1 K-Meter

Implementation of a K-System meter according to Bob Katz' specifications.

2 The loudness race

When comparing two similar pieces of music, the louder one is perceived as sounding better (although this is only true for short periods of time). Accordingly, the loudness of music productions has continuously grown during the last decades.

As maximum levels of records, tapes and digital media are limited, however, mastering engineers have started using sophisticated dynamic compression techniques to achieve higher loudness without distorting the music (as of 2010, distortion is increasingly being used in order to achieve even higher loudness).

Unfortunately, this decrease in dynamic range does not leave the music unharmed. Current compressed music blasts away your ears and makes you turn down the volume of your amplifier. Having lowered the volume, you'll find that the "better-sounding" compressed music suddenly sounds pretty dull and boring compared to uncompressed music. In contrast, music with high dynamic range makes you turn up the volume.

3 The K-System meter

The K-System meter has been devised by mastering engineer Bob Katz in order to counteract the ongoing loudness race and also help adjusting the levels of different songs during mastering.

4 Installation of pre-compiled binaries

In order to use the pre-compiled binaries, please install the *Fastest Fourier Transform in the West* library first (see http://www.fftw.org/download.html#platformspecific for instructions).

When you're done, simply extract the K-Meter files from the downloaded archive. For the VST plug-in, you'll then have to move the files to your plug-in folder (`~/.vst`, `C:\ProgramFiles\Steinberg\VstPlugins\` or the like).

5 Preparing to build K-Meter

To build K-Meter yourself, you'll first have to install the dependencies listed below. To compile on 64-bit GNU/Linux operating systems, you'll also have to install the multilib files for `g++` (Debian package: `g++-multilib`).

5.1 premake (required)

Version: 3.7
License: GPL v2
Homepage: http://premake.sourceforge.net/

5.2 premake4 (required)

Version: 4.3
License: GPL v2
Homepage: http://industriousone.com/premake

5.3 Fastest Fourier Transform in the West (required)

Version: 3.2.2
License: GPL v2
Homepage: http://www.fftw.org/

5.3.1 Installation on GNU/Linux

Extract the archive into the directory `libraries/fftw3`, change into this directory and run:

```
./configure --enable-float CC="gcc -m32"
make
mkdir bin/
mv .libs/* bin/
```

5.3.2 Installation on Microsoft Windows

Extract the source code archive into the directory `libraries/fftw3` and the archive containing the pre-compiled binaries into the directories `libraries/fftw3/bin` and `%SystemDirectory%` (usually `C:\WINDOWS\system32\`).

5.4 JUCE library (required)

Version: 1.51
License: GPL v2
Homepage: http://www.rawmaterialsoftware.com/juce.php

5.4.1 Installation on GNU/Linux

Extract the archive into the directory `libraries/juce`, change into the directory `libraries/juce/build/linux/` and edit the following lines in `juce_premake.lua`:

```
package.config["Debug"].target = "juce_debug32"
package.config["Release"].target = "juce32"
```

```
package.config["Debug"].buildoptions = \
  { "-D_DEBUG -ggdb -Wall -fPIC -m32" }
package.config["Release"].buildoptions = \
  { "-fvisibility=hidden -fPIC -m32" }
```

Finally, run:

```
chmod +x runpremake
./runpremake
make CONFIG=Debug
make CONFIG=Release
```

5.4.2 Installation on Microsoft Windows

Extract the archive into the directory `libraries/juce`.

5.5 Virtual Studio Technology SDK (VST, optional)

Version: 2.4
License: proprietary
Homepage: http://ygrabit.steinberg.de/~ygrabit/public_html/

5.5.1 Installation on GNU/Linux

Extract the archive into the directory `libraries/vstsdk2.4`.

5.5.2 Installation on Microsoft Windows

Extract the archive into the directory `libraries/vstsdk2.4`.

5.5.3 Building on GNU/Linux

After preparing the dependencies, change into the directory `build` and run

```
./run_premake
make config=CFG TARGET
```

where `CFG` is one of `debug32` and `release32`, and `TARGET` is one of `linux_standalone` and `linux_vst`. The compiled binaries will end up in the directory `bin`.

5.5.4 Building on Microsoft Windows

After preparing the dependencies, change into the directory `build/windows/vs_2010`, open `kmeter.sln` with Visual C++ 2010 and build the project. The compiled binaries will end up in the directory `bin`.

6 Final words

I love to hear from people who use my applications. So if you can spare the time, have any suggestions or want to create a nice theme, head over to my website www.mzuther.de and drop me an email!

And now again: thanks for using free software. I hope you'll enjoy this experience ...