K-Meter

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

Loudness race

Louder music is perceived to sound better, although that's only true for short periods of time. Due to this fact, the loudness of music productions has continuously grown during the last decades. As maximum levels of records, tapes and digital media are limited, producers and mastering engineers have to use compression to achieve higher loudness without distorting the music (actually, as of 2010, mastering engineers are already using distortion to achieve even higher loudness).

Unfortunately, an excessive increase in loudness leads to a decrease in the dynamic range. When you're listening to current compressed music, it 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" music suddenly sounds pretty dull and boring compared to uncompressed music, whereas music with high dynamic range makes you turn up the volume.

The K-System meter

The K-System meter has been devised by mastering engineer Bob Katz in order to counteract the ongoing loudness race.

Installation of pre-compiled binaries

If you use the pre-compiled binaries, simply extract the files from the downloaded archive. For the VST plugin, you'll then have to move the files to your plugin folder (`~/.vst`, `C:\Program Files\Steinberg\VstPlugins\` or the like).

Preparing to build K-Meter

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

premake (required)

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

premake4 (required)

Version: 4.2
License: GPL v2
Homepage: http://industriousone.com/premake

Fastest Fourier Transform in the West (required)

Version: 3.2
License: GPL v2
Homepage: http://www.fftw.org/

Installation on GNU/Linux

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

```
./configure --enable-float CC="gcc -m32"
make
```

Installation on Microsoft Windows

Extract the archive into the directory `libraries/fftw3`.

JUCE library (required)

Version: 1.5
License: GPL v2
Homepage: http://www.rawmaterialsoftware.com/juce.php

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
```

Installation on Microsoft Windows

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

Virtual Studio Technology SDK (VST, optional)

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

Installation on GNU/Linux

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

Installation on Microsoft Windows

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

Building on GNU/Linux

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

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

where `CFG` is one of `debug32` and `release32`, and `TARGET` is one of `linux_standalone` and `linux_vst`. You'll find the binaries in the directory `bin`.

Building on Microsoft Windows

After preparing the dependencies, change into the directory `build/windows/vc_2010`, open `K-Meter.vcxproj` with Visual C++ 2010 and build the project. You'll find the binaries in the directory `bin`.