maxymiser

A MUSICIAN'S GUIDE

Contents

If you read nothing else, read this

Obviously, I hope all of this will be worth reading, but if you want to read as little as possible, and get started using the maxYMiser as quickly as possible, read these chapters:

Introduction

Who is this text for?

This text is for you. Assuming, of course, that you're interested in learning how to make music using Gareth "gwEm" Morris' Atari ST program "maxYMiser".

This text assumes that you have a little experience with trackers, that you understand hexadecimal notation (i e counting from 0 to 15 using the numbers 0 through 9 and the letters a through f), and that you know a little about music. I will not be using phrases like "phrygian cadences", but if words like "chord", "note", "pitch" and "octave" scare you, you might want to keep a friend (or wikipedia) close by while reading this.

This guide will not teach you everything about the maxYMiser. I strongly encourage you to read the maxYMiser documentation **and** all documentation that is recommended therein, but I hope that with this guide only, you should be able to get a feel for the program and hopefully finish your first maxYMiser song.

What does this text not claim to be?

This text is not a complete manual, or a reference to the maxYMiser, the YM2149 soundchip, the Atari ST or any Atari ST emulator for any operating system.
This guide does not tell you everything about the maxYMiser. The documentation that ships with the maxYMiser is to be considered mandatory reading.
Also, this text does not claim to be neutral or objective. I have tried to make it clear when I am conveying a personal opinion rather than undisputable fact, but I'm sure you'll find my own feelings on subjects shine through, and if this disturbs you, I apologize.

About the author

Per Almered is a 37-year-old songwriter/producer from Sweden. He's been making music on computers since 1984, and bought his first Atari ST in 1987. Between 1991 and 1995 he released 11 or 12 demos under the name "Excellence In Art (XiA)", and also did music and sound effects for UDS' pinball game Obsession, one of the last games (professionally) published for the ST.
These days he lives in Lund in the south of Sweden with his wife, two tortoises and a cat. He makes much too little money from the music business, but at least enjoys what he's doing, and sometimes he's even able to take the time to try and help out the "demo scene" community, that has given him so much over the years in terms of knowledge, enjoyment, friends and not least a wife (you collect coupons, and when you have 1500, you send them in, and a wife arrives in the mail 6-8 weeks later, I swear on my ukulele collection it's true). This document is one such little "giving back" project.

The maxYMiser website

The maxYMiser website, where you can find the maxYMiser and this guide is located at:
http://www.preromanbritain.com/maxymiser

Thanks

Thanks to gwEm for encouraging and proofreading this guide, and of course for making the maxYMiser in the first place.
Thanks must also go to Anders Eriksson (Evil/DHS) for luring me back to the Atari.

Concepts you should know (and understand)

Technical terms

A dollar sign ($) denotes a hexadecimal value. A “nibble” is a 4-bit value, or a single char in a hexadecimal number. For instance, in the hexadecimal number $c6, the “c” is the “high nibble” and the “6” is the “low nibble”.
Most values in maxYMiser are “signed” values. This means dual-nibble values between $00 and $7f are positive, and values between $80 and $ff are negative. So if, for example, you want to make a glissando going down as slowly as possible, you would enter the FX command “Hff”, where the “ff” part means $ff, i e “-1”. “fe” would mean “-2” etc.

Tracking in general

As you (hopefully) know from other tracker programs (or “trackers”), a song is made up from patterns, and a pattern is a sequence of notes and commands.
The tempo (although “speed” is a better word, to be honest) of a song is given in a value that’s usually between 3 and 7, meaning how many “ticks” (a tick is classically a 50th of a second) pass between each pattern row.

YM2149 specifics

Although I strongly recommend you read the YM2149 specifications, I suspect many of you are like me - you don't want to read more than is absolutely necessary, so here are some of the most important technical details you need to know about this soundchip:

Non-linear pitch The YM2149 doesn't have linear pitch. This means a vibrato with an amplitude of say, $10 will sound very different in the lower octaves than it will in the higher ones. The vibrato will be much wider, in cents, in the higher octaves. Put another way, the higher the octave, the smaller the pitch variations need to be to correspond to lower notes' pitch change in cents or semitones or whatever musical measurement of pitch you use. Put yet **another** way, if a portamento in octave 4 takes exactly one second to reach its target note 4 semitones up, it's not going to have arrived within once second if you use the same portamento value in octave 2.

Noise The noise generator is a single generator, shared by all YM hardware channels. So you can't play one noise in one channel and another noise on another channel. Higher channels have priority over lower channels.

Buzz sounds Buzz and SyncBuzz sounds use the YM2149's hardware envelopes (and I use the word "envelope" quite loosely, because these envelopes are really crap), and these envelopes have the disadvantage of not allowing any volume control ("crap" really isn't a strong enough word for these envelopes). Only one envelope can be active at a time, practically meaning you probably

won't use it in more than one channel. Higher channels have priority over lower channels. Buzz sounds are popular as bass sounds, and can create complex sounds at no CPU cost at all by combining them with a square wave. SyncBuzz sounds were, to the best of my knowledge, the first timer based effect Atari musicians used, and it's still useful (and used) today.

Timer tricks

Several of the "cooler" sounds created by the maxYMiser aren't made entirely by the soundchip itself. SID sounds, SyncBuzz sounds and DigiDrums (but not hardware samples) are created using timer tricks, meaning the CPU jumps in and modifies the sound, sometimes as often as several thousand times a second. If your music is to be used in a CPU critical environment, such as a demo or game, you need to discuss with the programmer what timer-based instruments your song can be allowed to use. Also, some timer tricks will be severely mangled if the demo or game uses the blitter chip available in several ST/Falcon models.

Specific to maxYMiser

Songs and patterns

A song in maxYMiser consists of a number of patterns. However, to save memory, each channel has its own pattern.
A pattern has 64 rows, and each row can contain a note, volume command and two FX commands, or any combination of these. Using the "Bxx" FX command, you can shorten a pattern.

Everything's a sequence

One of the basic building blocks in maxYMiser is the "sequence". A sequence is a list of values, nothing else. Whether a sequence controls

volume or arpeggio or YM mixer settings depends on where it's used. Volume, YM mixer settings, arpeggios, vibrato, noise settings and much much more are controlled by sequences. A maxYMiser instrument is really little else than a collection of different sequences controlling different aspects of the sound.

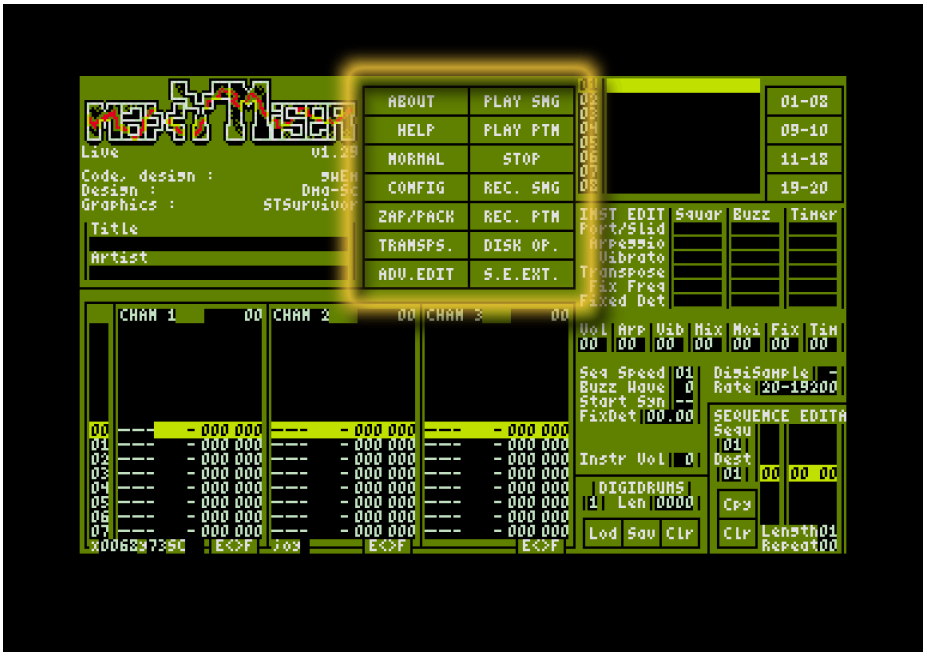
Sound channels

Depending on memory and CPU constraints, a maxYMiser song can use between 3 and 5 channels of audio. The first three are "YM channels", representing the three hardware channels of the YM2149 soundchip. The last two are "sample channels".
Sample channels use the STe DMA to play back samples (and thus won't make any sound on the ST, STF or STFM models). Used in native mode, the CPU cost is zero, but you're limited to playback in four sample rates: 6 258Hz, 12 517Hz, 25 033Hz or 50 066Hz. Also, in native mode, the replayer will reserve a bit of memory for different volume levels.
You also set the sample channels to "1ch" or "2ch", meaning 1- or 2-channel modes. Here, you can replay samples in seminotes (exact frequencies are found in the HELP page), and use the volume column without the extra memory demands of the native mode. However, the resampling involved will cost a bit of CPU.
Note that you **can** play samples in a YM channel too, using the DigiDrums mode, but your sample rate control is limited (so it's mainly useful for atonal material), and the CPU cost will be high.

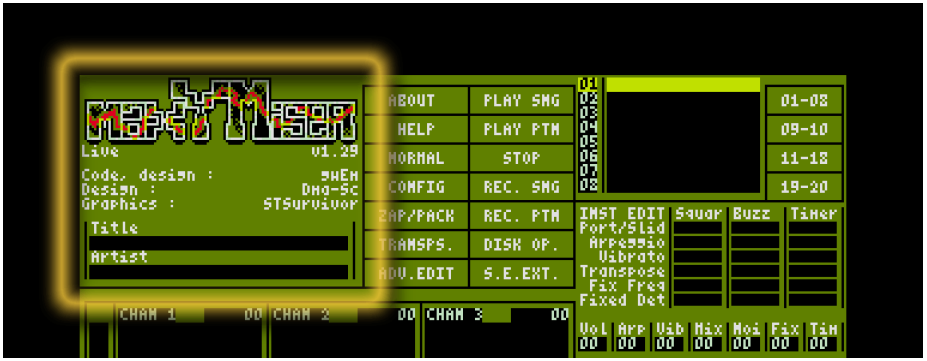
How things work

First of all, the maxYMiser's layout

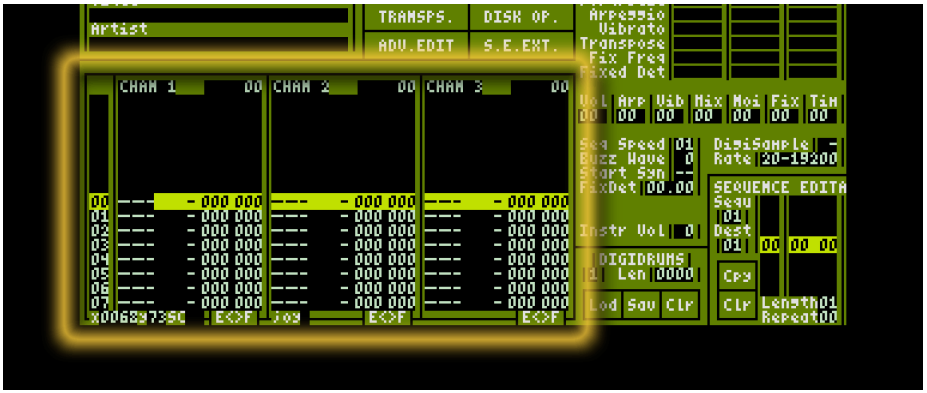
Just to make sure we're all on the same page when I start throwing around terms like "de-tune matrix" and "sequence editor", let me show you the basics of the maxYMiser's layout.



These buttons are the main dashboard. Here, you can play the current song ("PLAY SNG"), play just the currently selected patterns over and over again ("PLAY PTN"), access disk options to load and save songs and instruments, read the built-in help etc.



All the buttons in the left column, as well as the bottom two in the right column, open up pages in this area.



This is the pattern editor, visible at all times. In this screenshot, all you see is the three YM channels, but using the "Tab" key or the arrow keys to move to the right, you can access the two sample channels as well.

The HELP page is your friend

The HELP page is a brilliant little thing to have, especially if you're not in an emulator (where you can easily switch to documentation you may have open in other windows). Take a look at it, learn what information it contains (it even contains stuff that's not in the maxYMiser documentation!), and make a habit of using it.
The HELP page is accessed by clicking the "HELP" button, and clicking inside the text area makes it "Active", which means you can change pages using the up and down arrow keys.

Patterns

As mentioned earlier, a maxYMiser pattern contains data for one channel only.
A pattern row for a YM channel contains the following columns:

Note - Octave, Instrument, Volume, FX1, FX2

A pattern row for a sample channel contains the following columns:

Note - Octave, Sample, Volume

Common to both channel types, "Volume" is used as a negative offset value, meaning that if you enter a "3", the current volume output on that channel is reduced by 3.
The FX1 and FX2 columns (only available in YM channels) are three-nibble values, the first of which determines which FX command is used (and isn't really a nibble at all, it accepts any letter of the alphabet, although not all of them represent a command), and the second two depend on that FX command. One extremely powerful feature of the maxYMiser is the fact that it has two FX columns, which at times seems unnecessary, but really can save quite a lot of headaches at times.

FX commands

The most common FX commands (for a complete list, please refer to the maxYMiser documentation or the HELP page) are:

Bxx Pattern break. When the replayer encounters this, it immediately moves **all** channels to the next song position. Yes, you only need this in **one** of the patterns, which can save quite a bit of memory. Place the "Bxx" FX command on the last row you want to be played. So if you want to make a pattern play only its first 32 rows, you would place the "Bxx" command on row $1f.

Hxx Pitch slide. This command is new from v1.29 (which appeared as a beta while I was working on this document, hopefully it will be out as a stable version by the time you read this), and I personally drew a **huge** sigh of relief when it was introduced, because I had missed it badly. The current note's pitch simply starts sliding up (if "xx" is a value between $01 and $7f) or down (if "xx" is a value between $ff and $80 with the speed "xx".

Lxx Volume sequence. The volume sequence "xx" replaces the current instrument's volume sequence. The position in the volume sequence is ignored, by the way; the replayer starts playing the new volume sequence from start. Personally, I mainly use this as a "note off" command, to fade out the currently playing note's volume.

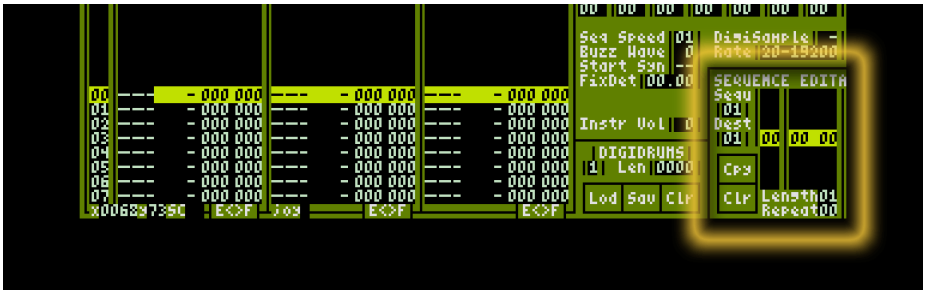
Pxx Portamento. When combined with a note, the pitch of the previous note glides towards the new note's pitch with a speed of "xx". If **not** combined with a note, **and** a portamento is currently playing, the command simply changes the speed of the current portamento.

Vxx Vibrato sequence. The vibrato sequence "xx" replaces the currently playing vibrato sequence, and starts playing from the start of the new sequence. Personally, this is how I add vibrato to a lead sound, because I usually don't want the vibrato playing from the start of the sound, but a bit in, and by adding the vibrato manually in the pattern, I get a lot more control. Also, I can use different vibratos at different times, either depending on the octave or the situation.

Zxx Demo synchronisation code. This command does absolutely **nothing** to your music, but if your music is to be used in a program, the programmer can read this code and let it control ...stuff. For instance, if you're making a demo, and the design document (you **do** have a design document, don't you? Hah!) states "logo appears on first drum beat", you can simply add a "Z01" command at the pattern row of that drum beat, and tell the programmer "at the first drum beat, I set the sync code to 01", and he or she will love you forever for making his or her job so much easier. Please note that a sync code of $00 is the default value, so make it a habit not to use the value $00; you still have a total of 255 different values to play around with.

Sequences

I **know** you're dying to get to making your first song by now, and to make a song you need to make a couple of instruments. But to make an instrument, you need to know how to make and edit a sequence. Sequences are edited in the lower right part of the maxYMiser screen, under the label "SEQUENCE EDITA".



The sequence editor sits happily in the lower right corner of the screen. It pays only a modest rent, and the landlady bakes scones for it every other sunday.

Under the label "Sequ" is a number field containing a hexadecimal value representing the currently edited sequence. Clicking the field lets you edit it, either by using the up/down arrow keys to scroll the value up or down, entering a new value using the keys 0-9 and a-f, **or** (and this is a **brilliant** feature!) pressing the "n" key to select the next unused sequence. You will use this shortcut a **lot**, trust me. Clicking outside of the number field or pressing the Return key (or the Esc key) leaves the field. In the same way, you can edit the "Length" number field (this determines how long the sequence is, max is $1f) and the "Repeat" number field (which determines where the sequence should jump to after reaching the end, it's basically a "GOTO" command).
The sequence itself is edited in the large section between the "SEQUENCE EDITA" label and the "Length" label. Click anywhere inside the sequence area to edit it, using the arrow keys to move around and the 0-9 and a-f keys to edit the values.

If you hold down the “Shift” key, you can use the up and down arrow keys to increase or decrease the value currently under the cursor.
While editing a sequence, you can change its length using the “Insert” key to insert a row, and the “Delete” key to delete a row.
To stop editing, click outside the area, or press the Return key (or the Esc key).

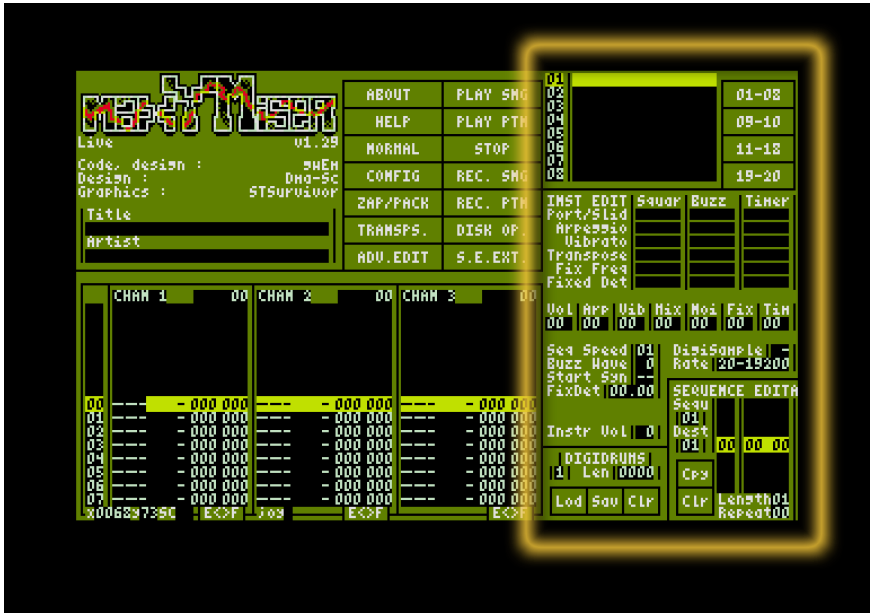
Instruments

As said elsewhere, there are two different types of channels in maxYMiser: YM channels and sample channels. And there are two types of instruments as well; YM and timer based instruments (or just “YM instruments”) and sample based instruments (or just “sample instruments”).
IMPORTANT: To hear a YM instrument when playing the Atari keyboard as a piano-style keyboard, the cursor needs to be on the note-octave part of a YM channel. To hear a sample instrument, the cursor needs to be on the note-octave part of a sample channel.
HOT TIP: You can connect a MIDI keyboard to your Atari and enter notes using that instead. And as if that wasn’t cool enough, you can even do this in the emulator Steem, which is just brilliant. See the maxYMiser documentation and if applicable, the Steem documentation, for more info.

YM and timer based instruments

The structure of an instrument

Instrument editing is done in the rightmost third of the maxYMiser screen.



This area of the screen is used for editing instruments and sequences.

The top section shows the instrument list, where you can change the currently selected instrument using the mouse, or the “+” and “-” keys of the numeric keypad.
Just below that is the de-tune matrix, which is a square of clickable flags that determine what features the current instrument can use, for what sound generator (Square, Buzz or Timer).

The de-tune matrix determines what features an instrument is allowed to use.

If the "Arpeggio" block isn't flagged for the correct sound generator, that instrument isn't going to be allowed to play an arpeggio, no matter if you give it a correct arpeggio sequence (however, you **can** set these flags using pattern commands, see the maxYMiser documentation for details). Below the de-tune matrix you find the seven sequence pointers, and this is where you decide what sequence does what:

Vol Volume sequence. The three first nibbles in the sequence are ignored, the fourth sets the volume value for the channel. $0 is the lowest, $f is the highest. Unless your sound is a Buzz or SyncBuzz sound (these types of sounds are always at maximum volume, $f, for technical reasons), you need to set a volume, or your instrument will be quiet. Too quiet...

Arp Arpeggio sequence. All nibbles are used, so if you want to use negative numbers, remember that -1 is $ffff, -2 is $fffe etc.

Vib Vibrato sequence. Similar to arpeggio, except values are hardware pitch offsets and not semitones.

Mix YM mixer sequence. This is absolutely crucial for sound generation. An empty sequence here is going to result in an instrument with no sound. Rather than explain all values here, I'm just going to give you the most common combinations,

and point you to the maxYMiser documentation (by which I also mean the HELP page, by the way) for an in-depth explanation.

10 00	Noise
01 00	Square
00 10	Buzz (requires you to set Buzz Wave to $8, $a, $c or $e)
01 05	SID (requires a timer sequence of "00 0f, 00 00")
00 1b	SyncBuzz (similar to buzz, but stays in tune in higher octaves ...at a CPU cost. It is similar to the "sync" function sometimes found in analog synths)
00 05	SID with custom waveform (last nibble in timer sequence gives waveform)
00 0d	DigiSample sound (choose sample with DigiSample field just below)

Noi Noise sequence. Sets the noise color of the noise generator. The first two nibbles are ignored, the range is $00 to $1f.

Fix Fixed frequency sequence. All four nibbles are used to set a fixed frequency.

Tim Timer sequence. Only the last nibble is used. For SyncBuzz sounds, you can give a sequence of up to 8 steps of different buzzer waveforms. For SID sounds (set mixer to 00 05), you can draw your own waveform (up to 16 steps; the more steps, the higher the CPU cost). Note that the Repeat field isn't used in a timer sequence; the replayer always loops back to the first value once it's reached the end of the sequence.

Below the sequence columns are a number of other settings. "Seq Speed" sets how many ticks (a tick is classically a 50th of a second, but you can change it in the "Interrupt" field on the Normal page) the replayer will wait between steps in ALL sequences. So you can't use it to slow down just a single sequence, this setting affects all seven sequence columns.

"Buzz Wave" selects which waveform will be used for Buzz sounds. Only the values $8, $a, $c and $e will produce anything else than a click. Notice that $8 and $c will often sound the same (as will $a and $e), but once you start playing around with SyncBuzz, there **is** a difference.
"Instr Vol" is a negative offset for the entire instrument, similar to if you use the volume column in the pattern to decrease the volume of the instrument. This can make life easier in several ways; thanks to this, you can design all your volume sequences to "max out" at volume $f, and then adjust down using this field. This also lets you reuse volume sequences for instruments with similar envelopes but different overall volumes.

Some ready-made instruments to play around with

Sine Bass - a complete type-in walkthrough

This is an instrument I like a lot, I call it "Sine Bass", and I thought I'd give you step-by-step instructions for how to type it in.
For this tutorial, you need to start with a completely empty maxYMiser. The fastest way to do that is to click the "ZAP/PACK" button, and on the screen that shows up in the top left corner, click the "ZAP" button in the top left of the matrix, representing "Devastate" and "All".
Ok, let's go:

1. The first thing I **always** do is to set the YM mixer, otherwise I tend to forget it. So, we need to create a new sequence: click the "Mix" field just below the de-tune matrix and press the "n" key. The maxYMiser now gives you the next unused sequence. The Mix field says "01", and the sequence editor displays your new sequence. Press Return or click anywhere once to leave the field.

2 Next we enter the YM mixer data into sequence $01. Click the sequence area (anywhere between the "SEQUENCE EDITA" label and the "Length" label) and change the last nibble into a "5". The sequence now reads "00 05", which means we're using a SID sound with a custom waveform. Press Return (or click anywhere) to finish editing the field.

3 Now we need to set up a volume sequence. Click the "Vol" field and press the "n" key to get a new unused sequence. Press Return (or Esc) to finish editing the field.

4 This sequence is going to be longer than a single step, so start by editing the "Length" field to say "06". Click the length field and either type in "06" or use the arrow key up to adjust the value to "06". Press Return to finish editing the field.

5 Now edit the volume sequence to read "00 0f, 00 0e, 00 0d, 00 0c, 00 0b, 00 0a" and press Return to finish the editing. So, that's a fairly quick volume change from $f to $a. However, if we leave it like this, with the repeat set to $00, it's going to go back to the first row when it reaches the last one, creating a machine-gun sound (...sort of), and we don't want that. So let's set the repeat to "05". Click the "Repeat" field and set it to "05". Press Return to finish editing the field.

6 Now we just need to enter the sample waveform. For a SID sound with a custom waveform, we place the waveform in a timer sequence. So, to create a new one, click the "Tim" field and press the "n" key to create a new sequence, followed by the Return key.

7 Enter a length of "10" for the new sequence, leave the repeat at "00" and enter the values "8, a, d, e, f, e, d, a, 8, 5, 2, 1, 0, 1, 2, 5" into the last nibbles of the sequence rows. Press Return to finish editing, and you have your first sound! Play around with it a bit and enjoy your hard work!

8 Now all you have to do is name the sound. Right-click the sound's (empty) name in the instrument list and enter a name you like (like "Sine Bass"). Also, save it to disk, using "DISK OP." and "Instrument" under the "Save" label. Now you can recall the sound into any song you feel you want to use it in.

Classic SID Lead

Here's a classic SID lead sound (although technically, it's "just" PWM, pulse width modulation, of a square wave).

Mix sequence: "01 05"
Vol sequence: "00 0a"
Tim sequence: Length 02, Repeat 00: "00 0f, 00 00"

Square wave major chord arpeggio, starting an octave up

Here's an example using arpeggio. Don't forget to set the "Arpeggio" flag in the "Square" column in the de-tune matrix!

Mix sequence: "01 00"
Vol sequence: "00 0a"
Arp sequence: Length 06, Repeat 03: "00 13, 00 10, 00 0c, 00 07, 00 04, 00 00"

Sample based instruments

Sample based instruments offer at lot less editing possibilities than YM and timer based instruments, and they can eat a bit of CPU (at least if you're not in native mode, see the previous chapter "Sound channels"), but they offer a lot of interesting possibilities. For instance, in my academy award winning song "Brit", I use sampled chords, which brings a new and interesting sound to the song, as well as saving YM channels.
A song can contain up to 8 samples, each with a maximum size of 32 kilobyte.
Under the label "DIGIDRUMS" at the bottom of, and near the far right of the screen, you can see the current sample's length, and load ("Lod"), save ("Sav") or clear ("Clr") the current sample.

The maxYMiser expects samples to be in 8-bit signed format. If you load a .WAV file (which needs to be 8-bit), it will sound nasty, because its sample data will be unsigned, and maxYMiser expects signed sample data. Also, the header information will make a tiny click in the beginning of the sound, which probably won't be noticeable on a crash cymbal, but can be **very** annoying on a bass drum. However, the maxYMizer documentation has a solution: go to the "S. E. EXT" page and click the "Sign/Unsign" button to recalculate the sample data from unsigned to signed format. Then set the "Start" field to "0044" to skip the file header.

One thing you will notice fairly quickly is that sampled instruments are fairly quiet. To combat this, the "S. E. EXT" page has "Vol" buttons to change the volume of the sample by (destructively) modifying the sample data. Personal reflection: And I do mean "destructively". The algorithm in maxYMiser won't hesitate to introduce digital distortion noise to your sound. If you have access to professional audio processing software, a good brickwall limiter will do a much better job. Even so, I generally don't use volumes higher than $a or $b in songs using sampled instruments, to give the sampled instruments a chance to be heard.

By the way, if you use non-Atari software to create or edit samples, remember the Atari ST only allows "8+3" filenames, meaning the file name has a maximum of 8 characters (spaces and several special characters aren't allowed), with an extension of up to 3 characters.

Pattern editing - a quick reference

A row in a YM channel consists of 5 items: note-octave, instrument number, volume offset, FX1 command, and finally FX2 command.
Removing the instrument number (set it to "00") means the note isn't retriggered.
The volume offset is a negative offset, meaning the value $4 decreases the volume by 4.
The two FX commands are identical, and the reason there's two of them is simply so you can combine different commands.
To get in and out of pattern editing mode, press the spacebar.
The "Keys" section on the HELP page contains a list of keys used for copying/pasting patterns/blocks etc.
When you're in a note position (the leftmost part of a channel's pattern), the letter-keys of the Atari keyboard turn into a two-octave piano-style keyboard, with the second-lowest row of keys (the one that spells Z X C etc on an english keyboard) and the row two rows above it (starting with Q W E on an english keyboard) representing the white keys. You can select octaves using the F1-F8 keys.
HOT TIP: You can use a MIDI keyboard too, check the maxYMiser documentation for details (this even works in the Steem emulator).

Song editing - a quick reference

The song editor is found in the “Normal” screen.

The song editor is where you put all your patterns together to make them into a song. Clicking on a pattern number in the song editor places the cursor there, and you can either type in a pattern number using the 0-9 and a-f keys, or press the “n” key to get the next unused pattern.
Using the “Tab” key or arrow keys you can navigate around the song, and the “Insert” and “Delete” keys inserts and deletes rows, respectively.
Holding down the “Shift” key allows you to step the current pattern number up and down using the up and down arrow keys.
To leave the song editor, press the “Return” key, the “Esc” key, or click anywhere.

Preset patterns

The song editor also has some interesting preset patterns:
Pressing the “Backspace” key while editing a position in the song editor places an empty pattern at that position.
The “CapsLock” key places a “Note off” pattern, which is technically identical to a blank pattern with a note off command at its first position.
The spacebar key places a “Looping pattern”, which is only used in Jam mode, which is outside the scope of this guide. In Normal mode, a Looping pattern makes the replayer simply skip that entire row of the song editor.

How do I do a...?

The maxYMiser does many things its own way, and while almost all the solutions are very good ones, I myself found myself scratching my head a couple of times while making my first songs in it. So this section is basically what I would have wanted during those first hours of playing around with the program. Hope it helps you too.

Note off

There a number of ways to make a note stop sounding. The built-in "note off" command is placed in a pattern using the "Caps Lock" key (technically, this does the same thing as putting a $f in the volume column, but at a lower CPU cost). However, this kills the volume immediately, which is rarely what I want. More often than not, I want the volume to fade out, and there are a couple of ways to do this. The easiest way would be to simply do a fade out using the pattern's volume column, but this only really works well for slow fade-outs (because you can't change the volume faster than the speed of the song). For fast fade outs, I use a separate volume sequence that I trigger using the FX command "Lxx". For a practical example, take a look at the chapter "A case study - a maxYMiser song".

Glissando a k a pitch bend

A glissando (or pitch bend) is when the pitch of a note rises or falls, usually more smoothly than semitone steps. In real life, a guitarist bending a string is an upwards glissando, and a violinist letting a finger slide up or down the neck is a glissando.
Glissandos in maxYMiser are done using the "Hxx" FX command. The "xx" value determines the speed of the glissando. Remember that values $00 to $7f are positive values, and that values $ff to $80 are negative ones, so to make a glissando going down as slowly as possible, use an "xx" value of "ff" (-1).
IMPORTANT: For an instrument to be able to use glissando, the "Port/Slid" flag has to be set in the de-tune matrix.

Portamento a k a note glide

A portamento is a pitch gliding between two notes, stopping when it hits the target note. (Technically, many working musicians make no difference between a glissando and a portamento, but in trackers, it is traditional to make the distinction that a portamento has a target note, at which the pitch stops changing, whereas a glissando never stops ...er, never stops ..."glissing".) A "true" portamento isn't possible on a guitar (although it can be argued "chromatic portamentos" are used in some musical styles), but on a violin it's when the player lets the finger slide across the neck of the instrument until it reaches the target note.
Portamentos can't be built into the instrument, it's one of those things you add in the pattern, using the "Pxx" FX command, where "xx" is the speed, "01" being the slowest.
IMPORTANT: For an instrument to be able to use portamento, the "Port/Slid" flag has to be set in the de-tune matrix.

Vibrato

Vibrato is an oscillating (or perhaps more clumsily, "vibrating") pitch.
In real life, a guitarist or violinist letting the finger on the neck of the instrument move slightly up or down the neck creates a vibrato.
In maxYMiser, vibratos can be done in several ways. You can have vibrato built-in in the instrument, you can add vibrato with the "Vxx" FX command (the sequence "xx" replaces any built-in vibrato in the instrument), or (and this is a lot more advanced and time-consuming, but sometimes very effective) you can "fake" a vibrato using portamentos or glissandos.
In the vibrato sequence (if that's the way you choose to go), remember that values from $0000 up to $7fff are positive, and values from $ffff to $8000 are negative, so the smallest possible vibrato in a sawtooth shape would be a sequence of "00 01, 00 00, ff ff, 00 00".
IMPORTANT: For an instrument to be able to use vibrato, the "Vibrato" flag has to be set in the de-tune matrix.
HOT TIP: While you **can** type in vibrato manually in a sequence, there's a little tool that makes it a **lot** easier.
On the "S. E. EXT" page, under the "SEQUENCE EDITOR EXTENSION" label,

you can type in minimum and maximum values for the vibrato. The "Oscillats" field ranges from $01 to $0f, the value $00 is displayed as "Ra", meaning "Ramp". Now, the numerical values are vibrato speeds, with $01 being the slowest, $0f the fastest (it sets how many times the oscillation will be repeated for the duration of the sequence), and Ramp mode meaning the waveform of the vibrato goes from one end of the range to the other, then snaps back to the start value again, like a sawtooth (the numerical values all use a triangle waveform). Below is a matrix of four buttons that modify the currently edited sequence in the sequence editor. The "Gen" column means "Generate" and simply overwrites whatever's in the sequence with new data. The "Mod" column means "Modify" and somehow modifies the existing sequence with what's generated (no, I can't really find a use for it, so I haven't bothered researching what it does **exactly**). The rows "Up" and "Down" determine if the generated oscillation should start at the minimum or maximum value, respectively.
IMPORTANT: Remember to set a length for the sequence before generating it, a sequence length of 1 won't create a vibrato.
In fact, when playing around with this function, I just set the length of my sequence to $1f and played with the "Minimum", "Maximum" and "Oscillats" values, and was a bit disappointed because the steps between "Oscillats" values are very coarse. However, by accident I discovered that you can use the length of the sequence to fine-tune the vibrato speed; lower lengths means higher vibrato frequency. Just remember to rebuild the vibrato when you change the length of the sequence. Very clever, very useful.
Personally, I always create vibrato with the "Gen/Up" button because I feel it creates a smoother transition and, to get rid of annoying glitches in the sound, I insert a number of rows at the beginning of the sequence to make a soft dip from "0000" to the minimum value. Also, I want the center point of the vibrato at 0000, so for a vibrato with an amplitude of 9, I would set "Minimum" to "fffc" and "Maximum" to "0004".
As a side note, a guitarist CAN'T bend a string negatively (well, Steve Vai probably can, now that I think of it), so if for some reason you want a vibrato that's "guitar correct", set "Minimum" to "0000" and put a positive value in "Maximum". (Nothing in this paragraph applies to tremolo arm vibratos, of course)

Arpeggio

Traditionally, arpeggio is a “broken chord”, meaning the harmonic effect of a chord is created by playing the chord’s notes in quick succession. In computer music, an arpeggio is one of the oldest ways in the book to have complex harmonies play out on limited hardware.

In maxYMiser, like almost everything else, an arpeggio is defined as a sequence containing note offsets, so to make a standard major chord, you make a sequence containing the offset values relative to whatever note the chord is played at, in this case “00 00, 00 04, 00 07”. A standard minor chord would be “00 00, 00 03, 00 07”.

There are three ways to make an instrument play an arpeggio in maxYMiser: you can either build an arpeggio sequence into the instrument itself, you can trigger an arpeggio sequence in the pattern using the “Axx” FX command (where “xx” is the arpeggio sequence), or (but this is limited to 2- or 3-step arpeggios) you can use the “Xyz” FX command in the pattern, creating the same effect as the arpeggio sequence 0, “y”, “z”. If “z” is $f, only the “y” value is used, creating a 0, “y” arpeggio. Please note that “y” and “z” can’t be negative values, and can’t be higher than $e.

Remember you can use negative values in the sequence. If it makes more sense to you to play a C chord at the note C in maxYMiser, but you want it in the order (highest-to-lowest-note) “E - C - G”. The arpeggio sequence would then be “00 04, 00 00, ff fb”.

Personally, I prefer playing arpeggiated chords from highest to lowest note, because the ear notices higher notes better than lower ones (making chord changes easier to spot), so to play a minor 6th chord, the note offset sequence I would use would be “00 09, 00 07, 00 03, 00 00”.

IMPORTANT: For an instrument to be able to use arpeggio, the “Arpeggio” block has to be set in the de-tune matrix.

Hammer-on or tapping

A hammer-on is when a note changes to a different note, but without retriggering the instrument. In real life, “hammer-ons” or “tapping” is when you play a new, higher, note on a vibrating string by placing a

finger on the fretboard, creating a new note without picking the string (if the new note is lower in pitch, it's technically called a "hammer-off" in guitarist language, because you lift a finger off of the fretboard).
In maxYMiser, a hammer-on is created by removing the instrument information from the new note. So yes, technically the instrument number, not the note/octave combination is what causes the replayer to retrigger the instrument. Please note that to remove the instrument you can't just place the cursor over the instrument number and press the "delete" key, that removes the entire note. Instead, just change the instrument number to "00".

Tremolo

A tremolo (when the word is used in tracking, for "real music" other definitions apply) is like a vibrato, but with the volume changing ("trembling") instead of the pitch.
In maxYMiser, a tremolo can be created either by building it into the instrument, or by triggering a new volume sequence in the pattern, using the "Lxx" FX command, where "xx" is the volume sequence to replace the instrument's own.

A case study - a maxYMiser song

To best illustrate a number of different techniques, I put together a maxYMiser song that I thought I'd walk you through and comment on what I do, and sometimes even **why**.

Before we get into the song, some general thoughts

I always leave pattern $00 blank, sometimes I add a "note off" command ("Caps Lock" key) at the top to stop notes "spilling over" from previous patterns. The default pattern at a new song position is $00, so having this pattern empty (or at least silent) will be a time-saver.

I often use vibrato, but I rarely put vibratos in the instrument themselves. I prefer setting them in the pattern, for a number of reasons (which are listed under "Vibrato" in the "How do I do a...?" chapter). However, that means there's a sequence unused by instruments, and if we press the "n" key to get the next unused pattern, that vibrato pattern is very likely to show up, and if you're not careful you might overwrite it with a volume sequence or whatever you're working on at the time. My solution is to use one or a few placeholder instruments, instruments that aren't ever used in the song but use these sequences, this will "protect" sequences from being presented as unused. To make sure these instruments are separated from the ones I **do** use in the patterns, I place them at the last instrument position ($20) and backwards.

Another "utility sequence" I often use is a "fade out" sequence used for cutting off notes. Sure, I **could** use the "note off" command, but that just kills the volume immediately. By building my own fade out, I can make a smooth fade out. It's a good idea to protect these sequences (you may need more than one, to fade from different instrument levels) by placing them in a placeholder instrument, as described in the paragraph about vibrato above.

The song

The song file is available at:

http://brainfish.net/atari/maxymiser/hitroad.snd

This song is a cover of “Hit the Road Jack” by Percy Mayfield (although I don’t believe he ever recorded it himself. Arguably the best-known recording is the one Ray Charles did in 1961). My arrangement is crappy, predictable and not very interesting, but it’s not here to give you a cultural experience and enrich your life, we’re here to learn, ok?

Ok, so load the song in maxYMiser by clicking the “DISK OP.” button, and then the “SNDH” button below the “Load” label. Once it’s loaded, click the “NORMAL” button to see the song list that shows what order the patterns play in. Click the “PLAY SNG” button to listen to it at least once. It’s only just over 38 seconds long. If you want to hear just a single channel, you can mute and unmute channels using the buttons marked “Chan” 1-3, “STe” and “DMA” (although the last two aren’t used in this song).

These buttons mute and unmute the maxYMiser’s 5 channels.

As you will quickly notice, the first channel plays bass, the second plays lead, and the third plays chords.

The instruments

The song contains 7 instruments, one of which is a placeholder instrument ("PlaceholderInst" in instrument slot $1f). The first instrument, "TwinSaw Bass", is a SID instrument with a custom waveform designed as two sawteeth, the second quite a bit lower in amplitude. Instrument $02 is called "PWM Lead", and is a standard SID sound with a one octave higher single "blip" at the start, faking a transient. Notice that the volume envelope reads "$8, $a"; the little volume dip is there to de-emphasize the one-octave transient a little.

Instruments $03 to $06 are the different chords used. The chords all use the same Mix and Vol envelopes, the only difference between them is the arpeggio sequences used (which, by the way, is the same as the numbers in the name of the instrument, to make my life easier). Also note that the volume sequence is just a single "$8", all volume changes are done in the pattern. This way I can use the same instruments for the choruses and the verses. If I had been **really** tight for instrument space, I could have made just a single instrument for all the chords, and used the "Axx" FX command to make it arpeggiate. It would, however use up an FX column, and make the pattern a little harder to "read" while editing.

The song structure

As you can see, the bass channel uses the patterns $01, $0a and $0b. $01 is the default one, used in the choruses, $0a is the verse bass, and $0b is identical to $0a, except for the little "riff thingie" at the end of the verse, just before it loops back into the choruses.

The melody line uses the patterns $02 through $06, although $02 is really just a lead-in for pattern $03.

The chords use the patterns $07, $08, $09 and $0c. $08 is the staccato chorus chords and $07 is identical to it, but with the first half empty, for the intro section of the song. $09 and $0c, the verse chords, are identical except for the little "riff thingie" at the end of $0c.

This song doesn't use the sample based channels, and that's why those patterns are all $00. (in fact, why not familiarize yourself with the sample based channels by adding drums yourself?)

A closer look at a couple of the patterns

You may notice the "Speed" field at the top of the "Normal" page changes all the time, this is a trick to introduce triplet feel into the song. The bass patterns contain alternating "S08" and "S04" commands, setting the "Speed" field to $8 and $4 respectively, obviously.
Go to song position $00 using the arrow up and down buttons next to the song list on the Normal page. Using the "Chan 2" and "Chan 3" buttons, mute those two channels and click the "PLAY PTN" button to hear that pattern over and over. As you can hear, a number of the notes are cut off using an "L04" command to trigger the volume sequence $04, which quickly drops the volume to 0. It would be possible to simply drop the volume to 0 using either the "note off" command, or insert an "f" in the volume column, but I think this sounds more natural (and you agree, yes you do).

The melody

Ok, on to the melody, which is where most of the action is. Go to song position $01, make sure channels 1 and 3 are muted, and channel 2 isn't, so you can hear the melody line alone using the "PLAY PTN" button.
Now this is where things get interesting, with lots of articulation. The first note consists of **two** notes and three FX commands. What I'm doing is a C in octave 5 (instrument $02) gliding upwards to the D, and then vibrato is added.
Row $00: The C note is set, and the "H03" command (pitch slide) is used to make the pitch rise at speed $03.
Row $01: The D note is set, but without the instrument number, to prevent it from retriggering. Here, the portamento FX command is used to finish the pitch glide. Now, it **might** be possible to ignore this step, if you could find an exact number for the previous row's "Hxx" that will land on **exactly** the correct note, but for some notes it wouldn't be possible at all, and at any rate, it will take a **lot** of testing to get it right. (Oh, and I suspect you're wondering how to set a note without instrument number? Set the note as usual, then edit the instrument number to "00".)
Row $02: Vibrato sequence $08 is triggered using the "V08" command. If you load up sequence $08 in the sequence editor, you'll see I've inserted

three rows at the start of it, to make the transition from no vibrato to vibrato smoother (yes, it was clearly audible and very ugly).
Ok, so that's the **first** note... Are you sitting comfortably? 8-)
Row $08: To stop the currently playing note, as discussed earlier, I trigger volume sequence $09 using an "L09" command, but I also drop the pitch fairly quickly with an "HF0" command (remember, values between $00 and $7f are positive values, with lower values being slower, and values between $ff and $80 are negative values with $ff being -1, $fe -2 etc).
Row $0f: Again, the "L09" command is used to terminate a note manually.
Row $14: Here, I do the same C-to-D-pitch glide thingie as at the start of the pattern, but since the previous note was a C, I just have to use the portamento command to make the pitch glide up to the D.
Row $20: To make the volume fade up from a lower volume than the instrument's own, I do a manual fade using the volume column (remember, the volume of the channel is subtracted with the value of the volume column). If I wanted it any faster I would have had to use a volume sequence, but this sounds good.
Row $26: Yet again, I end a note with a pitch glide down, but with only a small decrease in volume, so I choose to do it using the volume column.
Now on to song position $02:
Rows $03-$04: Here I want to fake the effect of a guitarist slightly touching and immediately releasing the tremolo arm, which I do using a portamento. Notice I could use just about **any** note below C#5, because the portamento doesn't have time to reach even a C 5 anyway before row $04, where the new portamento sets the pitch on its path to D 5. Neither of these two notes have instrument numbers, of course, because retriggering would completely spoil the effect.
Rows $24-$3c: Here, I do a really really slow fade out using the volume column.
There's just one last thing I'd like to show you, at song position $03:
Rows $20-21: Here, using the portamento command, I create a slightly blue note by first gliding fairly quickly from the A 4 to the E 5, and then slowing the portamento down to a grind. And as you probably guessed, this took a couple of tries to get right.

The chords

And finally, the chords. Go to song position $01 or $02 (either is fine, the chords play the same pattern), mute all channels except the third.
Rows $03-$05: So I can reuse the chord instruments, I choose to do the fade out of the chorus chords using the volume column. It is **not** as smooth as it would have been if I had created a volume sequence to do it, but with arpeggios, the ear is much less discriminating, I find.
Song position $03:
Row $00: Not only are the verse chords played an octave lower than the chorus chords, they are also static in volume. To push them further back in the mix, I use the volume column to lower their volume.

Composing using an emulator

As always, there are advantages and disadvantages to using an emulator. The advantages are very tempting; you don't need to bring your old ST out of storage and find a TV set for it (Not to mention disks, where **do** you find 3.5" disks costing less than a cornea transplant these days? Also, floppy disks are **not** very reliable.), you can use your existing computer setup, meaning everything you do on the Atari is backed up (yes, you have backup routines, it's not the 1980s anymore), and your virtual ST is simply always available in a window.
The disadvantages depend to a large degree on what emulator you use, but obviously, you need a computer fast enough to run an emulated Atari (but even my cheap EEE is fast enough for that). Also, sound emulation seems to be the last thing emulator authors get right (and for good reason, there's little point in having perfect sound emulation if the other parts aren't working well). And there's always a bit of confusion when the machine running the emulator has a keyboard that's different from the original machine's.
Personally, I have never used maxYMiser on a real ST, because I feel Steem Engine v3.2 does a fantastic job on all the machines I own. And I feel that I know the YM2149 soundchip well enough (I've made music on it since before the Atari ST series even existed) to claim that Steem's

emulation of it (and the tricks the maxYMiser can throw at it) is as close to perfect as I could ever wish.
Since Steem is the only Atari ST emulator I've used long enough to claim I know it well, that's the one I'm going to discuss in this short chapter, but there **are** others, and you might very well be happy with one of those.

Steem and maxYMiser

I'm not going to teach you the basics of Steem and how to use it, there are resources out there that do this very well. I will, however, point you to where you can find it:

http://steem.atari.st

...and I'm going to give you some quick "get started" tips:
First of all, Steem supports making a folder on your harddisk an Atari harddisk. Use that feature. You **can** work with floppy images, but it's messy an unpractical. The folder Steem uses is accessible from your regular OS too, so you can use that OS's features to manage files (copying files, making folders etc was **not** as easy on the Atari as it is in modern OS's).
Second, give the Steem process high priority; the first thing that suffers when an emulator runs out of CPU time is sound. In the Win32 version of Steem 3.2, there's an option in the "General" section of the Options windows (accessed via the little monkey wrench/adjustable spanner in the top row of the Steem window) that says "Make Steem high priority". Make sure that's ticked-in.
And third, check the Sound section of the Options window. For the most honest representation of the sound, set "Output type" to "Direct". Also, to avoid unnecessary resampling, set "Frequency" to whatever your sound card supports natively (usually 44100Hz). Also, play around a little with the "Delay" setting (which sets latency). If you're using a pro sound card, you probably won't have a problem setting this to 20 or even 0 milliseconds, which means there's a minimal delay between what happens in the virtual Atari and your sound card's output. However, on "normal" (or

"consumer") sound cards, you may have to raise this value. On my EEE (with a "Realtek HD Audio" sound chip), for example, I can't go any lower than 60 milliseconds. To trim it in, start any of the demo songs at a high value (160 milliseconds or so), exit the Options page to make the change take effect and listen. Then make the value lower and lower (each time exiting the Options page) until it starts sounding crap, then back up one step.

GIVE MY BACK MY MOUSE! Sooner or later, you're going to notice your OS's mouse pointer gets "stolen" by Steem. I e once you've clicked in the Steem window, the only mouse pointer available is going to be the emulated Atari's mouse pointer. To release your mouse pointer back into your OS, press the "Pause/break" key.

The same key will rescue you if you accidentally set maxYMiser to fullscreen mode. In fullscreen mode, the "Pause/break" button will show a button row at the top of your screen; the rightmost button will drop you back to windowed mode (at least in the Win32 version, which is the only one I've used).

MIDI support in Steem works reasonably well. It's certainly worth the effort to hook up a MIDI keyboard to your PC and let Steem route it into the emulated Atari when entering notes in maxYMiser. However, with latency issues (and possibly Steem's method of forwarding the MIDI data to the emulated Atari), I wouldn't recommend using it for realtime playing. It might work better with a real Atari, but I haven't tested it myself.

Are you missing anything in this guide?

If you can think of other (maxYMiser related) things that need explaining in this guide, please get in touch with me at per@brainfish.net - if I have the time, I'll update this guide, and you can feel better for having helped improve it, hopefully making maxYMiser even **more** accessible to other musicians. Giving back to the scene rocks.