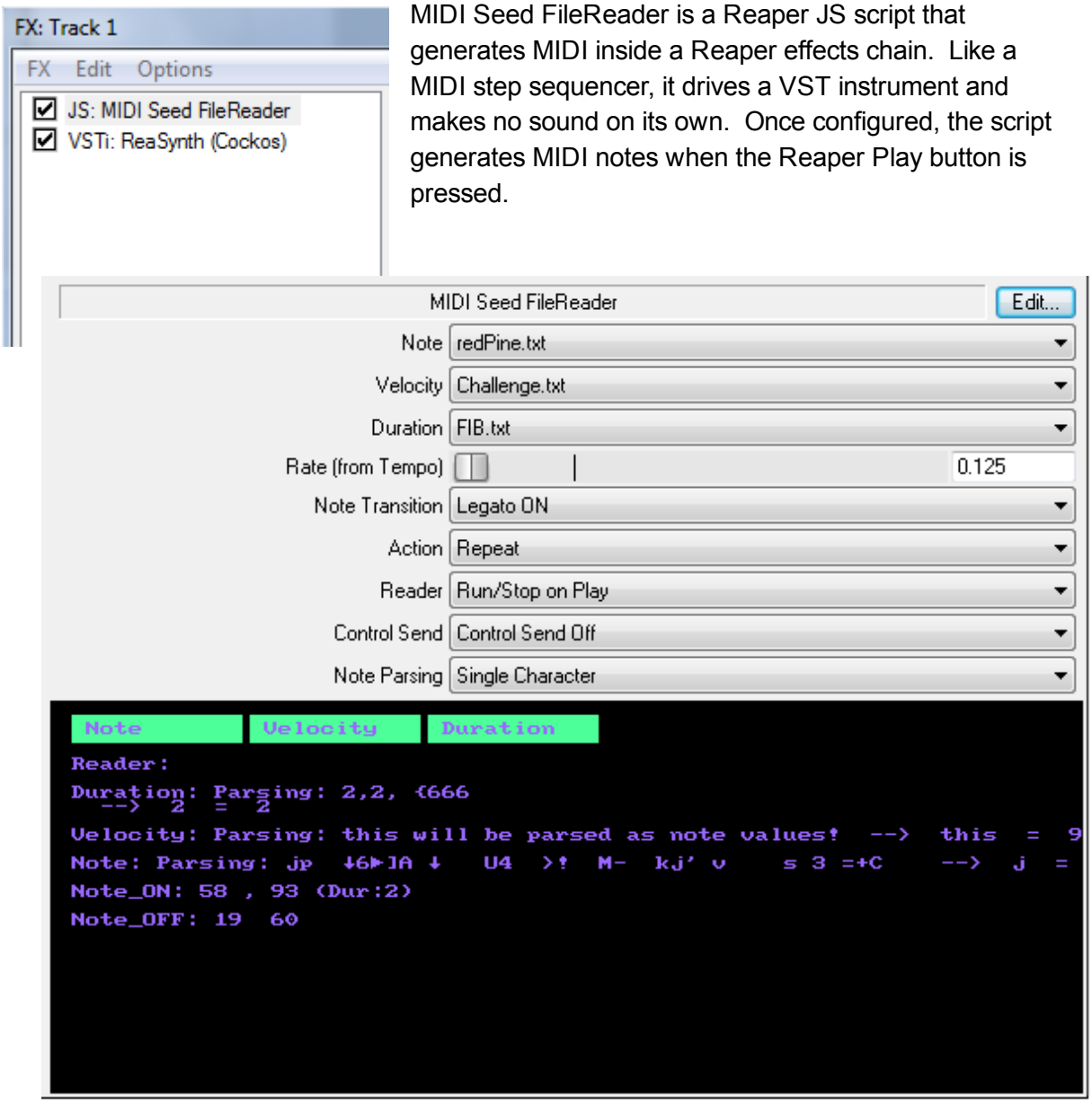
MIDI Seed FileReader

a JS File stream to MIDI script
Design: Pangaean Permafaction
Code: Nettles
GPL - http://www.gnu.org/licenses/gpl.html

Usage



MIDI Seed FileReader is a Reaper JS script that generates MIDI inside a Reaper effects chain. Like a MIDI step sequencer, it drives a VST instrument and makes no sound on its own. Once configured, the script generates MIDI notes when the Reaper Play button is pressed.

The MIDI is generated from values in the three **Note**, **Velocity** and **Duration** text files, chosen using the dropdown lists. These three files are located in the *Data* subfolder in Reaper's path.

- **Note** is the text file that contains the target pitches of the note.
- **Velocity** is the text file that defines the velocity of the note (0=silence=MIDI synonym for Note-OFF)

- **Duration** is the text file that defines the number of cycles to hold the note on for. This number counts down each cycle as set by the **Rate (from Tempo)** slider until it gets to zero (when it sends a MIDI 'note off'). Concurrently, the script moves on to the next set of values in the files — so sustained notes remain playing in the background as new notes are forming.
- **Rate (from Tempo)** — this slider is the base tempo value based on the Reaper BPM.
- **Note Transition** has two settings:
    - **Legato OFF** — any held notes are releasing before sending a new note-on.
    - **Legato ON** — the new note is sent immediately before any old notes are released. The delayed difference is extremely short, but may affect some synth's ADSR or mono/poly effects.
- **Action** has two settings:
    - **Sustain** — any held notes remain down for their entire duration.
    - **Repeat** — notes are repeatedly released and re-keyed from tempo for their entire duration. Any 'Legato' setting is only applicable to the final release.
- **Reader** has various settings to control start/stop interaction with play/stop of track transport; including an option to manually rewind all files.
    - **Run/Stop on Play** — The reader starts and stops along with Reaper's track transport play/stop. When stopped, all sustained notes are also released.
    - **Run/Stop/Rewind on Play** — The reader starts and stops along with Reaper's track transport play/stop; rewinding all the text files each time it stops. When stopped, all sustained notes are also released.
    - **Rewind All** — All text files are rewound for reading to resume from the start.
    - **Run Non-stop** — The reader always runs, including all sustained notes, regardless of track transport play/stop.
    - **Stop All** — The reader stops, including release of all sustained notes.
- **Control Send** has two settings:
    - **Control Send Off** — only MIDI note messages are sent, as previously described.
    - **GP_Slider_1/2/3** — note/velocity/duration values are echoed to MIDI control change messages, allowing external mix levels/filters/effects to be modulated in response to note events.
- **Note Parsing** has two settings:
    - **Numeric** — note, velocity and duration are interpreted as 'numeric' text; explained in detail later.
    - **Single Character** — velocity and duration are interpreted as 'numeric' text, but note value is assessed only a single character at a time.

All these settings can be automated using Reaper's envelope track and effect modulation capabilities. It can also be useful to combine the output with other MIDI filters to manipulate the range of notes generated. A MIDI monitor can help you understand what's going on, eg: http://rs-met.com/freebies.html

Understanding the three files

Each of the three values are read sequentially from the files; prone to various combinations of whitespace and/or punctuation as delimiters and converted to the range (0-127). Examples can be found in the supplied **TestData** files. Parsing activity of the file reader is displayed while the effect is running.
When the script reaches the end of a file, it independently cycles back to the top, regardless of whether all files are the same length. When the reader 'rewinds' a file, it will flash red in the activity display.
Changes to contents of the text files are picked up on each script run, and each time the script returns to the 'top' - meaning an external agent can update the files while the script is running. Addition of new text files to the data folder is only available in the dropdowns of effect instances created after.

- **Reading Values**
  The script reads the file as ASCII values. It finds 'tokens' to convert, based on surrounding whitespace and other delimiters. Each token is expected to be a three digit integer - so only three characters of a token will be converted. Longer numbers (or words) will have extra characters ignored. Shorter tokens (1 or 2 digits) are understood.
- **Where do the numbers come from?**
  Characters are examined starting from the righthand end of the token. '48' is subtracted to convert each character to a number. (48 because digits '0 to 9' are ASCII 48 to 57). Each resulting value is treated as a placeholder of 3 digits to multiply by 1,10,100 then summed together. This means various text and symbols can also create notes:
  - ", 123 ," = (49-48,50-48,51-48) → (1*100)+(2*10)+(3*1) = 123, as a MIDI note.
  - ", A ," = ASCII 65, so 65-48 = 17, meaning "A" = (65-48) → (17*1) = 17, and MIDI note 17 results.
  - ", CAT ," = (67-48,65-48,84-48) → (19*100)+(17*10)+(36*1) = 2106. The conversion proceeds, but such a big number is not usable, and will automatically be changed back to a range of 0 to 127, actually generating MIDI note 58.
  - In "**Single Character**" mode, velocity and duration are treated as above, but note values will be read left to right as single character tokens, simply subtracting ASCII value 48, without summing of multiple digits. This allows short spans of ASCII (eg: A...Z) to usefully map to a couple of octaves of adjacent notes.
    So ", CAT ," → notes 19,17,36 sequentially.
- **Delimiters**
  All characters with ASCII value less than 48 are considered delimiters. These include:
  `!"#$%&'()*+,-./<SPC><TAB><CR><LF>`
  This means that comma-separated lists are usable.
- **Line Breaks**

Line breaks are a special delimiter, because they force the conversion of a token. If the line is blank or has no preceding non-delimiter characters, an 'empty' token results. This special case is recognised, and a single blank line represents a 'rest' by default. No new note is played as the tempo cycle is counted.

- **Comments**
  Files can also contain comments; these are expected at the trailing end of a line, and are marked by any delimiter, then an open brace character "`{`". All text within the comment is ignored by the parser, but still displayed as the parser is running. A deliberate commented line could look like:
  `1,2,3,4,6,6!7+8  { Remind me where I put this in the file!`
  If a line has no preceding data, the comment character is bundled with any other characters for the parser to attempt conversion. For example, the following is not a properly commented line:
  `{ a line like this is parsed as note values!`
- **Errors and exceptions**
  Values greater than 127 are limited to 0 - 127 (value MOD 128). If no value is obtainable from a token/whitespace pattern, the parser returns exception code '-1'. Which is converted as follows:
  - Duration(-1) makes a duration of 0
  - Velocity(-1) makes a velocity of 80
  - Note(-1) makes a rest (skipped note)

Installation

Unpack the supplied Zip archive into your Reaper folder.
This will install required components, and add a 'project template' demonstrating the script.

You should find:

- Sample `*.txt` files in `C:\REAPER\Data\Pangaean` (obviously somewhere else if Reaper is installed elsewhere, but should be under `REAPER\Data\Pangaean`)
- The JS script file `MidiSeedFileReader` file in `C:\REAPER\Effects\Pangaean` (you can change this, so long as REAPER can find the effect when you 're-scan')
- A project file `MidiSeedFileReaderDemo.RPP` in `C:\REAPER\ProjectTemplates` (you can open this using file/project templates in Reaper, and customise it to suit your own project approach)

About Pangaean Permafaction

Cast in sediments of the last great flood, **Pangaean Permafaction** is a collective of experimental sound artists who may never physically meet. Samples of our work are available on Soundcloud. We create sonic fiction, sound art and soundtrack for radio and film using radical applications of acousmatic and concrete techniques, forming a rich rhythmic and harmonic palette that references and extends the stylings of contemporary electronic music. At home creating long form pieces fusing voice, sound and music, or glancing brief blows aside popular idioms.