DSpTracker Manual

Background & Overview

DSpTracker is a music tracker for the Nintendo DS. It draws inspiration other trackers that run on embedded hardware such as LSDJ and the Dirtywave M8, as well as from esoteric sequencing hardware such as the Monome Teletype and 100R's Orca. The synthesis techniques used here are inspired by techniques used by Ess Mattisson in the Elektron Model:Cycles.

The sound engine is an 8-voice phase modulation synthesis model, utilizing two operators per **Voice**. Phase modulation is a common way of achieving sounds similar to true frequency modulation synthesis on machines with limited processing power, and was perhaps most notably popularized by the Yamaha DX7.

The sound engine also includes a simple reverb model that utilizes 4 allpass filters whose delays in samples are set to coprime integers *(to avoid sympathetic ringing)*. I think it sounds pretty ok, but it still definitely rings a bit and could use some tweaking. It's easier than I'd like it to be to overload/clip it right now, so take care.

WARNING: *DSpTracker is capable of complex sequencing and while efforts have been made to keep it from eating up all of the Nintendo DS's CPU, it is very possible to build songs structures that will grind your system to a halt. DSpTracker is also under active development, which means that there* ***will*** *be bugs, and older save files may not be compatible with newer versions.* ***Please*** *back up your ROM with your song saves if you want to be able to load up older projects.*

I have used and loved many trackers, but this is the first one I've ever made. It takes a somewhat non-traditional approach to tracker-style sequencing. Some of these choices will feel really cool and fun, some of them may feel strange or inconvenient. If you have feature requests, suggestions, modifications, critiques, please feel free to open an Issue on Github, leave a comment on YouTube, or hit me up on Instagram. You can also find me on the Dirtywave Discord server and also a lot of places where synth hackers and live coders gather.

Functionality & Structure

DSpTracker represents music data in its **Sequencer**, which is comprised of 256 **Sequences** *(numbered 0-255)*, which in turn may contain an arbitrary number of **Columns**. Each **Column** may contain an arbitrary number of **Rows**. Each **Row** may contain a single **Command** as well as an 8-bit integer representing either one 8-bit **Value** or two 4-bit **Value**s.

Sequences, **Columns**, and **Rows** are processed left-to-right, top-to-bottom, lowest-to-highest at all times. **Sequences** can be triggered from other **Sequences**. This can be used as a technique to either create more complex meta-instruments or to construct songs from smaller sequences with as many layers of abstraction as you like. If you are familiar with LSDJ, consider that **Sequences** can be used like phrases, chains, or instrument tables.

NOTE: *The order in which you arrange your **Sequences** is important! Since **Sequences** are executed in ascending order, if you are triggering one **Sequence** from another, the **Sequence** doing the triggering should have a lower index than the one being triggered if you are concerned about accurate timing.*

By default, each **Column** advances one **Row** every 16 **Ticks**, though this can be customized on a per-**Column** basis. *(**Ticks** are the smallest unit of time understood by DSpTracker's **Sequencer**)*

About Commands

Each **Row** of every **Column** in a **Sequence** can contain a **Command**, which will be executed when the **Sequencer** steps to that **Row**. **Commands** are generally used to modify **Voice** parameters and to trigger **Voice**s to play notes/sounds, but they have access to most of the internals of DSpTracker and so they are capable of doing most anything.

A *Command* consists of a single-letter identifier and an 8-bit integer **Value** that is used as a parameter for the **Command**.

The current **Command**s supported by DSpTracker are as follows:

Commands:

N: Triggers the **Sequence**'s currently selected **Voice** and sets it to play the specified note. The first 4 bits specify the octave, the latter specify the note.

A: Sets the amplitude of the **Sequence**'s currently selected **Voice**.

E: Sets the decay of the envelope of the **Sequence**'s currently selected **Voice**.

e: Sets the attack of the envelope of the **Sequence**'s currently selected **Voice**.

M: Sets the modulation amount of the **Sequence**'s currently selected **Voice**.

m: Sets the envelope's effect on the modulation amount of the **Sequence**'s currently selected **Voice**.

F: Sets the modulator frequency of the **Sequence**'s currently selected **Voice**.

f: Sets the envelope's effect on the modulator frequency of the **Sequence**'s currently selected **Voice**.

B: Sets the feedback from carrier to modulator of the **Sequence**'s currently selected **Voice**.

b: Sets the envelope's effect on the feedback from carrier to modulator of the **Sequence**'s currently selected **Voice**.

p: Sets the envelope's effect on the pitch of the carrier of the **Sequence**'s currently selected **Voice**.

c: Sets the curve of the envelope of the **Sequence**'s currently selected **Voice**. First 4 bits affect the amplitude curve, second 4 bits affect the modulation curve.

R: Sets amplitude of the current **Voice** into the reverb mix.

V: Sets the **Sequence**'s current **Voice**.

J: Jump to the specified **Row** in the current **Column**.

S: Trigger a specified **Sequence**.

s: Set the row index that the `S` command will jump to when a **Sequence** is triggered.

T: Set the current tempo in BPM, based on a **Tick** rate of 16 **Tick**s-per-row.

I: Trigger the specified **Instrument**.

i: Set the **Sequence**'s current **Voice** to the parameters of the specified **Instrument**, but do not trigger a **Voice**.

Instruments

DSpTracker has a concept of an **Instrument**, which is actually just a collection of a subset of **Commands** and **Values** that can be applied to a **Voice** using the `I` and `i` **Commands**.

Instruments are an exception to the top-to-bottom, left-to-right, ascending execution rule in DSpTracker. **Instrument** parameters are applied right-to-left from how they are displayed in the **Instrument** editor **View**. **Instrument**s can be edited from the **Instrument** view either using the touch screen or buttons on the Nintendo DS.

Views

Views in DSpTracker provide different ways to observe and edit the state of various parts of the tracker. The current **View**s include a **Sequence** editor, an **Instrument** editor, a **View** for saving and loading a song file, and a simple oscillocope. The Nintendo DS has two screens, and either can have a single *View** loaded at a given time. Only one **View** can be active for editing with the Nintendo DS's buttons at any given time. The **View** on the touch screen can be interacted with using touch at any time. **Views** maintain their own state, so for instance, two **Sequence** views can be open at once, viewing different **Sequence**s.

Buttons & Combos

Change Page & Active Screen

SELECT + UP/DOWN: Change active screen **SELECT + LEFT/RIGHT**: Change **View** on active screen

Sequence View

UP/DOWN/LEFT/RIGHT: Move cursor

START: Play/stop **Sequencer**

B: Copy **Row**(s)

Y: Paste **Row**(s)

X + **UP/DOWN/LEFT/RIGHT**: Edit **Command** on current **Row**

A + **UP/DOWN/LEFT/RIGHT**: Edit **Value** on current **Row**

B + **UP/DOWN/LEFT/RIGHT**: Edit selection length

L + **UP/DOWN/LEFT/RIGHT/B/Y**: Move to, copy, paste **Sequence**

L + **START**: Play/stop current **Sequence** only

R + **UP/DOWN/LEFT/RIGHT/B/Y**: Move to, copy, paste **Column**

R + **START**: Restart **Sequence** at current **Row**

L + R + **UP/DOWN/LEFT/RIGHT**: Change column **Tick** rate

Instrument View

LEFT/RIGHT: Move cursor

B: Copy **Instrument**

Y: Paste **Instrument**

A + **UP/DOWN/LEFT/RIGHT**: Edit **Value** at cursor position

L + **UP/DOWN/LEFT/RIGHT**: Move to **Instrument**

R + **UP/DOWN/LEFT/RIGHT**: Move to **Instrument**