TIATracker v1.2

Manual

Andre “Kylearan” Wichmann, 2017

andre.wichmann@gmx.de

Table of Contents

1 Quickstart

To quickly start composing music with TIATracker without reading the whole manual, either load the example song that comes with the program and play around with it, or follow these basic steps:

1. Adapt the keyboard shortcuts in the file "**keymap.cfg**" to your keyboard layout and needs.
2. In the Options tab of TIATracker, select a preset **pitch guide** or create a new one if you roughly know which notes you will mainly want to use in your song. For the latter, you need to select the desired notes on the piano keyboard in the create guide dialog and select the waveforms you want those notes for, then click on the "*Create Guide*" button.
3. Create or import instruments for your song in the "Instruments" and "Percussion" tabs. TIATracker supports two types of instruments:
   - **Melodic instruments** have a base waveform and an ADSR envelope for setting the volume and change the frequency at which a note gets played. Depending on the waveform, different notes are available (shown on the piano keyboard at the bottom of the window). The duration of a note will be defined in the patterns of your song.
   - **Percussion instruments** can set a different waveform, frequency and volume value each frame. The duration of a percussion sound is determined by the number of frames, as each frame gets played exactly once. The pitch of a percussion instrument is defined by its frequency values and cannot be changed in the song.

   In both instrument editor tabs, right-click in the envelope area to insert or delete frames. In the percussion editor, right-click in the waveforms area to select and set a specific waveform, left-click to set the last selected waveform, or use the mouse wheel to scroll through all available waveforms. Press a key on the piano keyboard to hear an instrument.
4. The main track editor tab combines a **pattern editor** and **sequencer**. Right-click either in the column with the pattern names or in the timeline (on the right side) to get a pattern context menu. Use it to create, add, delete and move patterns, set the start patterns or edit goto commands. Right-click a row in a pattern to get a channel and row/note context menu. Use it to insert or delete a row and to edit notes/commands in the row.

   A row can contain a percussion sound, a melodic instrument note, a **hold** command ("|") which continues to play the last note, a **pause**/rest command ("-") which sends a melodic instrument into release and then ends the note, or a **slide** command ("SL *n*") which changes the frequency of the currently played melodic instrument note. A pause or slide command is only valid while a melodic instrument note is playing.
5. The **Info** tab displays a detailed breakdown on how many resources the song will use when played on the VCS. Use the "**export**" menu item to generate .asm or .k65 files for dasm or K65 respecively, to include in your VCS source code.

2 Introduction

TIATracker is a new music routine for the Atari VCS and a PC tracker/sequencer application for composing music for inclusion in VCS games and demos.

This manual targets both musicians and coders. For the impatient, section 1 “Quickstart“ provides a very brief introduction to get you started. Section 3 “VCS Audio“ explains the basic concepts and limitations of audio on the VCS from a user’s point of view. Skip this if you know the VCS already. Section 4 “For the Musician“ explains the concepts behind TIATracker and how to compose music with it, and 5 “For the Coder“ provides information and instructions to the coder how to integrate TIATracker music into their projects.

Several features and optimizations that are already planned have not yet made it into the current version. Contact me for requests, comments and bug reports at andre.wichmann@gmx.de and watch https://bitbucket.org/kylearan/tiatracker/ for new versions. Until then, have fun pushing the limits of VCS music!

3 VCS Audio

The Atari VCS has 2 audio channels. For each channel there are three parameters that can be set to produce sound:

- **Waveform** (“Distortion”). There are 11 different waveforms: Silence, white noise, and 9 others. Two of them have the same timbre, but different frequency ranges (see below).
- **Frequency**. There are only 32 frequencies that each waveform can be played at. These frequencies unfortunately do not match traditional notes. Instead, many notes are missing completely, and most of the notes that are available are off-tune to varying degrees. Even worse, the frequency ranges differ across waveforms. Notes that are available for some waveforms are missing for others. Frequencies will differ slightly depending on whether the VCS is a PAL or NTSC machine.
- **Volume**. There are 16 different volume levels available.

The VCS has no filters, in-built ADSR envelopes are other features; only the three parameters above define the sound output for each channel.

For technical details and a list of all waveforms and frequencies see Eckhard Stolberg’s “Atari 2600 VCS Sound Frequency and Waveform Guide“ at http://home.arcor.de/estolberg/texts/freqform.txt. To hear all available waveforms and frequencies, experiment in the melodic or percussion instrument tabs in TIATracker.

For more detailed information about the VCS audio capabilities and music culture, you can watch a seminar talk from the Revision 2016 demoparty called “News from an alien and twisted musical desert” at https://www.youtube.com/watch?v=PVujzQySZls which also includes a playback of all possible notes.

4 For the Musician

TIATracker, the tracker/sequencer application, can be used to compose and hear VCS music on the PC. It uses the sound emulation routines from the VCS emulator Stella, which are good but not perfect, and you can export a song to source code for inclusion into a game or a demo. Some of the general concepts are very similar to concepts from tracker/sequencer software for other platforms, but some of them are very specific to the VCS.

The file “keymap.cfg”, which is in the same folder as the TIATracker program itself contains all keyboard shortcut definitions. You will want to adapt this to your local keyboard layout and personal habits.

4.1 General Concepts

4.1.1 Song anatomy

A song in TIATracker is defined by two ***sequences*** of patterns, one sequence per channel. A ***pattern*** is a list of rows, each containing a note, a pause or a command, that will be played sequentially and that can be of variable length. Each sequence can play the same pattern multiple times, and playback of both channels is independent from each other, i.e. a pattern playing on one channel does not need to have the same length as the pattern playing at the same time on the other channel.

The player routine updates sound output once per TV ***frame***, i.e. 50 times per second on PAL systems and 60 times per second for NTSC. A ***tempo*** value defines how may frames a note is played before the next row in the pattern is consulted for what to play next. For example, a tempo of five means that every five frames a new note or command gets processed, which means 10 notes per second on a PAL system or 12 notes on an NTSC system. Even and odd rows can have different tempo values so that every second note can last shorter or longer than the others, potentially giving the song a “funky” style.

A song can either have one even and one odd tempo for the whole duration, which is called global tempo, or it can define even and odd tempo values individually per pattern. In the latter case, the tempo values of the pattern currently played in the first channel define the current tempo of the song.

4.1.2 Instruments

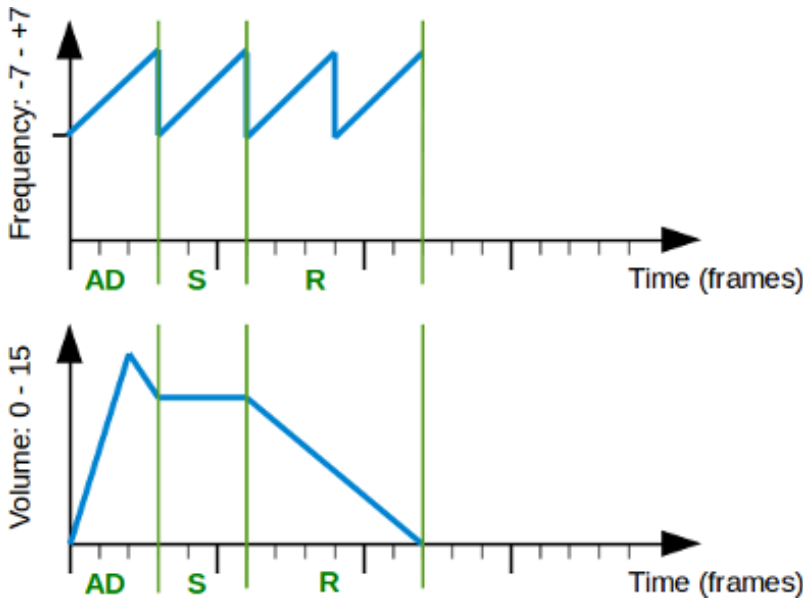
There are two fundamentally different types of instruments in TIATracker: Percussion and melodic instruments. Both types get updated by the player once per frame.

Percussion instruments can set a new value for waveform, frequency and volume each frame. When a percussion is encountered in a pattern, each frame of the respective percussion instrument gets played exactly once, or until a new note in the pattern interrupts it and cuts it off. A percussion instrument cannot be played at a different pitch, since each frame exactly defines which frequency to use.

Melodic instruments have a base waveform value that stays the same for all its frames, but they can be played at different frequencies as defined in the patterns to form melodies. To shape each note, a

melodic instrument can set a different volume value each frame and modify the note frequency by a value between -7 and +7 for arpeggio or tremolo effects.

A melodic instrument's lifetime is divided into three phases: Attack/Decay, Sustain, and Release. Together they define frequency and volume ***ADSR envelopes*** for this instrument, as depicted in the following graphics (the small ticks mark frames, the larger ticks the song tempo, in this example case 5).



When a note starts, all frames of the Attack/Decay phase are played. Afterwards, the Sustain phase frames are played in a loop. When a pause note is encountered in the pattern, the Release phase frames are played once and the note ends in silence. If a new note is encountered instead of a pause, the new note simply cuts off the current instrument without sending it into Release phase.

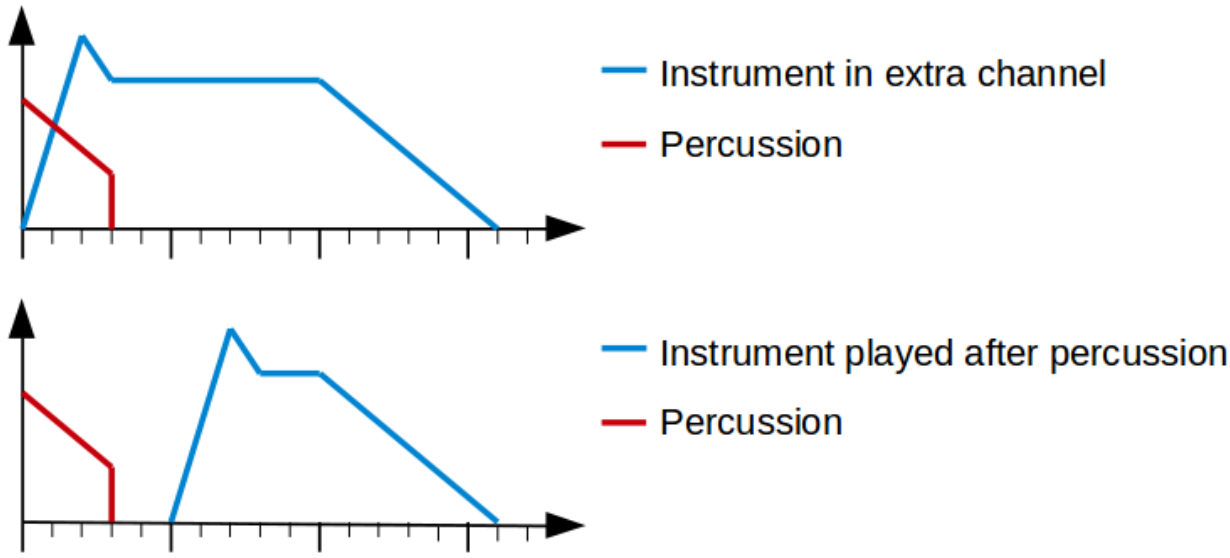
Since melodic instruments can be used to play different notes, they have a base waveform that cannot be changed each frame. This is because available frequencies vary across different waveforms, and thus playing a specific note using one waveform makes no sense for another waveform where that note might not be available. For the same reason, percussion instruments cannot be played at different frequencies, as they might set different waveforms each frame.

Frequency values for playing a melodic instrument notes range between 0 and 31, and that frequency value can be changed in the envelope each frame by +/- 7. Note that this might cause an underflow or overflow for some notes. In that case, the values simply wrap from low to high frequencies or vice versa, which might or might not be intended by the musician.

4.1.3 Overlay Percussion

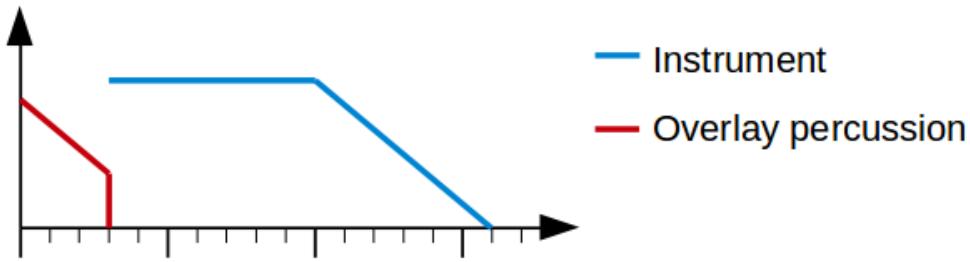
In normal music, you typcially have percussion instruments and melodic instruments playing at the same time, for example by arranging them in different channels. With only two channels, this is more difficult to do on the VCS and the composer will have to play percussion and melodic

instruments in the same channel in quick succession to create the audio illusion that they are played simultaneously. This is depicted in the following graphics.



The top picture shows the ideal situation, where the melodic and percussion instruments are played simultaneously in different channels. The bottom picture shows what happens if the composer plays them one after the other in the same channel. In this example with a song tempo of 5, you have a gap of two ticks after the percussion has finished before the next note is played.

TIATracker has the option to mark a percussion instrument as an ***overlay*** instrument. When an overlay percussion instrument has finished playing, the next note will be fetched out of order regardless of tempo. If it is a note for a melodic instrument, it will be played immediately, skipping its Attack/Decay phase and starting it right away in the Sustain phase. This is depicted in the following graphics.



Since percussion instruments are often louder, this can help to create the illusion that both instruments have been played at the same time but the percussion instrument has drowned out the first couple of frames from the melodic instrument.

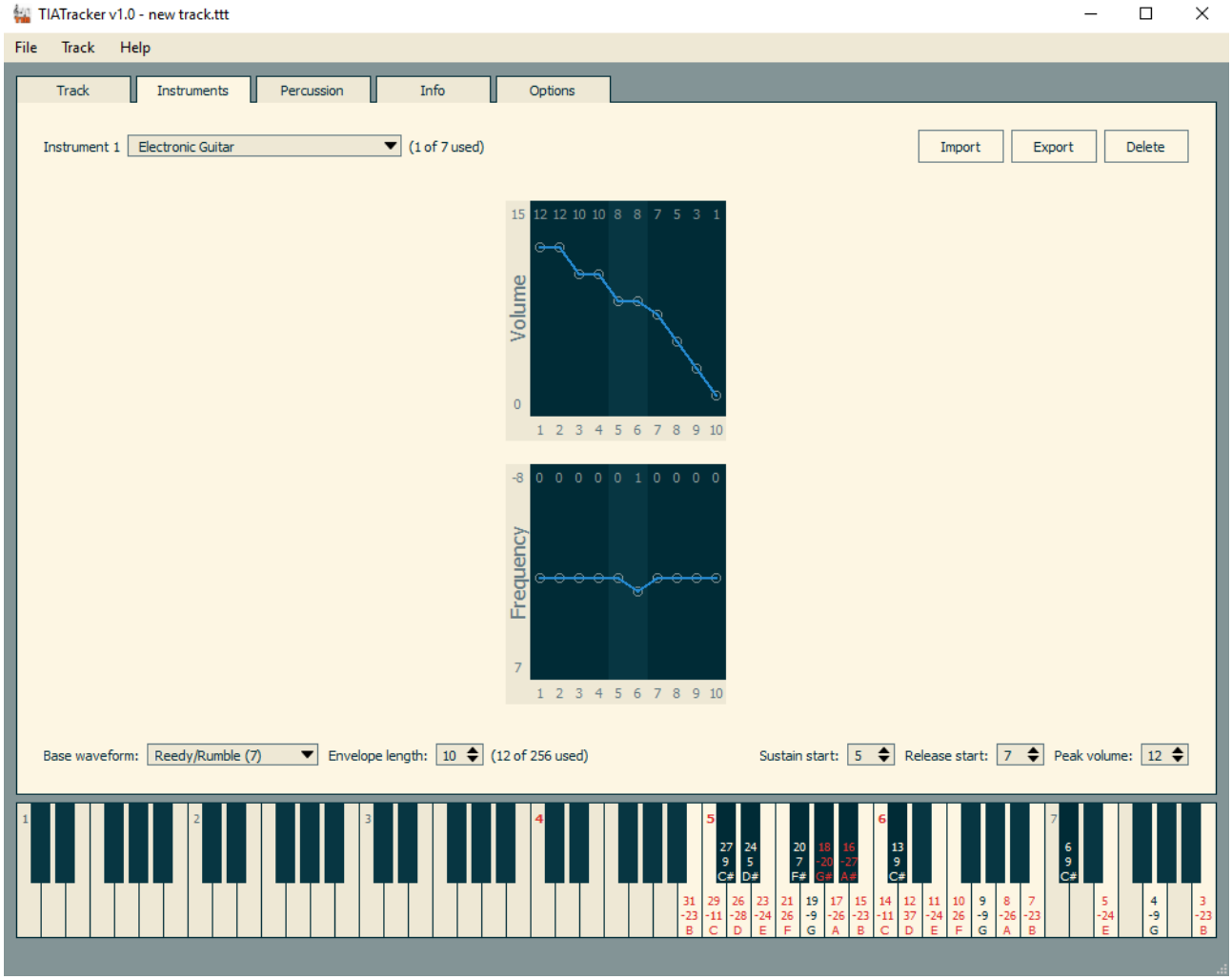
If the row following an overlay percussion instrument is another percussion instrument or a hold command, it will get processed normally.

Note however that you cannot specify this “overlay” behavior individually per note, only per percussion instrument. Thus, an overlay percussion instrument will always play the next melodic note immediately once it has finished.

4.2 Creating Instruments

TIATracker supports up to 7 melodic and up to 15 percussion instruments.

4.2.1 Melodic Instruments



In the melodic instruments editor tab, use the drop-down list in the top left corner to name your melodic instrument and to switch between the available instruments. The ***import*** and ***export*** buttons save and load individual instruments independent from the song, and ***delete*** clears the envelope and name of the currently selected instrument.

You can select the base waveform using the drop-down list in the lower left. Apart from the physically available waveforms from the VCS hardware, there is also a special convenience waveform called "***Pure Combined (4+12)***". This combines the two so-called "pure" waveforms (Pure High (4) and Pure Low (12)) which have the same timbre but different note ranges, into one "virtual" waveform with 64 instead of 32 frequencies. This allows the musician to compose a melody without having to switch between the two real "pure" waveforms depending on which note is needed. When the song gets exported and played on the VCS, TIATracker will automatically generate two different instruments (which share the same envelope) using the two "pure" waveforms and assigns the notes in the song accordingly. For this reason, the Pure Combined waveform counts as two instruments.

The structure of the ADSR envelope can be defined using the ***envelope length***, ***sustain start*** and ***release start*** controls. Note that a melodic instrument does not need to have an attack/decay phase, but the sustain and release phases both need to contain at least one frame. In the middle of the window, the volume and frequency envelopes are graphically displayed, with the sustain phase highlighted in the background. Left-click in a frame to set a new value, and right-click for a context menu to insert and delete individual frames.

The ***peak volume*** control shows the highest volume value of the whole envelope. Use it to shift all volumes up and down, to make the instrument as a whole louder or quieter.

Finally, click and hold on a key on the piano keyboard (see section 4.3) to hear how the melodic instrument sounds.

4.2.2 Percussion Instruments

The percussion instruments editor tab is very similar to that for melodic instruments. In addition to setting the values for volume and frequency, you can also set the waveform individually per frame. Right-click in the waveform area to select and set the waveform for a frame. A left click sets it to the last selected waveform, so you can use it to quickly set several frames to a specific waveform. In addition, you can use the mouse wheel to cycle through all available waveforms.

Using the "***Overlay***" checkbox, you can specify that the percussion instrument should use the overlay mechanic during replay, i.e. that it will start to play the next melodic note in sustain immediately when its last frame has been played (see section 4.1.3).

Since percussion instruments cannot be played at different pitch, the piano keyboard does not display any pitch guide. Use any key to on the piano keyboard to hear how the currently selected percussion instrument sounds.

4.3 The Piano Keyboard

The piano keyboard at the bottom of the window displays which notes are available for the selected waveform and how off-key they sound. Each available note has its frequency index value (0-31) listed as well as the percentage how far it deviates from the "real" frequency of the note. If this percentage is greater than a threshold value (which can be adjusted in the Options tab), the note is displayed in red.

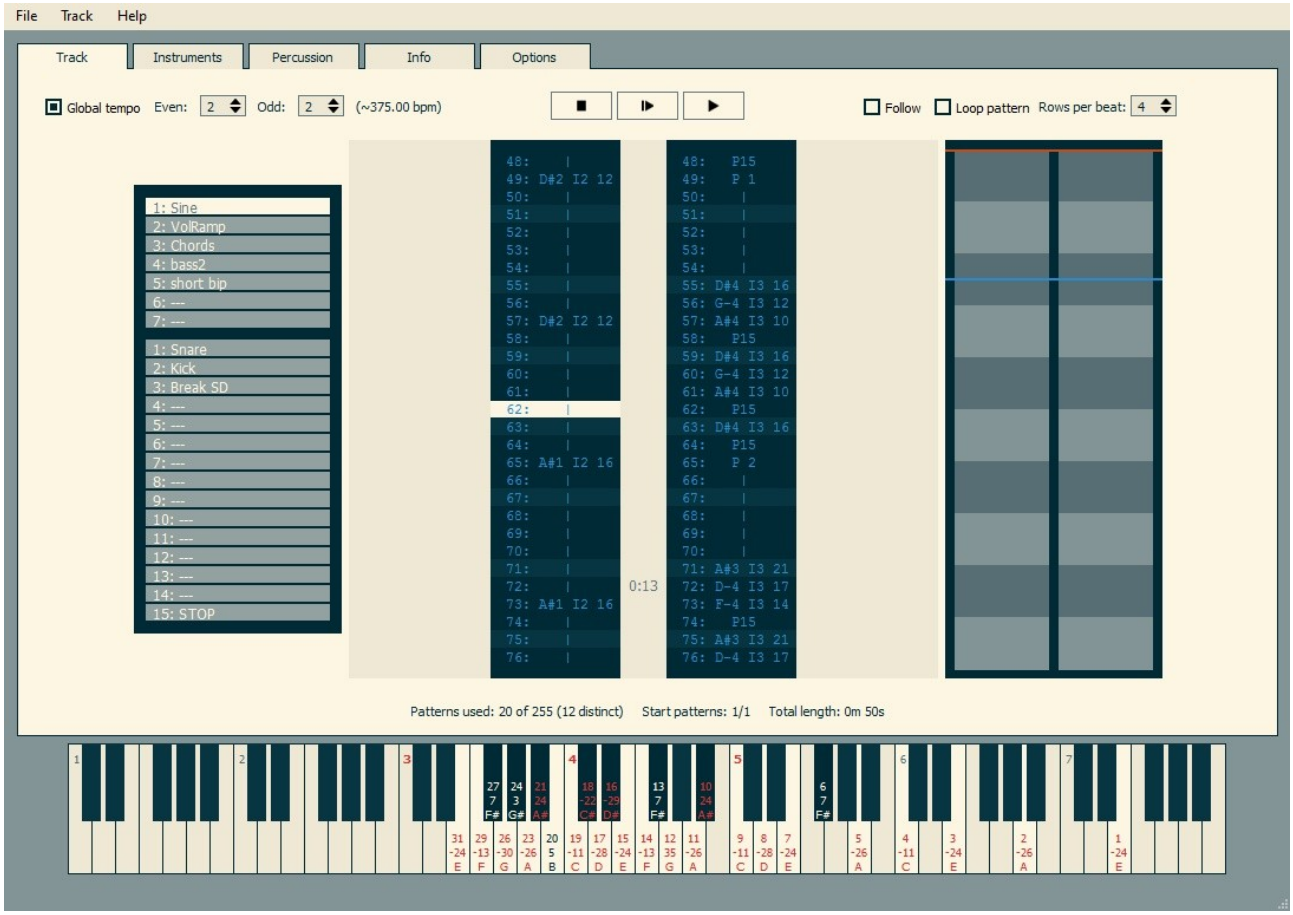
The exact details which notes are available for each waveform and how much out of tune they are depends on the currently selected pitch guide (see section 4.5.3). Note that this is only a visual aid for the musician. The song itself stores raw frequency values instead of notes, so selecting a different pitch guide will only change the *labels* on the piano keyboard and the rows in the track, not how the song will sound.

Apart from clicking with the mouse, you can also use keyboard shortcuts (see section 6.1.3) to access three consecutive octaves on the piano keyboard. Small numbers printed in red on the top of each C note key denote which three octaves can currently be accessed via keyboard shortcuts, and you can shift these three octaves also via a set of keyboard shortcuts. If you select a melodic

instrument in the selector, the octaves automatically get set so that the lowest note of the instrument is in the lowest octave accessible via keyboard.

4.4 Composing Music

The user interface in the “Track” tab for composing mainly consists of three areas, from left to right: The instrument selector, the pattern editor, and the timeline. Above those are the player and song option controls.



4.4.1 Instrument selector

Use the ***instrument selector*** to select the melodic or percussion instrument you want to use for editing. Depending on the selected instrument, the piano keyboard will show which notes are available based on the waveform of the melodic instrument and the pitch guide selected in the options tab (see section 4.5.3). In case of percussion instruments, no pitch guide is shown and any key can be used to enter the percussion instrument into a pattern.

4.4.2 Timeline

The ***timeline*** on the right side of the window depicts the pattern sequences for both channels, with alternating colors signifying when a new pattern starts. Hover the mouse over a pattern to see its name in a tooltip, and left-click on a pattern to jump to that position in the pattern editor. A right-click opens a pattern context menu which can also be accessed by right-clicking in the pattern area of the pattern editor, as described in the next section.

The blue line marks the current position of the editor cursor. The orange line shows the current position of the song player.

4.4.3 Pattern Editor

The ***pattern editor*** area in the center of the “Track” tab combines a way for entering notes into patterns and a sequencer for arranging the patterns into a song. For each channel, the patterns are displayed in the order they should be played. The rows of each pattern are displayed in the inner columns, and the outer columns contain the pattern names and potential goto commands. In the center between the channels timestamps are displayed for the rows, which depend on the specified song tempo and the TV standard set in the options tab (see section 4.5.2).

Use the mouse wheel or the corresponding keyboard shortcuts to move the editing cursor (the highlighted row in the middle) up or down, or to switch editing between the two channels. A right-click in the area with the pattern names opens a pattern-specific context menu where you can modify the pattern sequence. A right-click on a specific row opens a context menu for modifying rows.

Editing a pattern

A row in a pattern can either be a percussion instrument, a note for a melodic instrument, a “hold” command, a “pause” command, or a “slide” command. Use the instrument selector to change the currently active instrument.

- To enter a note for a ***melodic instrument***, you can use the piano keyboard at the bottom of the screen. Which notes are available for a given melodic instrument depends on the selected pitch guide (see section 4.5.3). Alternatively, three octaves are mapped to keyboard shortcuts (see section 6.1.3). The currently selected octaves are displayed in red on the piano keyboard, and they can also be changed by using keyboard shortcuts.

  If you want to manually enter a note with a specific frequency value not available on the piano keyboard, use the “***Change frequency...***” item in the context menu.

  If a “???” is displayed, it means that the frequency of that note is outside the range of C1 to B7. It will be played normally, however.

- To enter a ***percussion instrument***, simply press any key on the piano keyboard or use any corresponding keyboard shortcut.

- A “***Hold***” command is displayed as a “|” symbol in the pattern and can be entered with the corresponding keyboard shortcut. When playing the song, it means that the currently played note should continue playing. In case of a percussion instrument, this means that it simply continues to play frame after frame and that it falls silent afterwards. In case of a melodic instrument, it means that the sustain phase is looped as long as more “Hold” commands are encountered.

- A “***Pause***” command is shown as a “=” symbol in the pattern and can be entered with the corresponding keyboard shortcut. When a melodic instrument is currently played, it will cause it to play its release phase and fall silent afterwards.

It is not valid to place a “Pause” after a percussion instrument or a “Hold” command following a percussion instrument. This will throw an error when the player routine of the tracker arrives at such a row, and will cause undefined behavior in the VCS player routine. (This is due to size optimizations of the routine.)

- A “***Slide***” command is only valid when a melodic instrument is currently playing. It modifies the frequency of the currently played note by a value between -7 and +7 while keeping it in sustain phase. Use the “***Set to slide...***” item in the context menu to enter this command.

  It is not valid to place a “Slide” after a percussion instrument or a “Hold” command following a percussion instrument. This will throw an error when the player routine arrives at that row, and will cause undefined behavior in the VCS player routine.

Note that editing a row in a pattern means that all occurrences of that pattern in both sequences will be modified. You can also insert or delete rows in a pattern by using the context menu or the corresponding keyboard shortcuts.

As a pure visual aid, the pattern editor highlights every n-th row of a pattern so that something like regular beats can be visualized. The “***Rows per beat***” control specifies the distance between two highlighted rows, i.e. how many rows one beat comprises. This value is also used for computing the “beats per minute” value displayed near the tempo controls on the top left.

Sequencing

The pattern context menu, accessible via right-click either in the area with the pattern names or in the timelime, is used to create, delete, rename and order patterns in a channel sequence as well as set start patterns and goto commands. For each channel there is one pattern marked as the ***start pattern***, which is where replay begins. From there, it will play patterns in the order as they are displayed in the sequence and the timeline.

A ***goto*** command at the end of a pattern forces the player to jump to the specified pattern. That way, loops can be created. However, the target of a goto command has to be smaller than 128, and for a goto command in channel 1, the number of patterns in channel 0 are added to the target number as well. This is due to size optimizaion in the VCS player routine, and on export, the tracker will display an error if an invalid goto is encountered.

The same pattern can be played several times during a sequence. The “***insert pattern***” menu item lets you either insert an existing pattern, or create and insert a new, empty pattern. If you ***create*** a ***new*** pattern, a dialog appears where you can specify the name and the length of the new pattern. The “***Align***” button automatically sets the length to a value so that the end of the new pattern aligns itself with the end of the pattern played at the same time in the other channel. This makes it easier to keep patterns in both channels in sync.

You can also ***duplicate*** a pattern. This creates a copy of the current pattern which you can then edit without changing any occurrences of the original pattern in the sequences.

“***Remove pattern***” removes the current instance of a pattern from the sequence; the pattern itself can still be used elsewhere. If you remove the last instance of a pattern from the sequences, TIATracker

will ask if it should delete the pattern definition itself as well, in which case you will no longer be able to insert it again into the song. Note that when you later export your song to assembly language, unused patterns will not get exported and will not use up any resources, so you don't need to delete unused patterns if you think you might still need that pattern later.

4.4.4 Song Player

At the top of the track tab are the song player controls and some replay options. You can start replay at the current editing position, start replay from the start of the song (as defined by the start patterns in the sequencer) or stop replay using the icons in the middle. In addition, you can start replay at the start of the current pattern via the "Track" menu. When an invalid command is encountered during replay, for example a pause following a percussion instrument, replay will stop before the offending row and display an error message.

When the "***follow***" checkbox is ticked, the editing cursor of the currently selected channel in the pattern editor follows the current replay position so you will have the notes currently played always displayed in the center. "***Loop pattern***" ignores the defined sequence and causes the same pattern in the currently selected channel to be repeated. Use "***Mute channel***" in the context menu of a channel to toggle sound for that channel on and off.

Replay speed is defined by the "***Even tempo***" and "**Odd tempo**" controls in the top left. These tempo values define how many frames each even- and odd-numbered row in a pattern should last. Bigger values cause the song to be played slower, smaller values faster. If "**Global tempo**" is checked, the values will be used for the whole song. Otherwise, each pattern can have its individual tempo values, and the pattern currently played on the first (left) channel determines the current tempo of the song.

4.5 Info and Options Tabs

The info tab displays how many resources the current song will need on the VCS, and the options tab lets you select some global options and optionally create and use a pitch guide.

4.5.1 Info Tab

In most cases, music for the VCS needs to be very small. The info tab displays a detailed breakdown of how much resources the current song will use on the VCS, helping you to find out where you can save some more bytes to make a song smaller, if needed.

Component	Used?	ROM	RAM
Core:	X	166	9+4 tmp
Goto Pattern:	X	8	
Slide:	-	~~9~~	
Overlay Percussion:	-	~~40~~	
Funktempo:	-	~~7~~	
Starts with Hold:	-	~~2~~	
Instruments:	2	28	
Percussion:	4	134	
Patterns:	12	1560	
Channel Sequences:		22	
Total:		1918	9+4 tmp

The total size of a song includes the replay routine core which cannot be changed, optional routines depending on which features are used in the song, instrument data, pattern data, and sequence data. Instrument, pattern and sequence data can be reduced directly by making instrument definitions or the song itself shorter. Optional routines work a bit differently. As soon as a certain feature is used anywhere in the song, a fixed overhead for using this feature is added to the overall size. If a feature appears struck through in the breakdown, it means it is not used in this song.

The following features can increase the size of the replay routine:

- **Goto Pattern:** If you use the goto pattern feature in the sequencer. Note that if you do not use goto, behavior of the VCS replay routine is undefined once the song has finished playing and the programmer has to make sure nothing bad happens.
- **Slide:** If you modify the note of a melodic instrument anywhere in the song with the slide feature.
- **Overlay Percussion:** If any of the percussion instruments used in the song has the “overlay” feature checked.
- **Funktempo:** If the even and odd tempo values are different. Size requirements are differently depending on the use of *No global speed*.
- **No global speed:** If each pattern has its individual tempo values or if only one set of values is used for the whole song. Size requirements are differently depending on the use of *Funktempo*.
- **Starts with Hold:** If the very first note of the song for both channels (in the start patterns) is not a melodic or percussion instrument.

Note that the RAM usage currently is the same for all features. In future versions, different player variants optimized for minimal RAM usage at the cost of size will be introduced.

4.5.2 Options Tab

In the options tab, you can set the ***TV standard*** that should be used for song replay. PAL means that the music gets updated 50 times per second, NTSC 60 times per second. This affects both replay speed, making NTSC music play slightly faster than PAL music, as well as the tuning. Waveforms use slightly different frequencies depending on whether they are played on PAL or NTSC systems.

In addition, you can enter the song author's name, the name of the song and a comment. Author and song name will get exported to the assembly source as a comment, but will not affect the resulting VCS binary in any way.

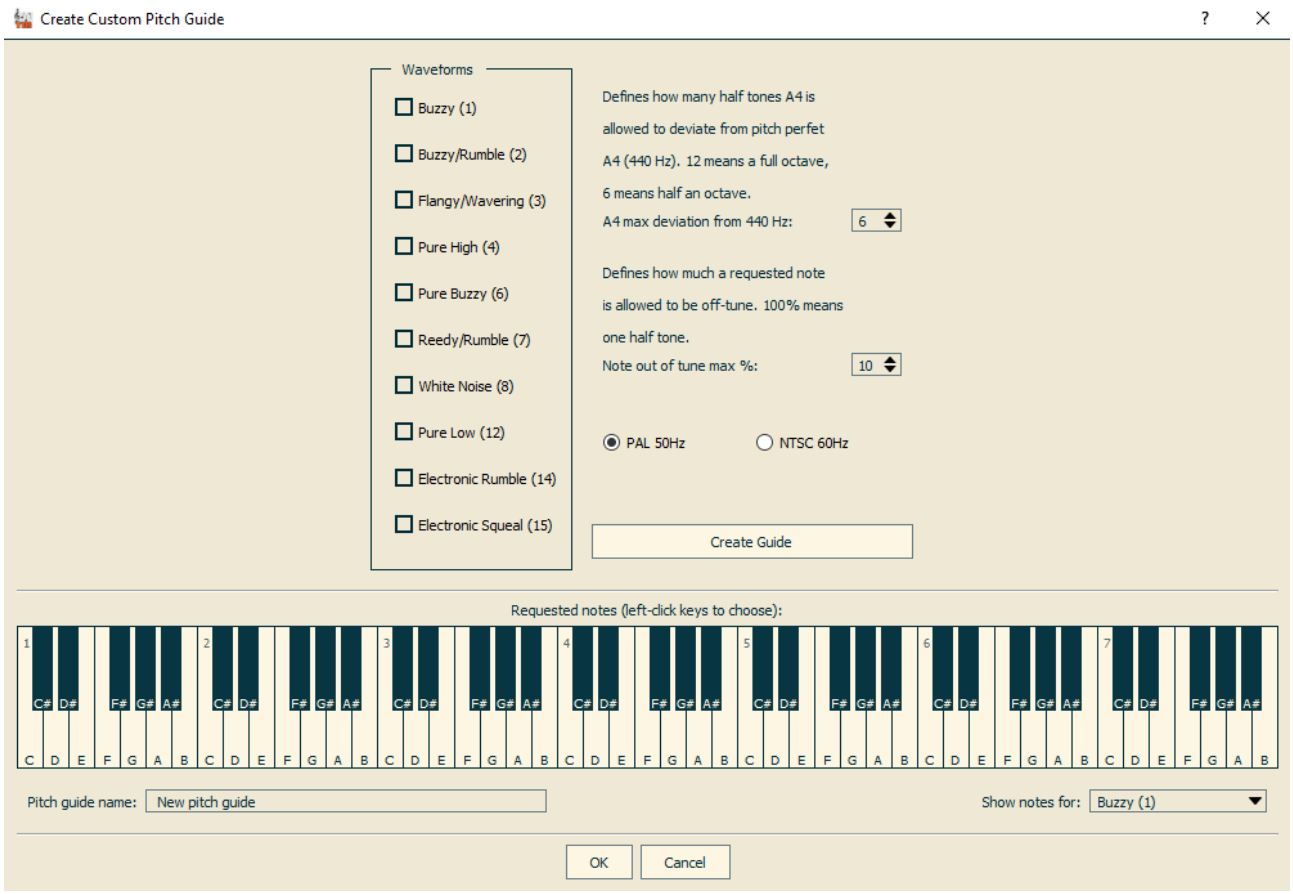
Finally, you can select the pitch guide to use from a list of currently available pitch guides (several preset and optional user-created ones), export the currently selected pitch guide to disk or import a previously stored one. The "***Off-tune % warning at***" control lets you set the percentage value for how off-tune a note may be before it is displayed in red on the piano keyboard.

The preset pitch guides simply optimize the number of in-tune notes for a given waveform without preferring certain notes or taking into account if the combinations of notes will be usable to a musician or not. With TIATracker also come two example pitch guides ready for import that optimize the number of notes in C major between C-3 and C-6 for the Pure Low and Pure High waveforms, one for PAL and one for NTSC. Use the pitch guide creator to create different scales for other waveforms.

4.5.3 Pitch Guides

Pitch Guides are a concept to make the lack of available in-tune notes (see section 3) easier to deal with. It exploits the fact that unless you have pitch-perfect hearing, the human ear does not care about absolute frequencies when listening to music. Instead, only the relative difference between frequencies will cause a chord to sound harmonic or out of tune.

For normal instruments or other sound chips, the note A4 is usually tuned to 440 Hz so that different instruments can play music together. This tuning creates the problem that the frequencies available on the VCS will make most of the notes to be horribly out of tune. However, if we define the frequency to use for A4 differently, more notes with better tuning can be made available. Changing this base frequency, different sets of notes depending on your needs can be created. This is what the pitch guide creator is used for, accessible via the Create Guide button in the Options tab.



Using the piano keyboard in the pitch guide creator window, you can select the notes which you would like to use in your song. On the left side, select the VCS waveforms you would like to use these notes for. Three additional parameters can be defined for creating a pitch guide:

- **TV Standard:** Frequencies for PAL and NTSC machines are slightly different, so select the TV standard of your target architecture here.
- **A4 max deviation from 440 Hz:** This value specifies how many half-tones A4 may deviate from the “real” A4 at 440 Hz. For example, a value of 12 means that the resulting A4 may be up to one octave lower or higher than standard A4, since one octave consists of 12 half-tones. Similarly, a value of 6 means A4 could deviate only half an octave at most.
- **Note out of tune max %:** This percentage value defines how off-tune each note you requested may be allowed to be. 0% means only notes with exactly the correct frequency will be allowed, while 100% means a note might deviate up to one half-tone.

The “***Create Guide***” button computes the optimal frequency for A4 so that for the waveforms you selected, as many in-tune notes (as defined by the max deviation threshold) as possible become available. Afterwards, the result is displayed on the piano keyboard. Select the waveform for which you want to see the available notes in the drop-down list in the lower right to check if you are satisfied with the results. If not, you can change your selection of waveforms, notes and parameters and re-create the guide.

Once you are finished, you can name your pitch guide and after clicking the “***OK***” button, it will be added to the list of available pitch guides and automatically becomes selected.

If you save your song, the currently used pitch guide will be saved along with it. You can also use the "***Import...***" and "***Export...***" buttons in the options tab to manage pitch guides separately.

Note that pitch guides are a purely visual convenience for the musician and will not affect in any way how a song will sound. Each note in a song is defined by its VCS frequency value. A pitch guide will only change the labeling, i.e. which note name will be displayed for a given frequency in the patterns or on the piano keyboard.

5 For the Coder

There are three different formats TIATracker can export a song to: *.asm* for use with the ***dasm*** assembler, *.k65* for use with the ***K65*** compiler framework, and generic *.csv* to import into other programs like a spreadsheet.

5.1 dasm

TIATracker can export the current song to assembly in dasm format for use on the VCS in two ways, accessible via the File menu. "***Export complete player to dasm...***" will write five different files:

- **<filename>_variables.asm:** This contains compiler flags defining tempo, TV standard and configuration of the player routine depending on which features are used in the song. In addition, it defines the RAM variables used in the player. *Permanent* variables need to keep their state between calls to the player routine, while *temporary* variables will be overwritten by the routine but can be used for other purposes in between calls.

  Include this file in your source where you define your RAM variables.

- **<filename>_init.asm**: This will initialize the player variables so that the song will start at the beginning. It assumes that all RAM variables have been initialized to 0 during startup. If that is not the case, or if you want to restart the song at some point during your program, you have to set some player variables to 0 yourself first (see comments in *<filename>_init.asm*).

  Include this file anywhere in your source before your first call to the player routine.

- **<filename>_player.asm:** This is the player routine. It has to be executed once per frame to replay the song, for example during vertical blank or overscan. The routine is static and does not depend on the song to be played, except for using the compiler flags defined in *<filename>_variables.asm* to strip out unneeded parts of the code during compilation.

  Include this file in your source where you want to execute it once per frame. Note that it is *not* a subroutine. If you want to use JSR to call the player routine, you will have to insert an RTS instruction manually after the include statement.

- **<filename>_trackdata.asm:** This file contains all song-dependent data structures: Instruments, patterns, and the channel sequences.

  Include this file anywhere in your source.

- **<filename>_player_framework.asm:** This file is a small, yet complete VCS program that initializes the machine, sets up a simple display kernel and calls the player routine once per frame. This allows you to quickly test your song on a real VCS or in an emulator. It is completely independent from the song to be played.

  Compile this file with dasm along with the appropriate vcs.h include to get a binary that will play your song. This can be done with the following command:
  `dasm <filename>_player_framework.asm -f3 -o<filename>.bin`

The file *<filename>_player_framework.asm* also serves as an example of how to include and use all necessary files to add a song to a VCS program.

The menu item "***Export track data to dasm...***" only exports the files that directly depend on the current song: *<filename>_variables.asm*, *<filename>_init.asm* and *<filename>_trackdata.asm*. This can be used if you have a VCS project that already has the player routine and you only want to update the song itself.

5.2 K65

Similar to exporting to dasm format, TIATracker can either export a complete player that compiles in K65 and plays a song, or only files that directly depend on the current song.

"***Export complete player to k65...***" will write four different files:

- **<filename>_trackdata.k65:** This file contains all song-dependent data structures: Instruments, patterns, and the channel sequences. In addition, it also contains the inlined function "*tt_init*" that initialize the player variables so that the song will start at the beginning. It assumes that all RAM variables have been initialized to 0 during startup. If that is not the case, or if you want to restart the song at some point during your program, you have to set some player variables to 0 yourself first (see comments in the source file). Finally, it contains compiler flags defining tempo, TV standard and configuration of the player routine depending on which features are used in the song.
  Include this file before the player routine, as the player will reference the data structures defined in this file.

- **<filename>_player.k65:** This defines the inlined player routine, called "*tt_player*". This routine has to be called once per frame to replay the song, for example during vertical blank or overscan. The routine is static and does not depend on the song to be played, except for using the compiler flags defined in *<filename>_trackdata.k65* to strip out unneeded parts of the code during compilation.
  It also contains the RAM variables used in the player. *Permanent* variables need to keep their state between calls to the player routine, while *temporary* variables will be overwritten by the routine but can be used for other purposes in between calls.
  Caution: The variable definitions set the first variable of the player routine to be at memory location 0x80. Adapt this to your needs if necessary.

- **<filename>_player_main.k65:** This file is a small, yet complete VCS program that initializes the machine, sets up a simple display kernel and calls the player routine once per

frame. This allows you to quickly test your song on a real VCS or in an emulator. It is completely independent from the song to be played except for the TV standard (PAL or NTSC) defined as a compiler flag.

- **<filename>_player_framework.lst:** This file defines which .k65 files to compile in which order for the small test program. Compile this with K65 to get a VCS binary. It is completely independent from the song to be played. For a manual on how to use k65 to generate the binary, refer to http://devkk.net/wiki/index.php?title=K65.

The file *<filename>_player_main.k65* also serves as an example of how to use all necessary files to add a song to a VCS program.

The menu item "***Export track data to k65...***" only exports the file that directly depend on the current song: *<filename>_trackdata.k65*. This can be used if you have a VCS project that already has the player routine and you only want to update the song itself.

5.3 CSV

"Export track to csv..." creates a comma-separated values file that contains a high-level description of the current song. Each row corresponds to a row in TIATracker and contains data about the instruments or commands for each channel and data about the song position (sequence number, pattern number and name). This could be imported into a spreadsheet application and annotated with additional rows for synchronizing demo effects to music events, for example.

Caution: In the current version, the values correspond to the values displayed in the TIATracker PC application, *not* to the values used in the exported VCS binary. That is because during export, only instruments and patterns that are actually used in a song get exported, and thus numbering will be different.

5.4 Adapting and optimizing the player routine

All source files are richly documented so you can easily adapt them to your needs. In particular, *<filename>_trackdata* contains a detailed specification of all data structures used for song replay.

In their current form, the player routine and the data structures are optimized for minimal ROM usage at the cost of RAM and speed. Less RAM variables could be used by reconstructing some values each frame instead of keeping them in permanent variables, and by limiting the number and/or the length of patterns and sequences. Less instruction cycles could be used by inlining some subroutines, by making sure loops and data structures will not cross page boundaries and by keeping more state information in RAM. (Note that some data structures are addressed with a negative offset in the code, so to keep the table accesses from crossing a boundary the tables would need to reside at a positive offset from the start of a memory page.)

If you have a specific need for optimization, feel free to contact me and I might be able to add that feature to TIATracker.

6 Appendix

6.1.1 Licences and Third-party Code

The tracker application is published under GPLv2 and can be found at https://bitbucket.org/kylearan/tiatracker. It uses SDL2 (https://www.libsdl.org) and Qt5 (http://www.qt.io). It also uses the TIA sound emulation from the Stella emulator (http://stella.sourceforge.net).

The player routine for the VCS is published under the Apache license v2 (http://www.apache.org/licenses/LICENSE-2.0).

6.1.2 Acknowledgements

Big thanks to Nikola “juice” Fox for great feedback and lots of advice on TIATracker and how trackers work in general, and for beta-testing.

Also thanks to Thomas Jentzsch for spending a lot of time with a very early alpha version of TIATracker and coming back with good questions and suggestions.

Dok Sae was the first one to create a full song with TIATracker, used in the VCS demo “odd” by Flush, and kindly allowed me to use it as an example song. Many thanks!

Sagamusix was kind enough to answer my newbie questions about tracker keyboard shortcuts and German vs. international musical scales.

6.1.3 Keyboard Shortcuts

The file “keymap.cfg” in the TIATracker folder contains the definitions for all keyboard shortcuts. In its default configuration, it’s configured for a German keyboard layout and loosely follows some conventions from OpenMPT. Use a text editor to change the key bindings, but be careful to preserve the general structure.