of the *Editor File* [33] menu.

Exits the application, offering the option of saving any changed buffers first.

3.1.1 Editor Edit

The Edit menu yields functions for manipulating the clipboard (cut, copy and paste), plus a couple of other general editing functions. See *Editing and using the clipboard* [20] for an overview of the Edit menu.

3.1.1 Editor Edit - Undo

This function is part of the *Editor Edit* [37] menu.

Undoes the effect of the last change to the buffer. This enables you to save your sanity if you do something really stupid by accident. (Some operations can't be undone, but those ones will normally warn you before you try to do them. In general, any operation that acts on more than one staff at once, or that changes the number of staffs in the piece, cannot be undone. The undo operation is also unable to restore any slurs, tie or dynamic hairpin marks that may have been lost.)

The editor tries to keep track of multiple changes to the buffer, and to allow you to Undo more than one. (In theory, if you only carried out simple operations, you would sometimes be able to Undo your entire life's work right back to the empty buffer you started with. In practice, you can't, because the editor deliberately only remembers at most twenty undoable changes -- each change could involve an arbitrarily large amount of work, and remembering very many could waste an awful lot of memory.)

3.1.1 Editor Edit - Redo

This function is part of the *Editor Edit* [37] menu.

If you have just used *Editor Edit - Undo* [37] to undo a change to the buffer, this function will allow you to restore that change --- to Undo the Undo --- provided you have not made any other changes since using the Undo function.

You can use the Undo and Redo functions to step backwards and forwards through the last sequence of simple changes you made to the current buffer.

3.1.1 Editor Edit - Delete

This function is part of the *Editor Edit* [37] menu.

Deletes the contents of the selected area.

3.1.1 Editor Edit - Cut

This function is part of the *Editor Edit* [37] menu.

Deletes the contents of the selected area, moving them into the clipboard. The previous contents of the clipboard, if any, are lost.

3.1.1 Editor Edit - Copy

This function is part of the *Editor Edit* [37] menu.

Copies the contents of the selected area to the clipboard. The previous contents of the clipboard, if any, are lost.

3.1.1 Editor Edit - Paste

This function is part of the *Editor Edit* [37] menu.

Inserts the contents of the clipboard into the current buffer at the cursor position. The clipboard itself is not changed.

3.1.1 Editor Edit - Show Clipboard

This function is part of the *Editor Edit* [37] menu. Pops up a small window showing the contents of the clipboard.

3.1.1 Editor Edit - Select this Bar

This function is part of the *Editor Edit* [37] menu.

Causes the bar containing the Edit cursor to be selected. Clicking on a point within the stave, then using this function, usually has the same effect as double-clicking on that point. See also *Using the editing cursor* [19] .

3.1.1 Editor Edit - Select this Staff

This function is part of the *Editor Edit* [37] menu.

Causes the whole staff containing the Edit cursor to be selected. Clicking on a point within the stave, then using this function, usually has the same effect as triple-clicking on that point. See also *Using the editing cursor* [19] .

3.1.1 Chord

The Chord menu contains functions for creating chords from individual notes or explicitly by name, and for changing chords and labelling them with notational names. See *Chords* [21] for a description of how to use some of the functions on the Chord menu.

3.1.1 Chord - Create Chord

This function is part of the *Chord* [38] menu.

Transforms the chords within the selected area into a single chord. This chord will contain a note of every pitch found in the selected area, and its length will be that of the last chord in the area.

This function is more-or-less the reverse of the *Chord - Break Chord* [42] function.

3.1.1 Chord - Create by Name

This function is part of the *Chord* [38] menu. It allows you to make chords by describing them in guitar chord notation.

The Create Chord dialog will accept one or several comma-separated chord names in standard ''Jazz'' notation (e.g. C, Gm, Amaj7, E#9) and insert them on the staff at the point of the current editing cursor. The chord it creates will be of crotchet length, with the root note in an arbitrary but sensible octave; both of these things can be modified afterwards with the *Chord - Edit Chord* [40] function.

Rosegarden is pretty tolerant about the ''syntax'' used to name the chord you wish to insert, and should recognize most of the chord names you can find in common music scores.

Here are a few rules and examples for the sake of clarity, which you should recognize as those you're already familiar with, give or take a few restrictions inherent to the ASCII nature of the thing:

- The root and the altered bass (if any) should always be specified with a capital letter.
- The altered bass should be specified with a leading slash character ('/'), e.g. C/G (C major with a G bass).
- Minor chords should be named with an lower case 'm', not a minus sign: Cm.
- Power chords (root, 5th, octave) should be noted with a trailing 5: C5
- Chord complexity can go up to the 13th: C7, C9, C11, C13.
- Suspended, diminished and augmented chords should be noted as follows, respectively: Csus (i.e. Csus4), Csus2, Cdim (Cdim7 should work), Caug (i.e. C+).
- Major seventh chord should be named as Cmaj7.

- Added 9th, 11th and 13th should be noted as Cadd9, Cadd11 and so on.
- Altered notes should be specified with leading '-' ('b') or '+' ('#') signs: C-5, C7-5, C7#9, C6b11, C13+9. Only the 5th, 9th, 11th and 13th can be specified as being altered this way.

Note that a '#' or 'b' right after the root will always be treated as altering the root itself and not the following note. So Ab5 is an Ab power chord, not an A major with a flattened fifth. Use '+' or '-' to express this: A-5.

This function and its help text were written and contributed by Guillaume Laurent.

3.1.1 Chord - Edit Chord

This function is part of the *Chord* [38] menu.

Invokes a window containing functions for making changes to various properties of the selected chord.

The chord edit window is split into three parts:

- The chord display, plus four directional buttons and four other command buttons. The display itself shows the chord and the clef it's in on the stave. One of the note heads of the chord (if it has more than one head) is indicated with an arrow. Clicking the up and down buttons, above and below the chord display, moves the pitch of the indicated note head up or down by one semitone respectively. Clicking the left and right buttons decreases or increases the length of the chord. Clicking the buttons marked '+8ve' and '-8ve', moves the pitch of the indicated note head up or down by one whole octave, respectively. Clicking the buttons marked 'Add' and 'Del' respectively add a new note head to the chord, and remove the indicated note head from the chord.
- A label showing which head of the chord is indicated; if the chord has more than one head, this can be changed by clicking the arrow buttons on the left and right of this label. Changing the indicated head will move the indicating arrow on the chord display up or down accordingly, and subsequent pitch changes will act upon the newly indicated note-head.
- A set of 'OK', 'Cancel' and 'Help' buttons.

3.1.1 Chord - Pitch Search and Replace

This function is part of the *Chord* [38] menu. It is used to replace selected pitches in a section of music with other pitches, for instance to carry out some limited transpositions. (If transposition by semitones is all you want to do, consider the *Group - Transpose* [47] function instead.)

To use it, select an area of music on the main stave and invoke this function. A window will appear showing two chords: on the left, a chord containing all the notes that will be replaced (the 'From' chord), and on the right a chord showing the notes they will be replaced by (the 'To') chord. You can change either of these by clicking on it, when you will be offered an edit dialog similar to that used in the *Chord - Edit Chord* [40] menu function.

There are also three option buttons: 'Query before each replacement', 'Replace pitches even if different durations', and 'Replace matching subsets of chord pitches'.

The first option is pretty obvious: if it's selected, you'll get the choice of whether to apply each individual replacement; if not, all possible replacements in the selected area will be carried out without question.

If 'Replace pitches even if different durations' is selected, the durations of the 'From' and 'To' chords will be disregarded and replacements will be chosen purely on the basis of pitch, rather than duration.

If 'Replace matching subsets of chord pitches' is selected, every chord that contains all the notes in the 'From' chord will have those notes replaced by those in the 'To' chord, regardless of whether it contains other notes as well. (If it does contain other notes, the extra notes will be unaffected.) If this option is not selected, only chords with pitches that exactly match the 'From' chord will be considered for replacement.

3.1.1 Chord - Guess Chord Names

This function is part of the *Chord* [38] menu. It inspects the selected chords, considering the notes in them together with the clef and key acting upon them, and tries to name them. The chord names are then drawn on the staff above the chords. See *Chords* [21] for a discussion of this function and particularly of how to influence when the chord names are visible.

This function was contributed by Guillaume Laurent.

3.1.1 Chord - Clear Chord Names

This function is part of the *Chord* [38] menu. It clears and chord names which have previously been obtained using the *Chord - Guess Chord Names* [41] function within the selected area. See *Chords* [21] for a discussion of the chord naming functions.

3.1.1 Chord - Label No Chords

This function is part of the *Chord* [38] menu.

If this menu option is selected, none of the chords or notes in the stave will be labelled with their names.

See *Chords* [21] for a discussion of the chord naming functions.

3.1.1 Chord - Label Named Chords Only

This function is part of the *Chord* [38] menu.

If this menu option is selected, only those chords in the stave whose names have been explicitly requested with the *Chord - Guess Chord Names* [41] function will be labelled with their names.

See *Chords* [21] for a discussion of the chord naming functions.

3.1.1 Chord - Label All Chords

This function is part of the *Chord* [38] menu.

If this menu option is selected, every chord within the stave will be labelled with a (guessed) name. Single notes will not be labelled.

See *Chords* [21] for a discussion of the chord naming functions.

3.1.1 Chord - Label Chords and Notes

This function is part of the *Chord* [38] menu.

If this menu option is selected, every chord and every lone note in the stave will be labelled with a (guessed) name.

See *Chords* [21] for a discussion of the chord naming functions.

3.1.1 Chord - Break Chord

This function is part of the *Chord* [38] menu.

Transforms the single chord within the selected area into a sequence of one-note chords. One chord will be created for every pitch found in the original chord. No guarantee is made about the order in which the new chords will appear.

3.1.1 Stave

The Stave menu contains functions for manipulating the stave by adding or removing staffs, performing staff-wide changes or altering the properties of each staff. Most of these are only available when a staff has been ‘‘selected’’, by positioning the cursor in it. More local changes, such as changing the time signature, are found on the *Bar* [48] menu. See *Operations on staffs* [22] for an overview of the functions on the Stave menu.

3.1.1 Stave - Add Staff

This function is part of the *Stave* [42] menu.

Adds a blank staff to the piece, first asking you where to place it. Staffs can be added first or last; if you’ve already placed the cursor on a particular staff, you can also choose to insert the new staff after that one on the score.

Adding a staff is an operation that cannot be undone using the *Editor Edit - Undo* [37] function; if you use this function, all the Undoable operations carried out previously will be forgotten by the editor. (It may seem a bit odd for adding a staff not to be undoable, but there is a solid technical reason for it.)

3.1.1 Stave - Delete Staff

This function is part of the *Stave* [42] menu.

Deletes the staff containing the cursor, and its entire contents.

3.1.1 Stave - Empty Staff

This function is part of the *Stave* [42] menu.

Deletes the entire contents of the staff containing the cursor.

3.1.1 Stave - Rename Staff

This function is part of the *Stave* [42] menu.

Allows you to enter a new name for the staff containing the cursor. This name is displayed to the left of the staff, and is used as the track name when that staff is exported in a MIDI file (see *Editor File - Export MIDI* [35]).

3.1.1 Stave - Show MIDI Patches

This function is part of the *Stave* [42] menu.

Each staff in a piece can be assigned a MIDI patch number (see *Stave - Assign MIDI Patch* [43]). This function toggles the display of patch numbers; if you switch on the display, patch numbers will be shown at the right-hand end of each staff in the Editor window.

3.1.1 Stave - Assign MIDI Patch

This function is part of the *Stave* [42] menu. It allows to you choose a MIDI patch number for the selected staff.

Each staff in a piece can be assigned a MIDI patch from 0 to 127. This will be used, when exporting or sequencing a MIDI file (see *Editor File - Export MIDI* [35] and *Editor File - Sequence* [36]), as the instrumental patch to be used for the track corresponding to that staff; the track will begin with a MIDI Program Change event to change to the specified patch.

The default MIDI patch is patch 0 (zero); any staffs that remain mapped to this patch when a MIDI file is exported will be written without any MIDI Program Change event.

Use *Stave - Show MIDI Patches* [43] to choose whether the Editor should display MIDI patch numbers on the screen during normal operation.

3.1.1 Stave - Bracket with Next

This function is part of the *Stave* [42] menu.

Declares that the staff containing the cursor is to be connected to that immediately below it; for example, if they belong to the same instrument. Bar lines on the two staffs will be joined together, and the two staffs will be bracketed together on the screen.

3.1.1 Stave - Unbracket

This function is part of the *Stave* [42] menu.

Declares that the staff containing the cursor is not to be connected to any other above or below it. This is the default state for staffs; this function is used to negate the effect of *Stave - Bracket with Next* [44] .

3.1.1 Stave - Fill to End

This function is part of the *Stave* [42] menu.

Ensures that the staff containing the cursor is of the same length as the longest staff in the current piece. This is effected by filling up the staff with rests, from the end of the last note or rest in it up to the end of the piece.

If no staff is selected, or optionally even if one is selected, all staffs will be filled with rests so as to be the length of the longest staff in the piece.

(This function is a bit stupid at the moment; for the best chance of success you should make sure your chosen staff already contains an exact number of full bars.)

3.1.1 Group

The Group menu contains functions which act on selected areas of music. These are only available when an area has been swept out with the pointer. There are functions to carry out actions such as beaming groups, adding or removing slurs, forming triplets and such like. See also *Grouping* [22] , which contains an overview of these functions.

3.1.1 Group - Tie

This function is part of the *Group* [44] menu.

Ties together two adjacent chords so as to play them as one long chord. You can only tie together two chords at a time, and they must be adjacent and have identical sets of pitches. To remove a tie, drag out the same two chords and use the *Group - Remove Indication* [45] function.

Tie information is used in producing MIDI output.

3.1.1 Group - Slur

This function is part of the *Group* [44] menu.

Places a slur (or phrase) mark over the selected group of items. To remove a slur, drag out the same area and use the *Group - Remove Indication* [45] function.

Slur marks are not currently considered when producing MIDI from a score.

3.1.1 Group - Crescendo

This function is part of the *Group* [44] menu.

Places a crescendo mark (opening hairpin) below the selected items. To remove a crescendo mark, drag out the same area and use the *Group - Remove Indication* [45] function.

Crescendo marks are only considered when producing MIDI from a score if you have the 'midiDynamics' resource set to True in the Rosegarden resource file. This resource defaults to False.

3.1.1 Group - Decrescendo

This function is part of the *Group* [44] menu.

Places a decrescendo mark (closing hairpin) below the selected items. To remove a decrescendo mark, drag out the same area and use the *Group - Remove Indication* [45] function.

Decrescendo marks are only considered when producing MIDI from a score if you have the 'midiDynamics' resource set to True in the Rosegarden resource file. This resource defaults to False.

3.1.1 Group - Remove Indication

This function is part of the *Group* [44] menu.

Removes any slurs, ties or crescendo or decrescendo marks which start and end completely within the selected area.

3.1.1 Group - Beam

This function is part of the *Group* [44] menu.

Beams together the selected chords. The angle and placement of the beam are entirely at the editor program's own discretion.

3.1.1 Group - Simple Tuplet

This function is part of the *Group* [44] menu.

Forms the selected chords into a triplet group, or similarly structured group (containing however many notes you have selected).

The Simple Tuplet function will form a group of three notes into a triplet group, four notes into a group which plays in the time of three, five notes into a group playing in the time of four, and so on. (As a special case, a group of two notes will be made to play in the time of three, rather than one.)

Using the Simple Tuplet function always has the same effect as using the *Group - Tuplet* [46] function and then accepting the default tuplet setup from its dialogue box.

3.1.1 Group - Tuplet

This function is part of the *Group* [44] menu.

Forms the selected chords into a triplet or other tuplet group.

This function allows you to re-express a group of any number of beats so as to play in the time of any other number. This allows you to make triplet groups, but also a lot of other madly stretched or squashed groups. When this function is activated, it will first show a dialogue box for you to select the particular kind of tuplet you want for the group of notes. This gives you two things to change:

- a menu offering all the possible expressions of the current length of the selected group in terms of basic notes of various lengths. For instance, if you have selected a group containing a crotchet and a quaver, you will be offered the choice of considering it as three quavers, six semiquavers, twelve demisemiquavers, four dotted semiquavers or various other possibilities.
- a field in which you can enter the number of your selected basic notes in which you want the group re-expressed.

Thus to play a crotchet and quaver group in triplet time, you'd select ''3 quavers'' as the original time of the group and ''2'' as the number of quavers to squash it into. For a more ambitious tuplet group, you might draw say 14 semiquavers and then ask to squash them into the time of say 8. Anything is possible (and most of it turns up in real music, somewhere).

The Tuplet dialogue box will normally attempt to give you a sensible default choice --- for instance, if you have selected a group three quavers long, you will be offered default selections corresponding to quaver triplets.

Tuplet groups are always displayed with beams, if the constituent notes are short enough, and if you make tuplets of notes already grouped, the containing group will be broken.

Note that Rosegarden will not normally split tuplet groups across bar lines. It is possible to split up tuplet groups in certain situations (for instance by making a beamed group from part of an existing tuplet group); the results may be unexpected.

Notice also that it isn't possible to see, by looking at a tuplet group out of context, what time it's supposed to be played in. For instance, a group of three quavers marked with a

3 may be intended to play in the time of four, rather than the usual two. The only way to work out the intention is to look at the time signature and do some arithmetic. (This is a restriction of the musical notation, not of Rosegarden.)

3.1.1 Group - Break Group

This function is part of the *Group* [44] menu.

Breaks up any groups which have been formed on the items within the selected area. This undoes the effect of any uses of *Group - Beam* [45] and *Group - Tuplet* [46] on any of the selected items.

3.1.1 Group - Auto Beam

This function is part of the *Group* [44] menu.

Searches for sequences of notes within the selected area which could be beamed together but aren't, and beams them. The beaming algorithm is the same as that used when importing MIDI (see *Importing MIDI files* [25]); it is not guaranteed to make a good job of it.

Using this function will (probably) not disturb any beamed groups already present in the selected area. If you want to unbeam the area first, use *Group - Break Group* [47] on it first.

3.1.1 Group - Transpose

This function is part of the *Group* [44] menu.

Transposes a section of music, first prompting you to enter the number of semitones by which to transpose the music up or down.

If an area is selected, its contents are transposed; if the cursor is placed but no area is swept out, the contents of the staff containing the cursor are transposed, from the cursor position to the end of the staff; if no area is swept and the cursor is not placed, the whole piece is transposed.

Note that transposition is in terms of semitones and not the current key or harmony.

3.1.1 Group - Invert

This function is part of the *Group* [44] menu.

Inverts a section of music, first prompting you to enter the number of semitones by which the first note of the group differs from the central (fixed) point of the inversion.

If an area is selected, its contents are inverted; if the cursor is placed but no area is swept out, the contents of the staff containing the cursor are inverted, from the cursor position to the end of the staff; if no area is swept and the cursor is not placed, the whole piece is inverted.

Note that inversion is in terms of semitones and not the current key or harmony.

3.1.1 Group - Retrograde

This function is part of the *Group* [44] menu.

If an area is selected, its contents are reversed in retrograde motion; if the cursor is placed but no area is swept out, the contents of the staff containing the cursor are reversed, from the cursor position to the end of the staff; if no area is swept and the cursor is not placed, the whole piece is reversed.

3.1.1 Bar

The Bar menu contains functions which affect one or more bars. For these to be available, a bar must be "selected", by positioning the cursor in it. There are functions to change the properties of the selected bar (such as *Bar - Repeat at Start* [49]), and also to effect changes upon the whole of the music from the selected bar onwards (such as *Bar - Time Signature* [50]). See *Changing bar properties* [23] and *Tempos and times* [24] for an introduction to the Bar menu.

3.1.1 Bar - Plain

This function is part of the *Bar* [48] menu.

Makes the bar containing the cursor into a plain bar, with single bar lines at both ends.

3.1.1 Bar - Double Bar

This function is part of the *Bar* [48] menu.

Places a double bar line at the end of the bar containing the cursor.

You can only place double bar lines at points where there is already some music present on every staff in the score; the staffs will then all acquire the double bar and will also all be connected with a dotted line.

Rosegarden always tries to keep double bar lines in the same place as you originally placed them. In practice this means they always remain immediately before the item (note, rest or whatever) that was at the start of the bar following the one you asked to place the double bar after. In keeping the double bar line associated with that item, Rosegarden will happily ignore the fact that there may at some subsequent point be the wrong number of beats in the bar before it.

If you remove the item immediately after a double bar line, the bar line might also disappear, to be replaced by a simple single bar. This is a side-effect of Rosegarden's attempts to keep the double bar associated with the same item (and is a known bug).

The editor may also show strange side-effects if you try to place a double bar line at a point where there are tied notes on one or more staffs -- those staffs will not acquire the double bar, even when the tied notes are rejoined. (See *Inserting notes and signs* [18] for an explanation of automatically-tied notes.)

3.1.1 Bar - Repeat at Start

This function is part of the *Bar* [48] menu.

Places a double bar line (see *Bar - Double Bar* [48]) with repeat dots at the start of the bar containing the cursor. Repeat information is currently ignored when playing and sequencing music.

3.1.1 Bar - Repeat at End

This function is part of the *Bar* [48] menu.

Places a double bar line (see *Bar - Double Bar* [48]) with repeat dots at the end of the bar containing the cursor. Repeat information is currently ignored when playing and sequencing music.

3.1.1 Bar - End Bar Here

This function is part of the *Bar* [48] menu.

Inserts a 'fixed' barline at the selected point; this barline will remain in place, immediately following the item it was originally inserted after, even when other notes are inserted before it. This behaviour is very similar to that of double- and repeat-bars inserted using the *Bar - Double Bar* [48] , *Bar - Repeat at Start* [49] and *Bar - Repeat at End* [49] functions. You can remove a fixed barline by applying the *Bar - Plain* [48] function at an appropriate point.

(This function can be used to make anacruses: bars, normally at the start of a piece, that contain fewer beats than normal.)

3.1.1 Bar - Label

This function is part of the *Bar* [48] menu.

Prompts for a textual label for the bar containing the cursor. The label is displayed above the bar line at the start of the bar, and replaces any label previously marking that bar.

Labels are actually associated with sections of notes, rather than bar lines; therefore, if the music in the current bar is moved left or right as a result of editing something before it, the label will travel with it.

3.1.1 Bar - Remove Label

This function is part of the *Bar* [48] menu.

Removes any label currently marking the bar containing the cursor.

3.1.1 Bar - Metronome

This function is part of the *Bar* [48] menu.

Prompts for a pairing of a note length and a tempo indication, to be placed above the bar containing the cursor. The tempo indication is assumed to express the number of notes of the given length whose lengths together amount to one minute, and is preserved in any MIDI output generated from this buffer (see *Editor File - Export MIDI* [35]).

In which staff the cursor is currently placed is not important.

3.1.1 Bar - Remove Metronome

This function is part of the *Bar* [48] menu.

Removes any metronome marking found above the bar containing the cursor. In which staff the cursor is currently placed is not important.

3.1.1 Bar - Time Signature

This function is part of the *Bar* [48] menu.

Prompts for a new time signature to be placed at the start of the current bar. The time signature may be placed in the current staff only, or in all staffs in the piece, and affects the formatting of the music in this and all subsequent bars (or until another contradictory time signature is met). Time signatures are preserved in MIDI output (see *Editor File - Export MIDI* [35]).

3.1.1 Bar - Key Signature

This function is part of the *Bar* [48] menu.

Inserts a new key signature at the point where the cursor is placed. If a key signature is already in force, it will be cancelled appropriately in combination with introducing the new one.

To choose the new key signature, you will be given a dialogue box showing part of a staff with a treble clef, with the selected signature shown on the staff. The box has up and down arrow buttons which add or remove sharps, or remove or add flats, to the signature, plus an option menu which you can use to select a particular signature by name.

There is also an option to transpose the rest of the staff into the new key. If selected, this will simply cause any notes affected by the new key signature to be sharpened or flattened accordingly. Otherwise, the note pitches will be left as they were and accidentals will be introduced as necessary. (This sense of ''transposition'' is thus different from that invoked through the *Group - Transpose* [47] function.)

Once a key signature has been inserted into a piece of music, it works like any other musical item. In particular you can select it with the selection cursor and (for example) delete it as normal. See *Inserting notes and signs* [18] for comments on the semantics of key signatures.

Key signatures are preserved as meta-events in MIDI output (see *Editor File - Export MIDI* [35]), although their effects are also carried out explicitly on the note pitches.

3.1.1 Text

The Text menu contains functions for adding and removing textual labels to or from the musical score. These labels are saved with the music (see *Saving an editor file* [16]), but are not properly preserved in MIDI files (see *Exporting MIDI files* [26]). Most of these functions require that the cursor is placed, in order to position the text above or below the cursor position. Functions for handling labels above the bar line are in the *Bar* [48] menu. See also *Adding text* [24] for further descriptions of these functions.

3.1.1 Text - Add Text options

These functions are part of the *Text* [51] menu.

The Text menu contains four functions dedicated to adding text to the staff at the cursor position. The menu list simply shows ‘Text’ four times, each time accompanied by a small icon indicating the position and style which will be used by that function.

The four possible styles are: immediately above the staff, in a medium font; considerably above the staff, in a larger font; and below the staff, in medium and italic fonts respectively. There is also a style (*Text - Dynamic* [51]) for writing simple dynamic markings.

Each function prompts for the text you would like to display.

3.1.1 Text - Dynamic

This function is part of the *Text* [51] menu.

Allows you to select a simple dynamic force marking, from ‘pppp’ through to ‘ffff’. This marking will be placed below the staff as a normal Text object, except that it may be taken into account when writing MIDI files or sequencing.

3.1.1 Text - Clear Group

This function is part of the *Text* [51] menu.

Removes all text above or below any items in the selected area.

3.1.1 Text - Clear Bar

This function is part of the *Text* [51] menu.

Removes all text above or below any items in the bar containing the cursor. This includes the bar label, if one is present.

3.1.1 Text - Clear Staff

This function is part of the *Text* [51] menu.

Removes all text above or below any items in the staff containing the cursor. First checks to make sure of your intentions.

3.1.1 Marks

The Marks menu lists several kinds of markings. Sweeping out an area of music and selecting one of these marking functions will cause every chord within the area to be decorated with the given mark, whilst the Unmark function removes any decorations from any chords within the selected area. There is currently no way to place marks on items other than chords (for instance, you cannot place a pause mark over a rest). See also *Marking chords* [25], which describes further the use of the Marks menu.

3.1.1 Editor Filter

The Filter menu lists all the Petal filters the Editor found in preset and configurable filter directories when it was started up. It also contains functions to allow explicit loading and saving of staffs to Petal filter file formats.

To apply a Petal filter, select one staff of the piece (this will be the Petal 'main' staff; the Petal 'reference' staff will be the one below it, if there is one, or the topmost staff otherwise); then select the filter name from the menu. You can also use *Editor Filter - Apply Petal Filter* [52] to apply a filter from any suitable file in any directory. (If this is the first item in the menu, then the Editor found no filters in the standard directories when it started.)

This menu is very similar to the *Sequencer Filter* [65] menu.

For more documentation about Petal and its filters, consult the documentation that is distributed along with the Petal source release. Petal is a work-in-progress by Guillaume Laurent.

3.1.1 Editor Filter - Apply Petal Filter

This function is part of the *Editor Filter* [52] menu.

This allows you to apply a Petal filter from any appropriate filter file in any directory. To use it, you first should select one staff of the piece to be the Petal 'main' staff; the Petal 'reference' staff, if needed, will be the one below it, if there is one, or the topmost staff otherwise.

3.2 Sequencer

3.2.1 Sequencer File

The File menu in the Rosegarden Sequencer provides functions for loading, saving, recording and playing MIDI files. See *Loading a MIDI file* [28] , *Saving a MIDI file* [28] , *Recording MIDI data* [28] , *Playing a MIDI file* [32] and *Generating Csound* [32] for discussions of how to use this menu.

3.2.1 Sequencer File - Record

This function is part of the *Sequencer File* [53] menu.

Start recording MIDI data being broadcast by an external MIDI device. The incoming data is recorded until the *Sequencer File - Stop* button is pressed. The recorded information will appear as a new track on the track list.

See also *Recording MIDI data* [28] .

3.2.1 Sequencer File - Open

This function is part of the *Sequencer File* [53] menu.

This option allows you to read in a previously saved MIDI file. See *Loading a MIDI file* [28] .

3.2.1 Sequencer File - Close

This function is part of the *Sequencer File* [53] menu.

This option closes the current MIDI file. If the file has been altered since it was read in then you will be asked whether you wish to save it before it is discarded.

3.2.1 Sequencer File - Save

This function is part of the *Sequencer File* [53] menu.

Saves the current MIDI file. If the file has not been previously saved then you will be prompted to supply a file name. See *Saving a MIDI file* [28] .

3.2.1 Sequencer File - Save As

This function is part of the *Sequencer File* [53] menu.

Saves the current MIDI file under a new name. You will be prompted to supply the new file name. See *Saving a MIDI file* [28] .

3.2.1 Sequencer File - Return to Start

This function is part of the *Sequencer File* [53] menu.

Returns the track position counter to the beginning of the track stopping playback if the sequencer is in that mode. There is a toolbar shortcut to this function denoted by the barred left arrow.

3.2.1 Sequencer File - Rewind

This function is part of the *Sequencer File* [53] menu.

Moves the track counter back a preset amount in the track. If playback is in operation then all pending notes are silenced and playback resumed at the new position immediately. If the sequencer is not playing then this function will move the counter back, repeated applications can move the counter back to the beginning of the track (Time Position 0).

The toolbar shortcut to this function is the double backward arrowed icon.

See the *Track Counter* [55] .

3.2.1 Sequencer File - Play

This function is part of the *Sequencer File* [53] menu.

Plays the MIDI file out via the MIDI port. See *Playing a MIDI file* [32] . Note that playback takes up from the point at which it was last stopped, unless you've used a control such as *Sequencer File - Return to Start* [53] in the meantime.

3.2.1 Sequencer File - Stop

This function is part of the *Sequencer File* [53] menu.

Stops the playback of the MIDI file. This menu option is only available while a file is being played or recorded. See *Playing a MIDI file* [32] and *Recording MIDI data* [28] .

3.2.1 Sequencer File - Fast Forward

This function is part of the *Sequencer File* [53] menu.

Moves the track counter a preset amount forward in the track. If the sequencer is playing then all pending notes are silenced and playback continued from the new position in the track. If the sequencer is not playing then this function will move the counter on, repeated applications taking the counter to the end of the track. The toolbar shortcut to this function is denoted by the double forward arrowed icon.

See the *Track Counter* [55] .

3.2.1 Sequencer File - Skip to End

This function is part of the *Sequencer File* [53] menu.

Moves the track to the end of the current file. Any playback is stopped. The *Track Counter* [55] will be incremented to match the value for the end of the track.

3.2.1 Sequencer File - Export Csound

This function is part of the *Sequencer File* [53] menu.

Converts the current MIDI file to Csound score format in a fairly simple format. It does translate tempo and notes; other MIDI events get translated as comments. There is no attempt to generate f statements or orchestra files.

See also *Generating Csound* [32] .

3.2.1 Sequencer File - Play through Slave

This function is part of the *Sequencer File* [53] menu.

Invokes an external program of your choosing, giving it (on the command-line) the name of a temporary MIDI file generated from the current file. This is would normally be used for playing MIDI through a supplied MIDI player on a system for which Rosegarden does not directly support MIDI playback.

3.2.1 Sequencer File - Notate

This function is part of the *Sequencer File* [53] menu.

Passes the MIDI file to *The Rosegarden Editor* [15] to be used as the basis for a new musical score. If the editor is not active at the time of the request then it is started automatically.

3.2.1 Sequencer File - Exit

This function is part of the *Sequencer File* [53] menu.

Use this option to leave the sequencer. If the current MIDI file has been altered but has not been saved then you will be asked whether you wish to save it before leaving the sequencer.

3.2.1 Track Counter

On the bottom left of the Track List display the Track Counter consists of three label boxes. The left-most label naming the function, the middle box giving current position in the track and the right-hand box giving the total track length.

As playback continues (see *Playing a MIDI file* [32]) the counter increments with each note that is played. The *Sequencer File - Fast Forward* [54] and *Sequencer File - Rewind* [54] functions move the current playing position forward and backwards through the track whether it is playing or not.

3.2.1 Playback Status

Each track in a MIDI file has an associated playback status in the Rosegarden Sequencer. The track can either be Active or Mute. If Active the track will be included in any playback (see *Playing a MIDI file* [32]). If Mute the track will not be included in playback and it will not be played when Recording is taking place (*Recording MIDI data* [28]).

3.2.1 Sequencer Edit

The sequencer *Edit* menu provides access to the clipboard functions (cut, copy, and paste) and related general editing functions. See the sections on *Editing with the Event List* [30] and *Viewing with the Piano Roll* [31] for further descriptions of how this menu works.

3.2.1 Sequencer Edit - Undo

This function is part of the *Sequencer Edit* [56] menu.

This function allows you to undo the last operation you performed on a MIDI track. Only operations that affect a single track can be undone in this manner.

3.2.1 Sequencer Edit - Delete

This function is part of the *Sequencer Edit* [56] menu.

This function deletes the selected area. The area to be deleted may be specified from either the main window track list or from areas selected in the *Piano Roll* or *Event List* sub-windows.

3.2.1 Sequencer Edit - Cut

This function is part of the *Sequencer Edit* [56] menu.

This function removes the contents of the selected area, moving them into the clipboard. The previous contents of the clipboard, if any, are lost. The area to be cut may be specified from either the main window track list or from areas selected in the *Piano Roll* or *Event List* sub-windows.

3.2.1 Sequencer Edit - Copy

This function is part of the *Sequencer Edit* [56] menu.

This function copies the contents of the selected area to the clipboard. The selected area is left unaltered by this operation. The previous contents of the clipboard, if any, are lost. The area to be copied may be specified from either the main window track list or from areas selected in the *Piano Roll* or *Event List* sub-windows.

3.2.1 Sequencer Edit - Paste

This function is part of the *Sequencer Edit* [56] menu.

This function inserts the contents of the clipboard at the point specified by the selection cursor. The insertion point may be specified from either the main window track list or selections made in the *Piano Roll* or *Event List* sub-windows.

3.2.1 Sequencer Edit - Show Clipboard

This function is part of the *Sequencer Edit* [56] menu. Displays a small window showing the contents of the clipboard in event list format.

3.2.1 Track

The track menu provides a number of functions that operate over whole tracks. The menu is accessible from the sequencer main window and from piano roll and event list windows. See also *Displaying a track* [30] , *Organising recorded data* [29] and *Preparing a MIDI file for the Editor* [31] .

3.2.1 Track - Show Event List

This function is part of the *Track* [58] menu. Pulls up a window displaying a list of the MIDI events in the selected track. From this new window you have access to the *Track*, *Edit* and *Event* menus, which you can use to manipulate events in this track. You may have as many event list windows as you like open simultaneously, with a limit of one window per track.

The event list gives a textual representation of the MIDI events in the order they appear in the track. The basic format for each event description is:

```
<Event Time> <Event Name> [<Event Data>]
```

The event time is given as the number of beats from the start of the track. The events can be of the following types:

```
Note On                <Chnl> <Note> <Octave> <Velocity> <Duration>
Controller Change      <Chnl> <Controller Number> <Value>
Program Change         <Chnl> <Program Number>
Polyphonic Aftertouch  <Chnl> <Note> <Octave> <Velocity>
Monophonic Aftertouch  <Chnl>
Pitch Bend             <Chnl> <Pitch Change>
Set Tempo              <Tempo> bpm
Set SMPTE Offset       <SMPTE Values>
Time Signature         <Time Sig>
Key Signature          <Key Sig>
Text Event             <Text String>
```

Any events which are encountered which the sequencer cannot support will be displayed as,

```
Unsupported Event      <Hex dump of event arguments.>
```

3.2.1 Track - Show Piano Roll

This function is part of the *Track* [58] menu. The Piano Roll window displays the contents of a track using a graphical notation similar to the roll of paper controlling a traditional pianola. Notes are displayed positioned on a musical stave and are represented as boxes. The length of the note is thus proportional to the size of the box, and its position denotes the pitch of the note and its place in the piece. Natural notes are shown as dark boxes, sharps/flats are shown as light boxes. MIDI events other than note events are not shown on the piano roll.

From the Piano Roll window you have access to the *Track* and *Edit* menus, which you can use to manipulate the track.

3.2.1 Track - Mute All Tracks

This function is part of the *Track* [58] menu.

Sets the *Playback Status* [55] of all tracks to Mute.

3.2.1 Track - Activate All Tracks

This function is part of the *Track* [58] menu.

Sets the *Playback Status* [55] of all tracks to Active.

3.2.1 Track - Change All to Device

This function is part of the *Track* [58] menu.

This option pops up a dialog to allow the user to select a device for all tracks to be assigned to for playback. Select the required device from the drop-down menu in the dialog and press Ok to set the tracks to this device. Press Cancel to leave the device allocations unchanged.

3.2.1 Track - Rename

This function is part of the *Track* [58] menu.

Renames the selected track. Any previous Track Name events for that track will be deleted.

3.2.1 Track - Track Info

This function is part of the *Track* [58] menu. Displays useful information about the track, including its length in beats and the total number of events contained in the track.

3.2.1 Track - Clone

This function is part of the *Track* [58] menu.

Produces an identical copy of the selected track. This new track is added to the end of the track list in the main window.

3.2.1 Track - Merge

This function is part of the *Track* [58] menu.

Merges the selected track with another track. After selecting this option you will be prompted for the number of the track you wish to merge with the selected track. Tracks are merged in a zip fastener fashion.

3.2.1 Track - Filter By Channel

This function is part of the *Track* [58] menu.

Removes all events from the selected track except for those on the specified channel. After selecting this option you will be prompted for the channel that is to be retained.

3.2.1 Track - Filter By Event

This function is part of the *Track* [58] menu. Removes all the events from the selected track that are not of the types specified by the user. After selecting this option you will be presented with a dialogue box showing the possible MIDI event types. Any event types that are ticked in this dialogue box will be retained in the track.

3.2.1 Track - Filter By Pitch

This function is part of the *Track* [58] menu. Removes all events from the selected track except those above (or below) the pitch specified by the user. On selecting this option you will be presented with a dialogue box in which you must enter the pitch from which to filter and the direction in which notes should be retained (i.e. above or below the specified pitch). The pitch is given in a note and octave format. Note C in octave 5 is middle C.

3.2.1 Track - Split By Channel

This function is part of the *Track* [58] menu.

Divides the selected track into two, creating a new track for all the events that are not on the specified channel. You will be prompted for the channel number whose events are to be retained in the original track; events not on that channel will be moved into a newly created track.

3.2.1 Track - Split By Pitch

This function is part of the *Track* [58] menu.

Divides the selected track into two, modifying the existing track exactly as if you had used *Track - Filter By Pitch* [60] and transferring all events which fall outside the selected pitch range onto a newly created track.

3.2.1 Track - Change Channel

This function is part of the *Track* [58] menu. Moves all events in the selected track that are on one channel to a different channel. After selecting this option you must enter the channel to change from and the channel to change to into the waiting dialogue box. Channels are specified by a number in the range 0 to 15.

3.2.1 Track - Quantize

This function is part of the *Track* [58] menu. This function allows you to regularize the duration and position of the notes in the selected MIDI track. If this option is selected from the main window Track menu, and no track is selected on the track list, then quantization will be carried out on all the tracks in the file.

After selecting this option you will be presented with a dialogue box containing two buttons and two menus. Pressing the ‘Quantize Note Positions’ or ‘Quantize Note Durations’ buttons enables or disables the specified type of quantization. When either of the quantization modes are enabled the associated menu will become sensitive, allowing you to select the granularity to quantize to (i.e. the smallest note duration or position permissible in the piece). Selecting OK will then perform the quantization.

Note that the ‘Quantize All Tracks’ operation cannot be undone.

3.2.1 Track - Transpose

This function is part of the *Track* [58] menu. Allows the user to transpose the selected track up or down by a number of semitones. If the option is selected from the main window and no track has been selected from the track list then the operation will be carried out across all the tracks in the file.

3.2.1 Midi

This menu contains options used for configuring the MIDI setup to suit your equipment. It is only available from the sequencer main window.

3.2.1 Midi - MIDI Setup

This function is part of the *Midi* [62] menu.

This function allows you to configure Rosegarden's setup for for MIDI playback and recording.

Upon selecting this option you are presented with a dialogue box from which you can change a number of sequencer MIDI-related parameters. Exactly what these are depends on which system you're running Rosegarden on. Here's a list.

- If you're using Linux with OSS/OSS-Lite, expect to find:

Device. This shows the logical device or port through which the sequencer will be driven. The initial value is taken from that of the 'midiPort' resource found in the Rosegarden application default file on startup.

Record Device. Showing the device from which the Sequencer will accept MIDI events. Currently the device selected will be at the Sequencer's discretion, choosing the first "Midi device" from the list of devices shown by performing (> is the Linux prompt):

```
> cat /dev/sndstat
```

and checking Midi device 0.

Event Buffer Size. This specifies the size of the storage buffer used when recording real-time MIDI data. The default size of 10,000 events should be large enough for most pieces.

Score Tracking. This menu gives you a choice of three levels of score tracking. If any other than ''No Score Tracking'' is selected, the sequencer will attempt to use the Rosegarden Editor to show the progress of playback, by shading in the musical notation score as it plays. (Note that the extra processor load might impair playback performance.)

- If you're using an SGI with the default (dmedia) MIDI support, expect to find exactly the same dialog as with the OSS driver for Linux with a single addition:

Synchronous Mode. This attempts to synchronise the output of the sequencer with the position of the sequencer in the track (*Track Counter* [55]).

- If you're using an SGI with the Zilog hardware controller sequencing code (probably on a system older than IRIX 5.3), expect to find the same as the with the SGI (dmedia) driver with the addition of another check box:

Maintain Tempo. With this option selected the sequencer will attempt to maintain the tempo of a piece even if it is not being given enough processing time to do the job properly. This option can be switched on or off according to taste. With the option enabled a heavily loaded machine will produce jerky output, but maintain the correct overall tempo. With the option disabled a heavily loaded machine will produce slower but slightly less jerky playback. (Of course the ideal solution is to use a lightly loaded machine when playback quality will be much higher).

- If you're using some other system, it may well not be supported at all. To find out if this is the case, or if you're having trouble with a system that is supposed to be supported, check the current version of the Rosegarden FAQ.

3.2.1 Midi - Set Initial Patches

This function is part of the *Midi* [62] menu.

This function provides a very simple framework for applying initial program changes (i.e. allocating voices) to MIDI channels. The channels listed from 0 to 15 can be populated in the dialog with values between 0 and 127 for Program Change events to send to the specified channel before any playback commences. This function is really only useful in the absence of Program Change events in the MIDI tracks themselves and will eventually become deprecated.

3.2.1 Midi - File Info

This function is part of the *Midi* [62] menu. Report useful information about the current MIDI file, including its full pathname, the number of tracks and events, and the file format type.

3.2.1 Midi - Set Timebase

This function is part of the *Midi* [62] menu.

Alters the timebase for the current file. The timebase corresponds to the number of time divisions per MIDI quarter note. Changing this value will result in tempo settings and note lengths being incorrect, and may cause playback to become garbled.

3.2.1 Midi - Reset

This function is part of the *Midi* [62] menu.

This function for the OSS driver sends a full device reset to all devices found in the configuration. (see SNDCTL_SEQ_RESET in /usr/include/linux/soundcard.h). It is not implemented for any other driver.

3.2.1 Midi - System Reset

This function is part of the *Midi* [62] menu.

Performs the same as a *Midi - Reset* [63].

This function for the OSS driver sends a full device reset to all devices found in the configuration. (see SNDCTL_SEQ_RESET in /usr/include/linux/soundcard.h). It is not implemented for any other driver.

3.2.1 Midi - System Exclusive

This function is part of the *Midi* [62] menu.

This function has not yet been implemented.

3.2.1 Event

The *Event* menu is available from *Event List* windows and allows the user to create or modify MIDI events for the track displayed in that window. See also *Editing with the Event List* [30] .

3.2.1 Event - Modify Event

This function is part of the *Event* [64] menu.

This calls up a dialogue box listing the properties of the selected event, and allowing you to change each by editing its textual representation.

If more than one MIDI event is selected, dialogue boxes will be shown for each selected event in turn.

3.2.1 Event - Insert Text Event

This function is part of the *Event* [64] menu. This allows you to insert text events into a MIDI track. You are presented with a dialogue box with a pulldown menu and two data fields. The menu allows you to choose the type of the text event you are about to insert which may be one of the following,

```
Text Event
Copyright Notice
Track Name
Instrument Name
Lyric
Text Marker
Cue Point
```

The two data fields allow you to enter the time of the event and the textual contents of the event.

3.2.1 Event - Insert Tempo Event

This function is part of the *Event* [64] menu.

This function is not yet implemented.

3.2.1 Event - Insert Note

This function is part of the *Event* [64] menu. It allows you to insert note events into a MIDI track. You are presented with a dialogue box asking you to provide values for the time, channel, pitch, octave, velocity and duration parameters of the event. The pitch should be specified as a note name. Sharps are indicated by appending the octothorpe ('#') sign to the name. Flats cannot be described; instead, write the note below with a sharp. The octave value must be in the range 0 to 10; middle C is the C in octave 5. The channel should be any number between 0 and 15. The velocity should be in the range 0 to 127, although the scale over which this operates is device-dependent. Duration is measured in beats, and may be a fractional value.

3.2.1 Event - Insert Controller Change

This function is part of the *Event* [64] menu. It allows you to enter controller change events at any point in a MIDI track. You are presented with a dialogue box and must supply data for the four fields in it. You must specify the time of the event, the channel the event is to be transmitted on, the number of the controller that is to be changed, and the new value for the controller. Controller numbers and values are specific to the MIDI devices you are using - your manuals should contain a table of appropriate values.

3.2.1 Event - Insert Aftertouch

This function is part of the *Event* [64] menu.

3.2.1 Event - Insert Program Change

This function is part of the *Event* [64] menu. It allows you to enter program change events at any point in a MIDI track. You are presented with a dialogue box and must supply data for the three fields in it. You must specify the time of the event, the channel that the event is to be transmitted on, and the new program number to be transmitted.

3.2.1 Event - Insert Pitch Change

This function is part of the *Event* [64] menu.

3.2.1 Sequencer Filter

The Filter menu lists all the Petal filters the Sequencer found in preset and configurable filter directories when it was started up. It also contains functions to allow explicit loading and saving of staffs to Petal filter file formats.

To apply a Petal filter, select one track of the piece (this will be the Petal 'main' track; the Petal 'reference' track will be the one below it, if there is one, or track 1, if there is one, or 0 otherwise); then select the filter name from the menu. If no track is selected, you will be prompted for the number of the 'main' track. You can also use *Sequencer Filter*

- *Apply Petal Filter* [66] to apply a filter from any suitable file in any directory. (If this is the first item in the menu, then the Sequencer found no filters in the standard directories when it started.)

This menu is very similar to the *Editor Filter* [52] menu.

For more documentation about Petal and its filters, consult the documentation that is distributed along with the Petal source release. Petal is a work-in-progress by Guillaume Laurent.

3.2.1 Sequencer Filter - Apply Petal Filter

This function is part of the *Sequencer Filter* [65] menu.

This allows you to apply a Petal filter from any appropriate filter file in any directory. To use it, you first should select one track of the piece to be the Petal ‘main’ track; the Petal ‘reference’ track, if needed, will be the one below it, if there is one, or track 1, if it exists, or 0 otherwise. If no track is selected, you will be prompted for the number of the ‘main’ track.

4. Petal

First, a warning:

THIS IS ALPHA CODE.

And now back to our scheduled program.

4.1 Introduction and presentation

4.1.1 What is Petal?

Petal is a Tcl-based language that is interfaced with Rosegarden. Although it can be used without Rosegarden (and Rosegarden can be used without it), it pretty much belongs to it. The concept of Petal stems from the CAL language that Cakewalk users are familiar with, except Petal is more powerful. Among the improvements are that Petal knows about harmonic concepts such as keys, scales and modes.

4.1.2 Requirements and building process

Petal requires Tcl v7.5 or above and tclmidi v3.1. To build it, just cd into the rosegarden/petal directory and type 'make'. This is fairly standard C code and there should be few problems from one platform to another. On some platforms, you may get a few error messages like 'xxx.tcl is not an ELF file' (if using Tcl v7.5) which you can safely disregard.

`tclmidi can be found at http://madonna.ME.Berkeley.EDU/greg/tclmidi/`

`Tcl can be found at http://www.sco.com/Technology/tcl/Tcl.html`

4.1.3 Basic concepts of Petal

Petal is intended to write tiny convenient tools called 'filters', which get some input from stdin, process it and spout some output on, guess what, stdout.

Rosegarden is actually made out of two seperate programs (the Editor and the Sequencer) and Petal is designed to be used from both. In a limited way, the same filter can be used both from the Editor and the Sequencer. This is achieved through a thick abstraction layer and a corresponding API.

4.1.3.1 Petal working mode

Internally, Petal can work on two different 'modes': MIDI, or EDITOR. The mode is passed as the first argument to the filter, and will trigger the loading of the relevant packages on Petal initialisation. The Sequencer and the Editor will set this argument accordingly, and you don't have to worry about it. But you'll have to take care of it whenever you are testing a newly written filter under your shell.

4.1.3.2 Main and reference track (staff), duplication

A filter normally operates one track (ie staff) only, called the main track, although it is being passed the whole piece. The number of this track is passed as the second argument to the filter. In the Editor, it is the currently selected staff. In the Sequencer, it's track #1 and you don't have any way to change this (yet).

Whether the output data will contain all the data in the main track or just the processed one is triggered by a boolean flag passed as third argument to the filter. 0 means no duplication, 1 means yes. Take the harmonizer filter for instance: it will turn only single notes into chords, and leave chords alone. If duplication is off, all the previously existing chords will be missing from the result. You can't change this argument's value from its current default (1) yet.

Finally, a complex filter may find it handy to have a reference track. For instance, a more elaborate harmonizer may like to know where the bass track is; the pattern.tcl filter is another example. The reference track number is passed as the fourth argument to the filter. In the Editor, it is the staff right below the selected (ie main) one, if any. In the Sequencer, it's the track next to the main one, and therefore you can't select it either (yet :-).

Note that all these arguments have defaults (which are irrelevant when calling a filter the normal way, that is from either the Sequencer or the Editor, rather than from you shell command line) which are discussed below (see Arguments section).

4.1.3.3 Sample filters

Two sample filters are included as `harmonizer.tcl` and `pattern.tcl`. They should be understandable without too many problems, and can be taken as samples for other filters you'd like to write.

4.2 Writing filters

As Petal is based on Tcl, you need to learn a bit of Tcl first unless you're already familiar with it, in which case you won't have any problems.

Every Petal filter should start with:

```
package require "Petal"
```

```
PetalInit
```

and end with

```
PetalExit
```

It should also be made executable, and the executable shell should be tclsh. In Unix, that just means you put '#!<path-to-tclsh>' on the first line, as usual. In Windows, I don't know.

4.2.1 Testing a filter

When writing a new filter, it is best to test it under your favourite shell before trying it with the 'Apply Filter' functions of the Editor or the Sequencer. A filter must be passed at least 1 and up to 4 arguments. It will read its input from stdin and dump the output to stdout, so don't forget redirections. For example, here's how to launch the harmonizer on a given MIDI file:

```
prompt> ./harmonizer.tcl MIDI maintrck# dupl reftrck# < test.mid > out.mid
```

For instance:

```
prompt> ./harmonizer.tcl MIDI 1 1 2 < test.mid > out.mid
```

This is how the filter will be invoked from the Sequencer.

To test a filter in the Editor mode, you must first use the Editor 'Dump All Staves' function to create a file that will be readable by a filter in the EDITOR mode. The created file will be /tmp/rosebowl.tcl (yes, it's Tcl code). Then apply the filter in the same way as above:

```
prompt>filter EDITOR maintrck# dupl reftrck# < /tmp/rosebowl.tcl > /tmp/rose.petal
```

The output file name should be /tmp/rose.petal since this is what the Editor function 'Read Staff' will look for, and this is the function you should use to read the result back into the Editor.

4.2.2 Arguments

- The first argument determines the Petal 'mode' ('MIDI' or 'EDITOR') according to what kind of data you intend to feed to it.

The next arguments are optional and have defaults. They are all integers.

- The second argument is the number of the main track (or staff) the filter will process.
- The third argument determines whether the content of the main track will be copied to the new track (the one which will contain the filter output) prior to any operation. 0 means no, any other value means yes.
- The fourth argument is the number of the reference track (see pattern.tcl for an example of its use).

4.2.3 About the 'Apply Filter' functions

In the Editor, applying a filter will leave your curent data untouched and just add one more staff. In the Sequencer, it will first close your file and reload something entirely new. Although a well-written filter should leave all previously existing data intact and return the input song with one more track, if something goes wrong you'll end up with nothing. It is best to save before you use this function, and you'll be offered to do so if your file wasn't. The two shell scripts 'fakeecho' and 'rawdump' were used to help debugging the 'Apply Filter' function in the Sequencer.

4.3 Petal commands

4.3.1 C commands

These commands are written in C for speed purposes. They could very well have been written in Tcl (and actually have been as prototypes), but since all of them are very heavily used, the speed gain is non-negligeable. These commands are:

- petalInitInternals
- ith
- isInScale
- noteIndex

All but the first of these commands accept pretty much the same set of options:

[-scale S] [-tonality K] [-mode M]

'tonality' and 'mode' expect integer arguments. You are strongly advised to use the pre-defined variables that contains all the appropriate values which you might need (see 'Variables' section below).

'scale' on the other hand expects a variable name, which holds a list of intervals. This is so that when you want to print the current scale, 'scaleMajor' is more readable than '2 2 1 2 2 2 1'.

The default values for these options are set in the corresponding 'default*' variables (see below).

When specifying a mode, you should also specify the scale as being the usual major scale (if not the current default). There is nothing in Petal to prevent you from not doing so, your are free to experiment with other scales, and the results should be consistent. But according to music theory (or more precisely, what my sources in music theory say), that's how it should be.

Note also that these commands accept arguments in two forms : a midi pitch, or a note symbolic name (e.g. C, C#, G, Gb etc..., the 'flat' sign being a lowercase 'b'). This last form is intended as a possible convenience only.

- [petalInitInternals]: actually doesn't do anything, is just here to trigger the loading and of the main Petal library and the initialization of all the internal vars.
- [ith][-scale S] [-tonality K] [-mode M] note|midi_pitch offset

Returns the note or midi pitch that is offset-th steps above (or below if offset is negative) the one passed as 1st argument. The use of the word 'step' here is perhaps misleading, as it should not be intended in the harmonic sense. Here are a few examples

```
(assuming major scale, in the key of C, ionian mode)

ith C 1

-> D

ith C 2

-> E

ith -scale scaleMinor C 2

-> D# Eb

ith -mode $modeDorian C 1

-> C# Db
```

- [isInScale][-scale S] [-tonality K] note|midi_pitch

Returns a boolean value indicating whether the given argument is in the scale. For example:

```
isInScale C

-> 1
```

- [noteIndex][-scale S] [-tonality K] [-mode M] note|midi_pitch

Returns the index (or position) of the note in the scale.

```
noteIndex C
```

```
-> 0
```

```
noteIndex D
```

```
-> 1
```

```
noteIndex G
```

```
-> 4
```

4.3.2 Pre-defined Variables

When the main Petal package is loaded and initialized, a bunch of ‘convenience’ variables are defined:

Some scales:

- scaleMajor
- scaleMinor, scaleHarmonicMinor, scaleMelodicMinor
- scaleWholeTone, scaleDiminished
- scalePentatonic, scalePentatonicMinor, scaleBlues
- scaleEnigmatic, scaleNeapolitan, scaleNeapolitanMinor, scaleHungarian, scaleChromatic.

Some modes:

- modeIonian
- modeDorian
- modePhrygian
- modeLydian
- modeMixolydian
- modeAeolian
- modeLocrian

Some tonalities:

- tonalityC
- tonalityCsharp, tonalityDflat
- tonalityD

- tonalityDsharp, tonalityEflat
- tonalityE
- tonalityF
- tonalityFsharp, tonalityGflat
- tonalityG
- tonalityGsharp, tonalityAflat
- tonalityA
- tonalityAsharp, tonalityBflat
- tonalityB

Those can typically be used as arguments to the options described above, for instance:

```
ith -mode $modeLocrian -tonality $tonalityD -scale scaleMinor E 1
```

Note that we want the variables' values for the mode and tonality only, not for the scale. You should also use these variables to set Petal's internal defaults, which are accessed through the following variables:

- defaultMode
- defaultTonality
- defaultScale

For instance:

```
set defaultMode $modeIonian
```

```
set defaultTonality $tonalityC
```

```
set defaultScale scaleMajor
```

There are automatic checkings on each of these variables which prevents from setting them to wrong values.

4.3.3 Data types: item, pitches and key changes

There currently are three abstract data types present both in the EDITOR and MIDI mode. Their sole purpose is to help unifying both concepts.

A track (staff) is an indexed set of several items, an item consisting of one or several pitches (positive integers) and/or possibly one key change, and having a given duration (which is zero in case of a single key change).

There are actually other things that can be in an item, but none of those are common to both modes and although you are perfectly allowed to fumble with them, you will lose compatibility if you do so.

The only thing that can be in an item in the EDITOR mode and not in the MIDI mode is a rest, since there are no MIDI events to denote such a thing. No functions are provided to specifically deal with rests in the EDITOR mode.

On the other hand, since an item in the MIDI mode is actually the list of all the MIDI events sharing an identical time index (in one given track, of course), any kind of MIDI event can be in an item.

4.3.4 Tcl commands

Every command or variable that is prefixed with "Petal" (PetalGetItem, PetalPutItem, $petalItemIndexes...) deals with Petal's internal data structures and should be used as the only API when dealing with those.

Below is the list of the provided functions common to both modes. Please refer to harmonizer.tcl and pattern.tcl for an example of how to use them. The arguments are cut-&-pasted from the actual Tcl declarations. If you're not familiar with Tcl, there are no types, and a sublist denotes a default value (e.g. PetalExit takes one argument which defaults to 0). The term 'track' is here used as a common term meaning either a midi track or a staff, depending on which mode Petal operates in.

- [AddPitches]{ item pitches } Adds the given pitches to the item and returns the resulting item.
- [ApplyPitchFilter]{ filter {track $mainTrack} } Applies the 'filter' proc on each pitches of the given track. The filter will be passed pitch lists only, key changes will be dealt with automatically.
- [GetDuration]{ item } returns the item duration
- [GetOnePitch]{ item N } return the Nth pitch in the item, or nothing if N is greater that the number of pitches.
- [GetPitches]{ item } returns the list of all the pitches in the item.
- [HarmonizePitch]{ pitch under over }
- [IsKeyChange]{ index } returns 1 if there is a key change at that index, 0 otherwise.
- [IsSinglePitch]{ item } returns 1 if there is only one pitch in the item, 0 otherwise.
- [NbOfPitches]{ item } returns the number of pitches in the item
- [NewNoteItem]{ duration pitches } creates a new item having the given duration and pitches (yes, it should be called 'NewChordItem' or something).

- [PetalExit]{ { exitCode 0 } } must be the last thing a filter calls, will 'do the right thing' depending on which mode Petal is in.
- [PetalGetItem]{ track index } returns the item in the given track at that time index.
- [PetalInit]The first Petal function a filter must call, like PetalExit 'does the right thing'.
- [PetalPutItem]{ item index } stores the item at that index, in the destination track.
- [Pitch2Triad]{ pitch } returns the chord of which the pitch is the root.
- [SetDuration]{ item duration } changes the item's duration and returns the result.
- [SetKey]{ item } sets the current key (in terms of scale and tonality) according to the key change event present in that item.
- [TransposePitch]{ pitch tonality } transposes pitch to the corresponding tonality.

4.4 Questions, bug reports and stuff

All questions and bug reports regarding Petal should be sent directly to me (glaurent@worldnet.fr) or to the Rosegarden mailing-list where I'll answer.

While this first release should be considered Alpha, it shouldn't crash on you too often. You are encouraged to write your own filters and submit them to either me or the mailing-list. This way we'll eventually be able to provide more already written filters than just the current two, and further refinement of the Petal API will be easier to do (although drastic changes are very unlikely).

If enough filters are sent to me or the mailing list, I'll probably host them on my homepage so that everyone can take advantage of them.