IODA

interexchange format of wavetables between oscilloscope tools

DRAFT
March 12 2023, Hansi Raber

FILE FORMAT

The idea is to not invent a new format, but rather to repurpose the extremely wide-spread riff-wav standard and restrict it further in the following way:

1. IODA files are wav files with riff header, file extension ioda, and mimetype application/ioda-file
2. the files are 100% binary compatible with wav files and can be renamed to wav.
3. the only allowed data formats are 0x01 (little endian pcm) and 0x03 (little endian float).
4. format 0x01 must use signed 2- or 4-byte values (16 bit and 32 bit integers, short/int in java terms), and format 0x03 must use 4- or 8-bytes IEEE 754 values (32 and 64 bit floats, float and double in java terms)
5. only 1-6 channels are allowed (see below, with the ambiguity removed)
6. if there are multiple data chunks inside a file, then all chunks must use the same data format and channel count.

INTERPRETATION OF THE RAW DATA

The possible channel counts (1 to 6 channels) carry a direct, unambiguous, meaning as to how the data is to be interpreted.

1. 1ch => y wavetable
2. 2ch => x,y wavetable
3. 3ch => x,y,z
4. 4ch => x,y,z,z-modulation
5. 5ch => x,y,r,g,b wavetable
6. 6ch => x,y,z,r,g,b

CONSIDERATIONS

Appropriating wav has it's up and downsides, and on top of that there are considerations not even made yet. Here is a quick overview:

- extremely simple file format
- interaction with every DAW and synth out there
- multiple existing implementations in pretty much every programming language
- possibly hard to extend the format for future wishes
- handling animations can be tricky in a single file. first of all there is no concept of "fps" in wav, secondly many DAWs (iirc ableton) don't handle wav files with more than one data chunk properly.
- Along the same lines, parallel data (for multiple oscilloscopes at the same time) might face the same issues as the animation problem, and on top of that it might be tricky to distuingish the intent of „here is an animation“ from „here is data for multiple scopes“. A tiny bit of more reading of the riff-wav standard shoud yield an answer to these questions.
- Good performance, low overhead (even though the overhead could be reduced by at least 50% with a fully custom binary format)
- File Format note (1) („100% binary compatible“) will be a problem with many existing file processing tools, because the magic header will be riff-wav. Not sure what to do about this.