Mixxx v1.4

DJ's manual

2004-October-8

Table of Contents

1.What is Mixxx?

Mixxx is software for DJ'ing. You can use wave based audio files, Ogg Vorbis and MP3 files as audio input. Mixxx can be controlled through the GUI and with external controllers including MIDI devices, joysticks and more. This release works on Linux, Windows and MacOS X (Recent version of MacOS X and at least G4 CPU).

To contact the developers of Mixxx, send mail to mailto:tuehaste@users.sourceforge.net or subscribe to mailto:mixxx-devel@lists.sourceforge.net and send a message to the list.

2.License

This version of Mixxx is released under the General Public License (GPL) version 2. A copy of the license can be found in the file LICENCE distributed with Mixxx, or at the website of Free Software Foundation: http://www.fsf.org/copyleft/gpl.html.

3.Installing Mixxx

MacOS X

Extract the downloaded file to a folder and start the installation by double-clicking on the program. Mixxx requires a recent version of MacOS X and at least a G4 CPU.

Windows

Start the installation by double-clicking on the downloaded program.

Linux

The easiest way to install Mixxx, is to use the binary version. If you have downloaded the binary version of Mixxx (i586), unpack the file and run the install script as root:

```
tar -xvzf mixxx-1.x-i586.tar.gz
cd mixxx-1.x
./install.pl
```

You will need to have QT installed already, but most distributions comes with QT.

The install script will install the program in `/usr/bin` with midi configs and skins in `/usr/share/mixxx`. The placement of the skin, keyboard and midi configs can be changed in the users `~/.mixxx.cfg` file created after the first startup of the program.

The binary version is not compiled with support for Jack. If you want to use Jack Sound Server you have to install Mixxx from source.

If you instead want to install Mixxx from source, you first need ensure that you have the following libraries and corresponding header files installed:

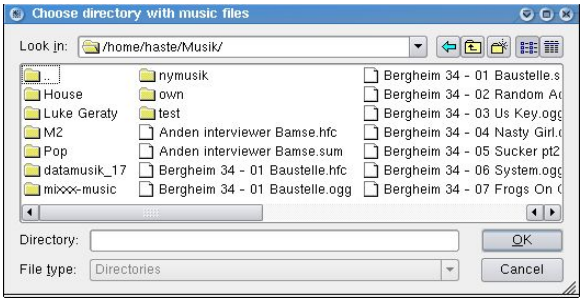
```
QT >= 3.1
libmad
libid3tag
libsndfile
vorbisfile
pkg-config
```

Download the source and run the following commands after you have unpacked the file:

```
cd src
./configure --enable-Jack
make
make install
```

This will compile and install Mixxx with Jack support. Use `./configure --help` to see other available options.

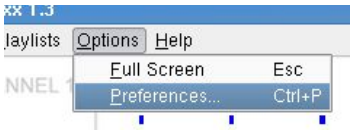
4. Setup



The first time Mixxx is started, you are requested to select a directory containing your song files. This directory is used as base, when later creating playlists in Mixxx. The directory can always be changed at a later time.

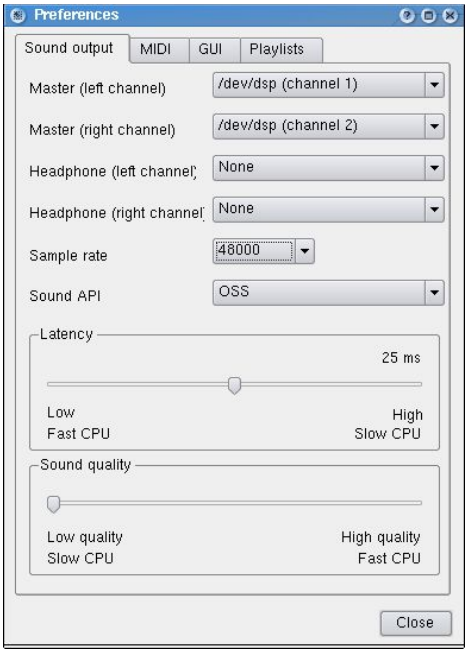
Note for MacOS X users: You will find mounted discs in the directory /Volumes.

Configure soundcard and latency



When Mixxx is launched first time it detects your installed sound cards, and tries to select a reasonable device for sound output. You can check the settings, by selecting Preferences from the Options menu.

In Mixxx you can use one sound card for output at a time. In general you will need a card with two stereo channels for live performance, one for Master output, and one for headphone output (cueing). If you practice you can work with one stereo card, mapping master output to the left channel of your card, and headphone output to the right channel. In this way you can work with both master and headphone output using a normal stereo sound card, although the output will be in mono.



Currently Mixxx does not support multiple sound cards open at the same time. The reason for this is mainly that it will increase the overall latency of the system to use several sound cards at the same time. Thus we recommend to use one multichannel card instead.

Furthermore, sample rate can be chosen, a value around 44100 is a reasonable choice (equal to CD quality).

With the latency slider you can adjust how long time it takes from you are changing a control in Mixxx until you will actually hear the result. Low latency is generally preferred. Not all sound API's and sound cards can run in low latency mode. Furthermore, to

use a low latency value, you may need a fast CPU. You can try adjusting the slider during playback. When set too low you will hear cracks in the sound output and the playback speed may slow down.

The sound quality slider adjust the quality of an interpolation algorithm used internally in Mixxx. In general this slider should be set to High, especially during live performance. You may try to set it lower if you have a slow CPU.

Finally the sound API can be chosen in the dialog. The choice of sound API depends on your platform. The following sections briefly describes the available options:

CoreAudio (MacOS X)

This is the default sound API for MacOS X. Low latency operation in the order of 10ms is possible using this API.

WMME (Windows)

This Sound API is most likely to work on all versions of Windows, however it does not allow for low latency operation.

Direct Sound (Windows)

DirectSound is usually a better choice than WMME. You should be able to achieve latency down to the range of 25-49 ms. It is important that you choose a non emulated DirectSound driver. If only emulated drivers are available for your card, you won't be able to achieve low latency operation, and you should probably use the WMME instead. To improve latency you may want to favor background processes under System in the Windows Control Panel.

OSS (Linux)

This is the most stable sound API on Linux. However, it does not always provide as good performance as ALSA and Jack. Is Jack doesn't work on your machine, OSS is likely to. To achieve low latency operation on Linux, Mixxx should be run as root, and use a low latency patched kernel

Jack (Linux)

To use Jack, you first have to setup the Jack sound server. The latency is chosen when Jack is started, and thus cannot be adjusted from the control panel of Mixxx. To achieve low latency operation on Linux, you should run both the Jack sound server and Mixxx as root, and use a low latency patched kernel. See documentation of Jack of how to setup the sound server.

ALSA (Linux)

Support for the ALSA sound API is preliminary, and should be considered ALPHA. ALSA has to be enabled at compile time, hereafter, you have to make a special device that Mixxx can use. Take a look at the file README.ALSA supplied with Mixxx.

Control devices

MIDI configuration

Selecting the “Input controllers” tap in the preference panel, you can change the MIDI device used to receive MIDI events, and also choose the mapping you want to use. A number of predefined mappings are available:

- MidiTech-MidiControl. This is a mapping that should work with most standard MIDI keyboards. You can use the pitch bend and modulation wheel to control Mixxx. Some of the tangents are mapped to push buttons in Mixxx.
- MixMan DM2. This mapping allows the use of the MixMan DM2 controller, a cheap DJ controller available at many stores, and at Ebay for around US $20. Note that you will need software to convert the USB signal to MIDI messages. Currently only a Windows driver, DM2toMidi is available for the device. For more information look here: http://www.pdoom.ch/dm2/ The midi directory also contains a definition file for DM2toMidi.

PowerMate

Powermates are supported under Linux and Windows. Currently only one PowerMate can be used at a time under Windows, but with Linux two can be used at the same time. For each Powermate a mapping can be chosen, either scratch or phase change mode. The main difference is that in scratch mode the controller has lower latency, while in phase change mode, the changes are more smooth.

Mouse

In Linux and Windows, Mixxx now has support for additional mice than the standard mouse pointer support. This means that additional mice can be attached and used to control playback. The mouse can be used on a table for scratching, but more importantly, it can be attached above a turntable platter, and used as a sensor for the turntable platter movement. In this way a DJ turntable can be used to control playback in Mixxx.

A mouse needs to be calibrated. This is done by moving the mouse or turntable at the speed that should equal normal playback speed, and the press the calibrate button. Continue to move the platter/mouse at the desired speed, until the calibration finshes.

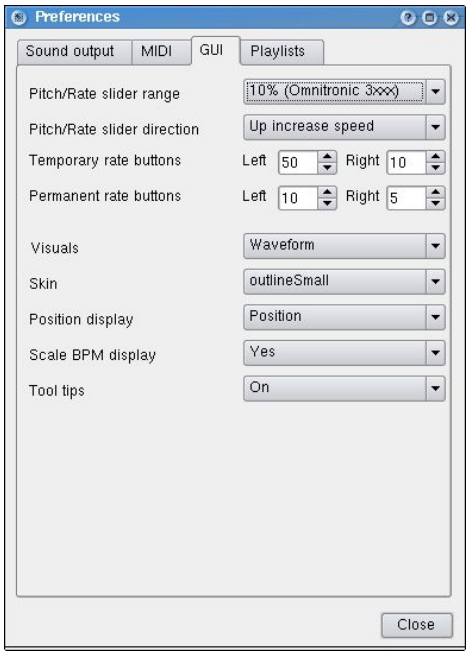
GUI configuration

Selecting the GUI tap in the preference panel lists a number of options controlling the Mixxx GUI. You have the option to control how much pitch shifting is allowed with the rate slider and the direction of the slider. Furthermore you can control the increment in speed by the temporary and permanent pitch buttons, but increasing or decreasing the values listed in the boxes. Both set of buttons allows for different values dependent on left or right clicking on the buttons.

Under Visuals you can select either Simple or Waveform. Waveform is the default and should be used if you have any decent graphics card with OpenGL. Most people have that today. On Linux, there can be problems with the configuration of the OpenGL driver, but Mixxx will give you a warning at startup if this is so.

You can select between various skins in the dialog too. It is possible to define your own skin. If you wish to do so, you should take a look at the files in the skin directory, to get an idea of what in involves, and furthermore you are welcome to contact the authors for further assistance.

Finally you can select weather the song timer should display position or remaining time, if the BPM display should show the BPM value of the song at normal playback rate, or at the current rate, and finally you have the option to enable or disable tool tips.



Playlists

Here you can select which directory contains your music files. This is the directory used in the file browser integrated in Mixxx.

Remote control

Mixxx can be controlled from the external media library software, Prokyon3. If you want to use Mixxx as a desktop media player, it is encouraged to use it with playlist software such as Prokyon3. The development teams of Prokyon3 and Mixxx plans to make an even stronger connection between the software in the future, enabling automatic cross fading between the two players.

5.DJ'ing with Mixxx

This section will give you an overview of how to use Mixxx for DJ'ing. Mixxx provides two playback devices, to which you can load and play tracks. By using the cross fader you can fade between the two tracks while they are playing.

Creating playlists

In Mixxx tracks can be organized into playlists. Mixxx can play the tracks consecutively, but the main purpose of the playlist, is to let you organize the tracks you use in a set into one compact list, where the tracks easily can be selected.

You can either create playlist through drag and drop or import them from WinAmp playlist format.

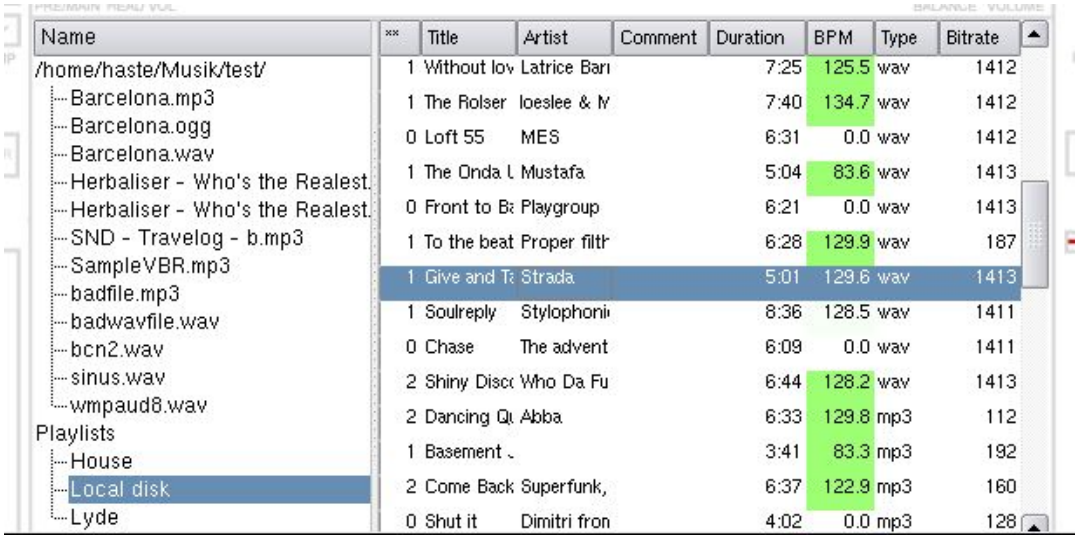


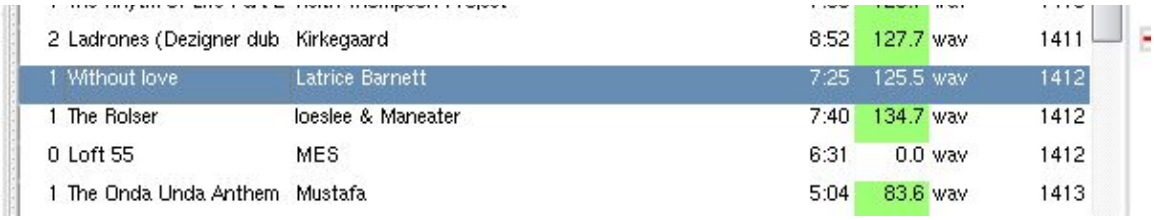
Illustration 1Shows the playlist interface in Mixxx. To the left is the treeview, and the right shows the track listing.

The screenshot above shows the playlist interface. To the left is the treeview, that allows you to browse the content of your song directory, and select active playlist. You can select a playlist by double clicking on the name, or by dragging the name to the track list on the right of the screenshot. By selecting a playlist the tracks are shown in the track list. You can also select the active playlist from the playlist menu.

By right-clicking on a playlist name, you have the option to delete or rename the playlist. To add tracks to a playlist drag song files from the tree view or an external file browser, to the tracklist (right).

Right clicking on a track in the tracklist, you can delete a track from the list. You can also add comments to each track by clicking in the comment field.

When you have loaded a playlist you can drag the vertical bar separating the treeview and the tracklist all to the left, giving more room for the tracklist.



New playlists can be created by selecting new in the playlist menu. The newly created playlist will appear in the treeview with the name “Default x”. You have to make it active by double clicking on it, before you can add tracks to it in the tracklist. You can also import existing playlist created in e.g. WinAmp or XMMS, by selecting Import.

Loading and changing tracks

Mixxx supports the following file formats:

- Wave (wav)
- Aiff (aiff, aif)
- MP3 (mp3)
- Ogg vorbis (ogg)

You load a track into one of the two players by right clicking on the track, and selecting the player you want, or by dragging the track to the waveform view of a player.

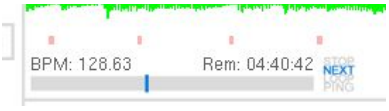
When a track is finished playing, it will either stop, load the next track in the active playlist, or loop the current loaded track. This can be selected by clicking on the widget next to the play position bar. The ping mode can currently not be selected.

Controlling the playback

Once a track is loaded, press play. You can *set a cue point* in the song by right clicking the play button. To do *cue preview* click the cue button, right click will do a *cue goto*. Rewind and fast forward will do what you expect them to do, right clicking goes to start and end of track correspondingly. During playback hold down the Rev button to do reverse playback.

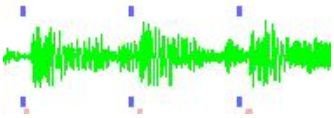
Use pitch slider to adjust pitch/tempo. Right click slider to reset it's value to 0%. The Perm buttons is useful for fine adjustments to the tempo, same as pulling the pitch slider slightly. By right clicking the buttons you get even finer adjustments. The temp buttons does the same, but only affects the slider while the buttons are hold down. When released, the pitch returns to its previous value.

You can also do temporary temp changes by dragging the waveform either backwards or forward. When not in playback mode, dragging the waveform is faster, for better allowing you to search through a track. You can also change the absolute play position of a track by dragging the play position slider.



By right clicking on the waveform displays you can get a zoomed in view of the waveform. This may be useful for precise cue point setting.

Beat mixing



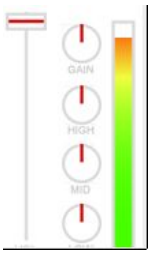
Mixxx provides some help to perform beat mixing. An automatic beat estimation is performed during playback of the track. After half a minute of playback the estimation may have stabilized. You may be able to see from the red or blue marks around the waveform if the estimated BPM is correct.

When the BPM value has been estimated in both players, you can sync one track to the other by pressing the Sync button. This will set the tempo of the song that is playing to match the tempo of the other song. Now you can adjust the phase of the beat by using the Temp buttons, or by dragging the waveform, before performing the cross-fade.

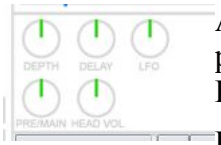
The estimated BPM value is furthermore stored in the tracklist. The background color of the BPM value in the tracklist, indicates the confidence of the value. If the field is nearly white background, it means that the BPM value is not to be trusted. If it is green it means that there is a higher probability that the BPM value is estimated correctly.

Filters, effects and headphone cueing

The main volume and balance is controlled by two knobs. Above them is shown shown a VU meter used in adjusting volume without causing distortion. To reset a knob to its default position, right click on the knob.



Volume for each channel can be adjusted using a slider. To adjust volume before entering the filters and effects, use the gain knob. The filter is operated by three knobs, controlling the high, mid and low frequency band.



A flanger sound effect can be activated for each player by pressing the corresponding Flanger button. The knobs Depth, Delay and LFO control flanger parameters.

Below the flanger knobs, are two knobs used to control the headphone channel. In general a player can either be in *normal mode* or in *pre-listen mode*, sometimes referred to as *cue mode*. When in normal mode, output goes to the main channel. When in pre-listen mode the output goes to the headphones. The mode is controlled by the Headphone button for each channel. The headphones are used to hear a mix between the main channel and the cue channel. The mix is controlled by the Pre/Main knob.

6. Controls

Mixxx can be controlled by mouse, keyboard, MIDI and other controllers. In the following sections each input type is described, along with instructions on how to change the mappings.

Keyboard shortcuts

Default keyboard shortcuts for Mixxx, is summarized in the table below. The shortcuts was defined by DJ^Disharmonic. You can download his music here: http://www.tanecni-olomouc.cz/disharmonic

Key	Function
Z (N)	Beat tap
G, H	Crossfader move left and right (soft step with SHIFT)
T, (Y)	Headphone toggle
5, (6)	Bass kill
4, (7)	Mid kill
3, (8)	Hi kill
2, (9)	Flanger
1, (0)	Sync
D, (K)	Start/Stop
SHIFT+D, (SHIFT+K)	Set cue
F, (L)	Cue goto
SHIFT+F, (SHIFT+L)	Cue preview
Q, (U)	Channel volume down
W, (I)	Channel volume up
A, (J)	Fast backward
S, (K)	Fast forward
Q,W,E,R,U,I,O,P	(reserved)
F1, (F5)	Permanent pitch down (fine with SHIFT)
F2, (F6)	Permanent pitch up (fine with SHIFT)
F3, (F7)	Temporary pitch down (fine with SHIFT)
F4, (F8)	Temporary pitch up (fine with SHIFT)

Keys in braces are for 2nd channel.

The shortcuts are defined in a text file, and can be changed by the user. The location of the file depends on the platform:

Linux: `/usr/share/mixxx/keyboard/Standard.kbd.cfg`

MacOS X: `<Mixxx bundle>/keyboard/Standard.kbd.cfg`

Windows: `<Mixxx directory>\keyboard\Standard.kbd.cfg`

Note: A Macintosh application bundle is opened by holding down Ctrl while clicking at the application program, and selecting "Show package content" from the pop up menu.

The format of the file is similar to the MIDI configuration files described in next section.

MIDI controllers

A The format of the file is really simple. There is four sections: `[Master]`, `[Flanger]`, `[Channel1]` and `[Channel2]`. For each sections a number of controls in the program can be assigned keyboard shortcuts.

Midi is configured in the same way as the keyboard. Different configurations can be saved in files ending with `.midi.cfg` and are placed in a subdirectory called midi, found the same place as the keyboard sub directory.

The format is basically the same as for keyboard configuration, but instead of the keyboard key, the midi key value is entered into the file along with the midi channel, as e.g. mapping the a midi key 36 at channel 6 to the play button in Mixxx:

```
[Channel1]
play Key 36 ch 6
```

For the master section the following controls are available:

```
crossfader
balance
volume
headVolume
headMix
```

These controls are all range controls, that is, they can be assigned an up or a down event, i.e. a certain key can be assigned to turn the volume up or down. That is done by adding `_up` or `_down` to the control value listed above. Thus to create a keyboard mapping for "p" turning volume up, and "l" turning volume down, the `Standard.kbd.cfg` file would contain three lines:

```
[Master]
volume_up p
volume_down l
```

The flanger uses these controls:

```
lfoDepth
lfoDelay
lfoPeriod
```

They are also range control, i.e. append `_up` or `_down` to them. For each channel the following range controls is available:

```
pregain
filterLow
filterMid
filterHigh
rate
wheel                   - Same as dragging the waveform with the mouse
playposSlider           - Seek file position
```

Furthermore a number of push button controls is available:

```
flanger                 - turn flanger on or off
filterLowKill           - kill buttons for filters
filterMidKill
filterHighKill
pfl                     - Headphone on/off
play
reverse
fwd
```

```
back
cue_set
cue_goto
cue_preview
rate_perm_down
rate_perm_up
rate_temp_down
rate_temp_up
```

Joysticks (Linux only)

The configuration goes into a midi configuration file like above but for Joystick MIDI channel 19 is used.

7. Development

Mixxx is developed in C/C++ using the QT toolkit. Development is primarily done by Tue Haste Andersen and Ken Haste Andersen, but a large number of developers are working on the development of Mixxx. This includes but is not limited to (in no particular order): Svein Magne Bang, Ludek Horácek, Lukas Zapletal, Jeremie Zimmermann, Ingo Kossyk, Gianluca Romanin, Kristoffer Jensen, Peter Chang and Berenger Enselme.

Furthermore Mixxx depends on a number of libraries. Thanks goes to all the creators of these great libraries which are used in Mixxx. Finally a great thanks to all the musicians giving feedback during development.

If you are looking for a special feature in Mixxx, or would like to see more development in an area, we encourage donations to the development team. This can happen through the Mixxx Sourceforge web page: http://sourceforge.net/projects/mixxx

If you want to help developing Mixxx you are most welcome. Please contact us. We are specifically looking for:

- Feedback from DJ's
- Developers
- Graphic artists
- Mac developer