

Audovia Documentation

1 Introduction

Audovia is a database application for making music on your Linux Ubuntu laptop or PC. Songs can have up to fifteen instrumental voices and a percussion track. Instruments can be chosen from either the default soundbank of 128 instruments or other soundbanks of your choice.

Songs can be developed, tested and edited very quickly and easily by virtue of the database structure and the **JFugue** MusicString notation. Notes within a MusicString are specified by their name and octave or by their MIDI value and their durations are specified either by character code, or numerically. You can use notes from C0 to G10, corresponding to MIDI values 0 to 127. Middle C is C5. Notes can be entered manually or by picking from graphic Treble, Alto, Tenor and Bass staves within the MusicString editor.

The opening phrase of Joy to the World can be written as:

```
C6h B5q. A5i G5h. F5q E5h D5h C5h.
```

A MusicString consists of one or more tokens separated by spaces, as above. Sequences of MusicStrings can be assembled into Patterns, which can be nested to any level. Songs can be constructed from Patterns either timewise or by voice.

The *File/Template* menu item creates song templates by voice where each voice is a Pattern containing other Patterns and MusicStrings. Bars (or measures) can be MusicStrings or Patterns. Pattern bars can be used to contain sequences of MusicStrings and/or Patterns.

All MusicStrings and Patterns within a song are given unique names which makes it easy to keep track of them within a composition. A MusicString can be shared between any of the Patterns in a song and it only needs to be edited once for the changes to be effective wherever that MusicString occurs within the song.

You can collaborate with your colleagues on a song by using a MySQL shared database. You can also share your songs by exporting and importing to and from XML files.

Audovia will play back your music and also export to MIDI, MusicXML and WAV files for music processing and music publishing. The WAV files can be opened in **Audacity**, then exported to MP3.

1.1 Background

Audovia is written in Java with a Swing forms user interface. It uses Apache Derby as the default database with an embedded JDBC driver. The music is generated by **JFugue**, a Java API for music programming.

Audovia was developed to produce backing tracks for playing along to. It is also useful for creating background music for videos or ringtones for mobile phones.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Audovia Documentation

2 Installation

To install and run **Audovia** on Ubuntu 16.04 or later, type the following commands at the terminal:

```
sudo snap install audovia
audovia.play
```

A desktop shortcut can be found at:

```
/snap/audovia/current/Audovia
```

For a quick start you can use *File/XML Import* and open the *Demo* folder, then select a song and *Import XML*. Then, from *File/Tree View* select "Song", press Play, then Default Soundbank.

Audovia stores its files in directories under:

```
~/snap/audovia/<version>/
```

2.1 Soundbanks

Audovia uses the Gervill synthesizer to generate sound. Gervill can use the default soundbank, TimGM6mb.sf2 by Tim Brechbill, or any soundbank with a .sf2 or .dls extension. These can be found by searching for SoundFont files on the Internet.

Soundbanks can be kept on file, in the SF2 directory, or you can use *Soundbanks/Manage Soundbanks* to upload these to the database.

2.2 Backing up your Songs

Use *File/XML Export* to export a song to an XML file or *File/XML Import* to import a song from an XML file. The entire song structure is exported or imported with the exception of any soundbank reference.

2.3 Shared Database (optional)

If you are using a Database as a Service (DBaaS) provider, then your hostname, port, database, username and password may be set up for you. In this case, use *Database/Database Connections* to create a new connection and enter the Connection Details. Please note that when entering the Connection Details, there should be no leading or trailing spaces in any of the fields as this will cause a communications failure. Then use *File/Create Tables* to create the database tables on the remote host.

If you are using MySQL as your shared database, we recommend that you create a user in your database called 'guest' with *EXECUTE* as the only privilege. Connecting as 'guest' enables read access to all songs and also the ability to create and log in to individual user accounts for the creation of songs. Full access to these songs can optionally be shared with selected other user accounts.

Audovia takes its default database configuration from *config/conf_audovia.xml*. If you would like to use any database other than Apache Derby as the default, you can change the parameters here. Template configuration files are provided for MySQL, Oracle and PostgreSQL.

Audovia Documentation

3 Tutorial

Audovia opens with the **Songs** form for managing your songs. If you are using numeric note durations within a song, use the **/durations** column to specify whether these are expressed as decimal values of a whole note or as pulses, where 32 pulses represents a quarter note.

To ensure that the different voices in a song are synchronized, all note durations should equate to a whole number of pulses.

Numeric durations should be used for triplets. A quarter note divided into three triplets should be given durations of 11, 11 and 10 pulses, or 12, 10 and 10 pulses if you want to give more emphasis to the first triplet.

To edit a cell, either select the cell and type something into it or double-click on the cell. To copy data in a cell, highlight the text to be copied and press Ctrl-C. To paste, press Ctrl-V. This only works if the cell is in edit mode (yellow).

The **Strings** button opens a form for defining the **JFugue** MusicStrings in a selected song and the **Patterns** button opens a form for declaring the Patterns in the song.

The naming of MusicStrings is independent from the naming of Patterns. You may find it helpful to name MusicStrings in lower case and Patterns with an initial capital.

Use *File/Template* to give you a start when creating new songs.

Use *File/Tree View* to display the structure of Patterns and MusicStrings in a selected song (see below).

3.1 Strings

The **Strings** form is used to define the MusicStrings in the selected song. MusicString notation is described in the next section.

MusicStrings can be edited within this form or, if they are more than two lines long, the **Editor** can be used to provide an editing window.

MusicStrings can be imported from other songs by using *File/Import Strings*. Libraries of MusicStrings can be built and used in this way.

3.2 Patterns

The **Patterns** form is used to declare the Patterns in the selected song. A Pattern is a container for other Patterns and MusicStrings.

The **Components** button opens a form for picking the Patterns and MusicStrings that are to be contained within a selected Pattern.

The **Play** button will play the selected Pattern.

The *File* options are *Export to MIDI*, *Export to MusicXML*, *Export to WAV* and *Clone*.

When exporting to WAV, you can set the amount of padding to be applied after a song to allow any reverb to die away.

If you want to create a Pattern that is very similar to an existing Pattern, use *Clone* and edit the clone.

Audovia Documentation

3.2.1 Example of Cloning

Suppose you have a song which is 128 bars long and the first 12 bars are repeated with a different 12th bar on the repeat.

Use the template to create a song where the voices have 2 parts with 128 bars per part. Clone each Part 1 and call this the Repeat. Insert the Repeat after Part 1 in the voice patterns. Delete bar 13 from Part 1 and delete bar 12 from the Repeat. Bar 13 in the Repeat becomes the different 12th bar.

You can keep the existing bar numbers and since bar numbers are independant for each part you can start Part 2 at bar 13.

3.3 Components

The **Components** form is used to pick the Patterns and MusicStrings that are to be contained within the selected Pattern and to specify their position in the sequence in which they are to be played. Picking is done via a drop down list which is displayed when you click on a **Component** cell.

The **Insert** and **Renumber** buttons can be used to insert components into an existing sequence.

The **Drill Down** button allows you to drill down to the child components of a selected Pattern. Drilling down on a MusicString will open a window for editing that MusicString. When selecting a component for **Drill Down** it is advisable to click on the **Position** cell otherwise the component pick list will be displayed.

If you would like to enter an anonymous MusicString into your Pattern, leave the **Component** cell blank and type the MusicString into the **Anonymous String** cell. This column can be made wider by dragging the boundary between the **Component** and **Anonymous String** headers.

3.4 Tree View

The tree view may be used to display the structure of Patterns and MusicStrings in a selected song.

Clicking on a MusicString will open a window for editing that MusicString.

Clicking on a Pattern will enable the **Play** button and also the *File* options: *Export to MIDI*, *Export to MusicXML*, *Export to WAV* and *Clone*.

3.5 MusicXML

If you export a Pattern to a MusicXML file, suitable for input to a music publishing system such as **MuseScore**, then for best results:

- Tempo should be the first token in the Pattern and each different Voice token should appear only once in the Pattern and be followed by one each of Instrument and Key tokens.
- Combined harmony and melody should be represented by separate voices. Notes connected to chords using the _ character will not be exported to the MusicXML file.
- Bar lines should be used.
- If numeric note durations are used they should match exactly to durations within the range whole note going down by halves to 128th note (dotted or undotted, but not a dotted 128th note).

Audovia Documentation

4 MusicStrings

4.1 Notes

A C Major scale of quarter notes, starting at middle C, can be written as:

```
C5q D5q E5q F5q G5q A5q B5q C6q
```

or as:

```
C5/0.25 D5/0.25 E5/0.25 F5/0.25 G5/0.25 A5/0.25 B5/0.25 C6/0.25
```

or, if you have **/durations** set to pulses, as:

```
C5/32 D5/32 E5/32 F5/32 G5/32 A5/32 B5/32 C6/32
```

In addition to the note letters, A to G, you can use R for a rest. Sharps, flats and naturals can be added by placing the character #, b or n immediately after the note letter so B-flat above middle C is written as Bb5.

Please note that, if you are transcribing music, accidentals in **Audovia** apply only to the immediately following note and not to the end of the bar as in conventional music notation.

MusicStrings can optionally be split into bars (or measures) by using the vertical bar character (|):

```
C5q D5q E5q F5q | G5q A5q B5q C6q |
```

As an alternative to note letters, MIDI values, enclosed in square brackets, may be used:

```
[60]q [62]q [64]q [65]q | [67]q [69]q [71]q [72]q |
```

4.1.1 Durations

The duration characters are:

w	whole note
h	half note
q	quarter note
i	eighth note
s	sixteenth note
t	thirty-second note
x	sixty-fourth note
o	one-twenty-eighth note

Dotted duration can be achieved by putting the period character (.) immediately after the duration character.

4.1.2 Chords

Chords are formed by adding the constituent notes together. A C Major chord can be written as:

```
C5q+E5q+G5q
```

4.1.3 Ties

Two or more notes of the same pitch can be tied together by using the hyphen character (-). Place the hyphen immediately after the duration of the note at the start of the tie and immediately before the duration of the note at the end of the tie. Notes in the middle of the tie have hyphens placed immediately before and after the note duration, as below.

```
C5q D5q E5q F5q- | F5-w- | F5-q G5q A5q B5q |
```

Audovia Documentation

4.2 Tempo

A tempo of 120 beats per minute can be expressed as:

```
T120
```

or as:

```
T[allegro]
```

Note the use of a predefined numeric constant within the square brackets. More tempo constants are available from *Insert/Tempo*.

4.3 Constants

Constants are defined using the \$ character followed by the constant name.

4.3.1 Numeric Constants

Numeric constants can be used anywhere that a number would appear in a MusicString. In addition to the predefined constants available from the *Insert* menu, you can define your own constants. For example, a scale with non-standard MIDI values could be defined as:

```
$A1=70
$G=68
$F=67
$E=65
$D=63
$C=62
$B=60
$A=58
```

and played as:

```
[A]q [B]q [C]q [D]q [E]q [F]q [G]q [A1]q
```

4.3.2 String Constants

Suppose you wanted to use the following arpeggio several times in your music.

```
F3i A3i C4i F4i C4i A3i
```

You could define a string constant as:

```
$arpeggioFoctave3=F3i~A3i~C4i~F4i~C4i~A3i
```

Then, in your music, you could refer to it as:

```
{arpeggioFoctave3}
```

Note the use of curly brackets for string constants.

4.4 Voices

Voices are specified by the V character followed by a number from 0 to 15. Note that V9 is the percussion voice and has its own set of instruments.

4.5 Key Signatures

Key signatures are specified by the K character followed by a note letter (or a note letter followed by # or b) followed by maj or min to indicate a major or minor scale so the key of G Major is written as KGmaj.

Audovia Documentation

4.6 Instruments

Instruments are specified by the I character followed by a number from 0 to 127. You can use *Insert/Instrument* to pick one of the predefined values.

4.7 MIDI Controller

MIDI controller events can be specified by the X character followed by the controller number followed by the equals sign (=) followed by a value. You can use *Insert/Controller* to pick one of the predefined controllers.

For example, if you want to set the volume of the current voice to a value of 12000, out of a possible 16383, use X[volume]=12000.

4.8 Pitch Wheel

A change of pitch can be specified by the & character followed by a number from 0 to 16383. This affects all following notes.

&0	lowers the pitch by a full tone;
&8192	returns the pitch to no change;
&16383	raises the pitch by a full tone.

5 Predefined Constants

5.1 Instrument Names

PIANO	0	GUITAR	24
ACOUSTIC_GRAND	0	NYLON_STRING_GUITAR	24
BRIGHT_ACOUSTIC	1	STEEL_STRING_GUITAR	25
ELECTRIC_GRAND	2	ELECTRIC_JAZZ_GUITAR	26
HONKEY_TONK	3	ELECTRIC_CLEAN_GUITAR	27
ELECTRIC_PIANO	4	ELECTRIC_MUTED_GUITAR	28
ELECTRIC_PIANO_1	4	OVERDRIVEN_GUITAR	29
ELECTRIC_PIANO_2	5	DISTORTION_GUITAR	30
HARPISCHORD	6	GUITAR_HARMONICS	31
HARPSICHORD	6	ACOUSTIC_BASS	32
CLAVINET	7	ELECTRIC_BASS_FINGER	33
CELESTA	8	ELECTRIC_BASS_PICK	34
GLOCKENSPIEL	9	FRETLESS_BASS	35
MUSIC_BOX	10	SLAP_BASS_1	36
VIBRAPHONE	11	SLAP_BASS_2	37
MARIMBA	12	SYNTH_BASS_1	38
XYLOPHONE	13	SYNTH_BASS_2	39
TUBULAR_BELLS	14	VIOLIN	40
DULCIMER	15	VIOLA	41
DRAWBAR_ORGAN	16	CELLO	42
PERCUSSIVE_ORGAN	17	CONTRABASS	43
ROCK_ORGAN	18	TREMOLO_STRINGS	44
CHURCH_ORGAN	19	PIZZICATO_STRINGS	45
REED_ORGAN	20	ORCHESTRAL_STRINGS	46
ACCORDIAN	21	TIMPANI	47
HARMONICA	22	STRING_ENSEMBLE_1	48
TANGO_ACCORDIAN	23	STRING_ENSEMBLE_2	49

Audovia Documentation

SYNTH_STRINGS_1	50	PAD_POLYSYNTH	90
SYNTH_STRINGS_2	51	POLYSYNTH	90
CHOIR_AAHS	52	PAD_CHOIR	91
VOICE_OOHS	53	CHOIR	91
SYNTH_VOICE	54	PAD_BOWED	92
ORCHESTRA_HIT	55	BOWED	92
TRUMPET	56	PAD_METALLIC	93
TROMBONE	57	METALLIC	93
TUBA	58	PAD_HALO	94
MUTED_TRUMPET	59	HALO	94
FRENCH_HORN	60	PAD_SWEEP	95
BRASS_SECTION	61	SWEEP	95
SYNTHBRASS_1	62	FX_RAIN	96
SYNTH_BRASS_1	62	RAIN	96
SYNTHBRASS_2	63	FX_SOUNDTRACK	97
SYNTH_BRASS_2	63	SOUNDTRACK	97
SOPRANO_SAX	64	FX_CRYSTAL	98
ALTO_SAX	65	CRYSTAL	98
TENOR_SAX	66	FX_ATMOSPHERE	99
BARITONE_SAX	67	ATMOSPHERE	99
OBOE	68	FX_BRIGHTNESS	100
ENGLISH_HORN	69	BRIGHTNESS	100
BASSOON	70	FX_GOBLINS	101
CLARINET	71	GOBLINS	101
PICCOLO	72	FX_ECHOES	102
FLUTE	73	ECHOES	102
RECORDER	74	FX_SCI-FI	103
PAN_FLUTE	75	SCI-FI	103
BLOWN_BOTTLE	76	SITAR	104
SKAKUHACHI	77	BANJO	105
WHISTLE	78	SHAMISEN	106
OCARINA	79	KOTO	107
LEAD_SQUARE	80	KALIMBA	108
SQUARE	80	BAGPIPE	109
LEAD_SAWTOOTH	81	FIDDLE	110
SAWTOOTH	81	SHANAI	111
LEAD_CALLIOPE	82	TINKLE_BELL	112
CALLIOPE	82	AGOGO	113
LEAD_CHIFF	83	STEEL_DRUMS	114
CHIFF	83	WOODBLOCK	115
LEAD_CHARANG	84	TAIKO_DRUM	116
CHARANG	84	MELODIC_TOM	117
LEAD_VOICE	85	SYNTH_DRUM	118
VOICE	85	REVERSE_CYMBAL	119
LEAD_FIFTHS	86	GUITAR_FRET_NOISE	120
FIFTHS	86	BREATH_NOISE	121
LEAD_BASSLEAD	87	SEASHORE	122
BASSLEAD	87	BIRD_TWEET	123
PAD_NEW_AGE	88	TELEPHONE_RING	124
NEW_AGE	88	HELICOPTER	125
PAD_WARM	89	APPLAUSE	126
WARM	89	GUNSHOT	127

Audovia Documentation

5.2 Percussion Names

ACOUSTIC_BASS_DRUM	35	RIDE_CYMBAL_2	59
BASS_DRUM	36	HI_BONGO	60
SIDE_STICK	37	LOW_BONGO	61
ACOUSTIC_SNARE	38	MUTE_HI_CONGA	62
HAND_CLAP	39	OPEN_HI_CONGA	63
ELECTRIC_SNARE	40	LOW_CONGA	64
LOW_FLOOR_TOM	41	HIGH_TIMBALE	65
CLOSED_HI_HAT	42	LOW_TIMBALE	66
HIGH_FLOOR_TOM	43	HIGH_AGOGO	67
PEDAL_HI_HAT	44	LOW_AGOGO	68
LOW_TOM	45	CABASA	69
OPEN_HI_HAT	46	MARACAS	70
LOW_MID_TOM	47	SHORT_WHISTLE	71
HI_MID_TOM	48	LONG_WHISTLE	72
CRASH_CYMBAL_1	49	SHORT_GUIRO	73
HIGH_TOM	50	LONG_GUIRO	74
RIDE_CYMBAL_1	51	CLAVES	75
CHINESE_CYMBAL	52	HI_WOOD_BLOCK	76
RIDE_BELL	53	LOW_WOOD_BLOCK	77
TAMBOURINE	54	MUTE_CUICA	78
SPLASH_CYMBAL	55	OPEN_CUICA	79
COWBELL	56	MUTE_TRIANGLE	80
CRASH_CYMBAL_2	57	OPEN_TRIANGLE	81
VIBRASLAP	58		

5.3 Controller Names

BANK_SELECT_COARSE	0	BALANCE_FINE	40
MOD_WHEEL_COARSE	1	PAN_POSITION_FINE	42
BREATH_COARSE	2	EXPRESSION_FINE	43
FOOT_PEDAL_COARSE	4	EFFECT_CONTROL_1_FINE	44
PORTAMENTO_TIME_COARSE	5	EFFECT_CONTROL_2_FINE	45
DATA_ENTRY_COARSE	6	HOLD_PEDAL	64
VOLUME_COARSE	7	HOLD	64
BALANCE_COARSE	8	PORTAMENTO	65
PAN_POSITION_COARSE	10	SUSTENUTO_PEDAL	66
EXPRESSION_COARSE	11	SUSTENUTO	66
EFFECT_CONTROL_1_COARSE	12	SOFT_PEDAL	67
EFFECT_CONTROL_2_COARSE	13	SOFT	67
SLIDER_1	16	LEGATO_PEDAL	68
SLIDER_2	17	LEGATO	68
SLIDER_3	18	HOLD_2_PEDAL	69
SLIDER_4	19	HOLD_2	69
BANK_SELECT_FINE	32	SOUND_VARIATION	70
MOD_WHEEL_FINE	33	VARIATION	70
BREATH_FINE	34	SOUND_TIMBRE	71
FOOT_PEDAL_FINE	36	TIMBRE	71
PORTAMENTO_TIME_FINE	37	SOUND_RELEASE_TIME	72
DATA_ENTRY_FINE	38	RELEASE_TIME	72
VOLUME_FINE	39	SOUND_ATTACK_TIME	73

Audovia Documentation

ATTACK_TIME	73	CHORUS_LEVEL	93
SOUND_BRIGHTNESS	74	CHORUS	93
BRIGHTNESS	74	CELESTE_LEVEL	94
SOUND_CONTROL_6	75	CELESTE	94
CONTROL_6	75	PHASER_LEVEL	95
SOUND_CONTROL_7	76	PHASER	95
CONTROL_7	76	DATA_BUTTON_INCREMENT	96
SOUND_CONTROL_8	77	DATA_BUTTON_INC	96
CONTROL_8	77	BUTTON_INC	96
SOUND_CONTROL_9	78	DATA_BUTTON_DECREMENT	97
CONTROL_9	78	DATA_BUTTON_DEC	97
SOUND_CONTROL_10	79	BUTTON_DEC	97
CONTROL_10	79	NON_REGISTERED_COARSE	98
GENERAL_PURPOSE_BUTTON_1	80	NON_REGISTERED_FINE	99
GENERAL_BUTTON_1	80	REGISTERED_COARSE	100
BUTTON_1	80	REGISTERED_FINE	101
GENERAL_PURPOSE_BUTTON_2	81	ALL_SOUND_OFF	120
GENERAL_BUTTON_2	81	ALL_CONTROLLERS_OFF	121
BUTTON_2	81	LOCAL_KEYBOARD	122
GENERAL_PURPOSE_BUTTON_3	82	ALL_NOTES_OFF	123
GENERAL_BUTTON_3	82	OMNI_MODE_OFF	124
BUTTON_3	82	OMNI_OFF	124
GENERAL_PURPOSE_BUTTON_4	83	OMNI_MODE_ON	125
GENERAL_BUTTON_4	83	OMNI_ON	125
BUTTON_4	83	MONO_OPERATION	126
EFFECTS_LEVEL	91	MONO	126
EFFECTS	91	POLY_OPERATION	127
TREMULO_LEVEL	92	POLY	127
TREMULO	92		

5.4 Combined Controller Names

(index = coarse_controller_index * 128 + fine_controller_index)

BANK_SELECT	16383	BALANCE	1064
MOD_WHEEL	161	PAN_POSITION	1322
BREATH	290	EXPRESSION	1451
FOOT_PEDAL	548	EFFECT_CONTROL_1	1580
PORTAMENTO_TIME	677	EFFECT_CONTROL_2	1709
DATA_ENTRY	806	NON_REGISTERED	12770
VOLUME	935	REGISTERED	13028

5.5 Values for some controllers

ON	127
OFF	0
DEFAULT	64

Audovia Documentation

5.6 Tempo Values

GRAVE	40	ANDANTINO	80
LARGO	45	MODERATO	95
LARGHETTO	50	ALLEGRETTO	110
LENTO	55	ALLEGRO	120
ADAGIO	60	VIVACE	145
ADAGIETTO	65	PRESTO	180
ANDANTE	70	PRETISSIMO	220

6 XML Specification

Here is the XML specification of the *.sbxml* files used in *File/XML Export* and *File/XML Import*.

```
<?xml version="1.0"?>
<songs>
  <song>
    <song_name>Song Name</song_name>
    <numeric_duration_type>decimal or pulses</numeric_duration_type>
    <components>
      <component>
        <component_type>pattern or string</component_type>
        <component_name>Pattern or String Name</component_name>
        <string_value>a JFugue MusicString</string_value>
      </component>
    </components>
    <pattern_components>
      <pattern_component>
        <pattern_name>Pattern Name</pattern_name>
        <component_position>an integer value</component_position>
        <component_type>pattern or string</component_type>
        <component_name>Pattern or String Name</component_name>
        <anonymous_string>a JFugue MusicString</anonymous_string>
      </pattern_component>
    </pattern_components>
  </song>
</songs>
```