BGM loop points

BGM loop points allow music to repeat forever without breaks or seams, like real arcade games. The concept is simple: if your song schematically looks like

```
A B B B B ...
```

where `A` is an intro section (possibly not present), and `B` is a repeating section, then setting loop points at the beginning and end of `B` will allow `B` to repeat indefinitely.

Because there are many very different kinds of music file formats, there is no one way to specify loop points that is most natural for all of them. Therefore, exactly how the loop point parameters are interpreted depends on the specific plugin. For streamed audio formats like WAV, OGG, and MP3, loop points are most naturally specified in terms of samples of the audio stream. For tracker-type files like MOD or MIDI, a more natural way to specify looppoints might be in terms of measures, beats, or pattern numbers.

For streamed audio files, loop points can be identified by opening the file in an audio editor and looking for the precise point where the music repeats itself. Use your ear to get a rough location for the loop points, and fine-tune them visually by looking at the shape of the waveform at the loop start and end. If possible, you should put the loop points at a zero crossing in a quiet part of the music. If the loop end occurs exactly at the end of the file, then you can specify loopend = -1 as a shortcut. However, for lossily compressed streaming audio, it is not good practice to loop exactly at the end of the file, because audible artifacts may be produced. The best practice is to pick an internal loop, as indicated schematically in the diagram below:

```
A---- B---- B---- B--
           ^     ^
           |_____|
```

Another alternative is to straddle the beginning of B:

```
A---- B---- B----
         ^    ^
         |____|
```

Although the preceding basic description holds for all streamed audio files, the details are a bit more complicated for MP3s in particular.

Finding loop points for MP3 files

MP3 files can be tricky to work with because of encoder padding. Every MP3 encoder inserts some silence at the beginning and end of your audio when you encode it. This has two deleterious effects on BGM looping: sample numbers are different from the original audio (because of the start padding), and there will be a silent gap if you try to loop at the end of the file (because of the end padding).

Encoders based on LAME support gapless encoding, meaning that they write extra metadata to the file that gives the precise start and end of the true audio data. This allows compatible decoders to trim the silent padding when playing back the file. If you don't know if your file was encoded with LAME, you can use a LAME tag reader to see if the LAME tag is present. The mpg123 decoder supports gapless playback for files with gapless info stored in the LAME tag.

Audio editors may or may not recognize the LAME tag when you open an MP3 in them. You will need to consult the documentation for your editor, or do some experimentation, to find out what it does. If your file was encoded with LAME, but your editor does not support the LAME tag, then you must manually correct sample positions by subtracting the encoder delay. (Use a LAME tag reader to find the encoder delay.)

WARNING: some editors may attempt to remove encoder padding by detecting silence, rather than by reading the LAME tag. The results of this procedure are unpredictable, so you should not use such editors to find loop points.

The following scenarios are all possible with a gapless decoder like mpg123:

1. LAME-encoded, using original audio in editor: looppoints derived from original audio are reliable, loopend = -1 works without a gap.
2. LAME-encoded, using MP3 in editor, LAME-aware editor: looppoints derived from editor are reliable, loopend = -1 works without a gap.
3. LAME-encoded, using MP3 in editor, LAME-unaware editor: looppoints derived from editor must be corrected by subtracting encoder delay. loopend = -1 works without a gap.
4. Non-LAME encoded, using original audio in editor: looppoints derived from original audio are not reliable. The encoder delay must be added, if you have some way of finding out what it is. Using loopend = -1 will leave a gap. To eliminate the gap, you must find a loopend that cuts out the end padding.
5. Non-LAME encoded, using MP3 in editor: looppoints derived from the editor are reliable. Using loopend = -1 will leave a gap. To eliminate the gap, you must find a loopend that cuts out the end padding.
6. Using any MP3 in an editor that tries to detect silence instead of reading the LAME tag: never reliable!

With a non-gapless-capable mp3 plugin, only scenarios 4-6 are possible.