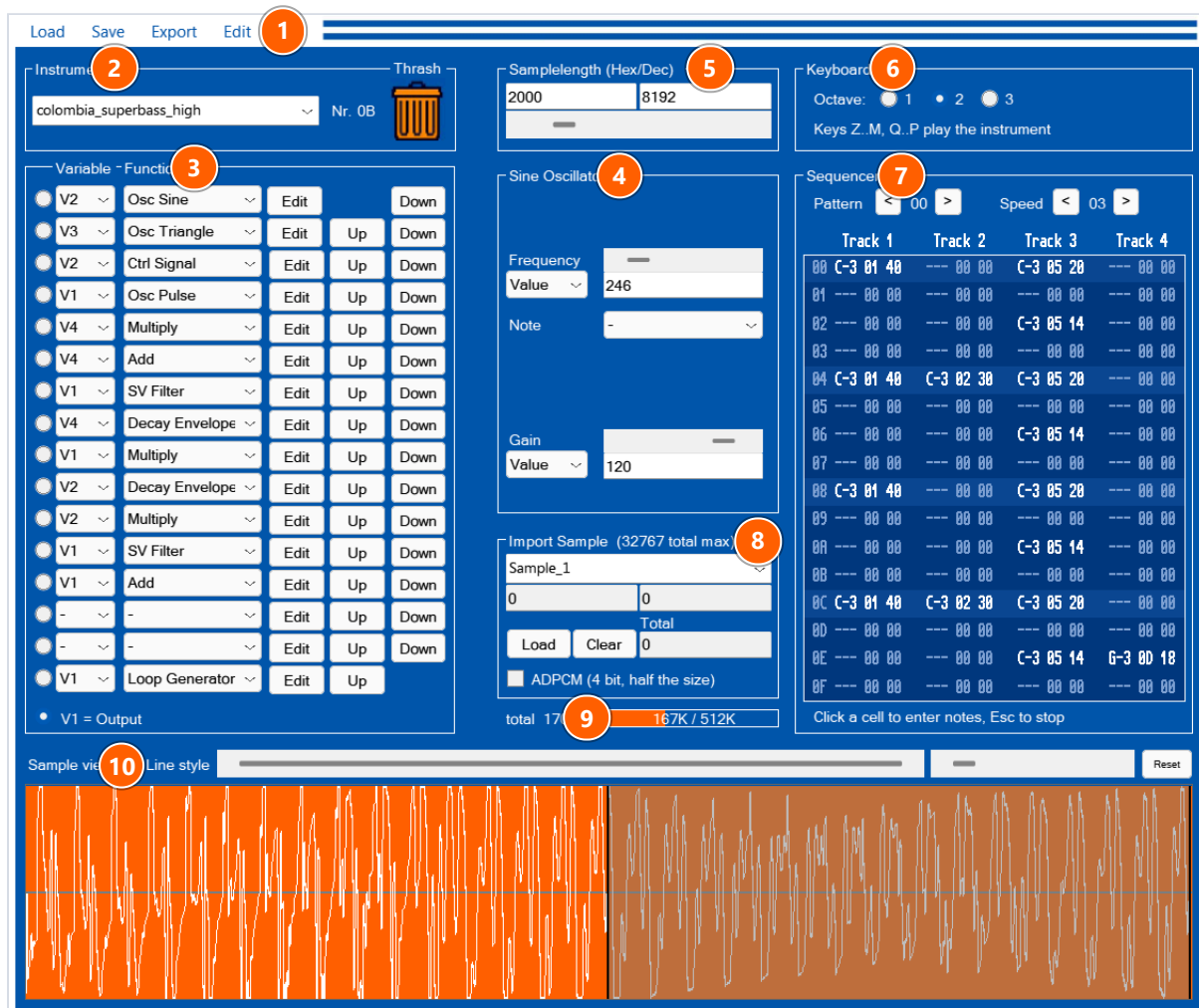


AmigaKlang - Quick Guide

Modular sample synthesizer for Amiga intros · ImGui editor V1.5 · October 2026

AmigaKlang builds instruments from small DSP building blocks. The editor renders them on the PC exactly like the 68000 code that later runs on the Amiga, so a song needs only the tiny patch instead of the samples: the samples are generated at the start of the intro. This guide shows the main window, how to build and play instruments, the sequencer, and how to turn a song into a standalone Amiga executable.

The main window



- 1 Menu** - load, save, export, undo, design.
- 2 Instrument** - select one of 31 instruments (also with the mouse wheel); *Nr.* shows its number in hex; the bin clears it.
- 3 Slots** - 16 processing steps per instrument (variable + function).
- 4 Parameters** of the slot opened with *Edit*.
- 5 Sample length** in hex and decimal, plus slider.
- 6 Keyboard** - octave for playing with the computer keyboard.
- 7 Sequencer** - 16 patterns, 4 tracks, 16 rows (for the Sample Mixer).
- 8 Import sample** - 12 slots for WAV/RAW samples, optional ADPCM.
- 9 Total and chip bar** - size of all samples against the chip memory budget.
- 10 Sample view** - the rendered sample, zoomable, with loop marker.

Getting started

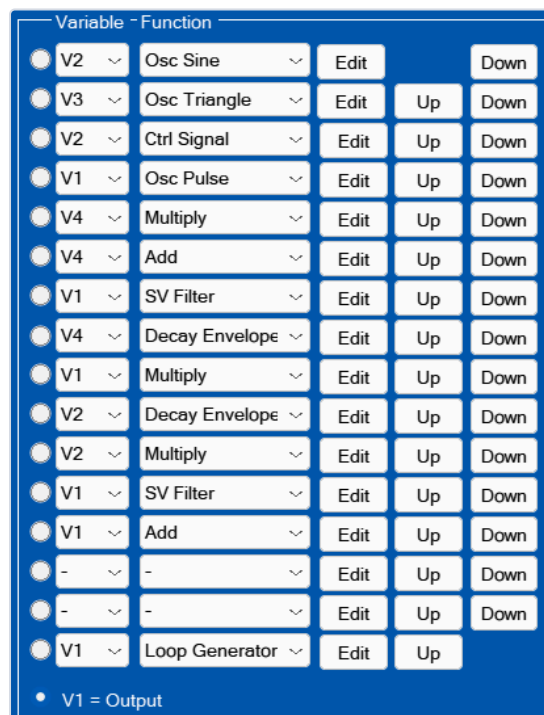
1. Start AmigaKlang.exe. For the executable exports the exe_creator folder must be next to it.
2. **Load** → **Load Patch** opens an .akp patch, for example from examples/patches.
3. Pick an instrument in the instrument box. Every change is rendered immediately in the background - there is no render button.
4. Play the instrument with the computer keyboard (see *Playing*). The window can be resized freely; everything scales and keeps its proportions.

Command line: AmigaKlang.exe [patch.akp] [--instrument 1..31] [--edit 1..16] [--design amiga|atari|archimedes] opens a patch, an instrument, the parameters of a slot and a design.

Files

Extension	Content	Menu
.akp	Patch: all 31 instruments, imported samples and sequencer patterns	Load Patch / Save Patch
.aki	A single instrument	Load Instrument / Save Instrument
.raw .wav	The current sample as 8 bit raw data or WAV; every used instrument as its own WAV file	Save Sample / Export all Samples as WAV
.mod	ProTracker module: the rendered samples can be written into it	Export Samples to mod
.xm	Extended Module (FastTracker 2, OpenMPT): the samples in 16 bit	Export Samples to XM

Building an instrument

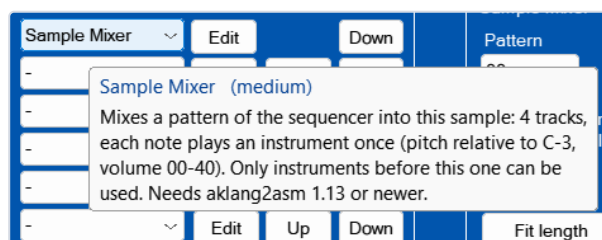


The 16 slots of an instrument.

An instrument is a chain of up to **16 slots**, processed from top to bottom for every sample.

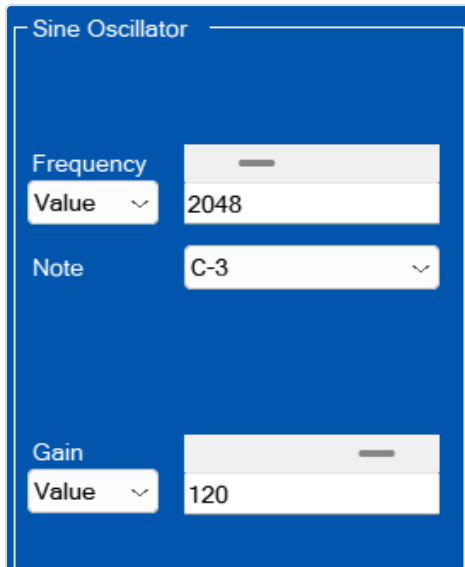
- **Variable** (V1-V4): where the result of the slot is stored.
- **Function**: oscillators, envelopes, filters, delay, reverb, distortion, clone, chord, imported sample, sample mixer, bytebeat, loop generator ...
- **Edit** shows the parameters of the slot in the middle column.
- **Up / Down** move the slot.
- The final result must end up in **V1** - that is the output.
- The circle on the left shows the output of that slot in the sample view; **V1 = Output** shows the final sample.
- An **orange circle** to the right of a slot marks a wiring error, a hollow one is a hint. Hover over it to read it.

Hover over any control: tooltips explain every function, parameter range and error.



Tooltip of a function.

Parameters



The Sine Oscillator interface features a blue background with white text. It includes a 'Frequency' section with a slider and a text box showing '2048', a 'Note' dropdown menu showing 'C-3', and a 'Gain' section with a slider and a text box showing '120'. Each parameter has a 'Value' dropdown menu next to it.

Parameters of a sine oscillator.

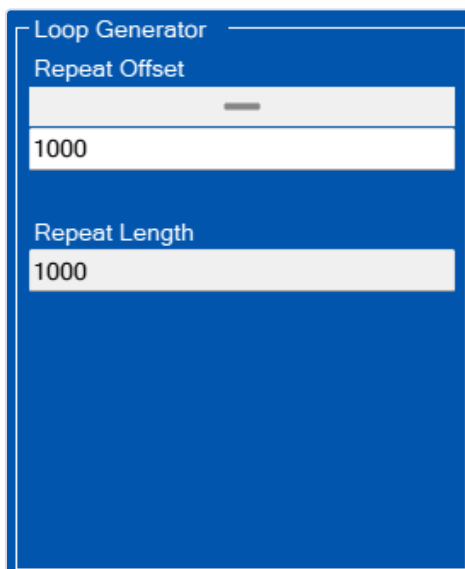
Each parameter has a slider and a text box. The combo box in front of it selects the source:

- **Value:** a fixed number (slider or typed, Enter confirms).
- **V1-V4:** the parameter is **modulated** by a variable, e.g. an envelope or a slow oscillator computed in an earlier slot.

The oscillators have a **Note** list below the frequency: a note sets the frequency. **C-3** is 2048, a wave of 32 samples - played at C-3 the oscillator sounds like a classic chip waveform at the chosen note. The list shows - while the frequency is between two notes; the mouse wheel steps through the notes.

Functions are tagged *fast*, *medium* or *slow* in their tooltip - this is the rendering time on the Amiga.

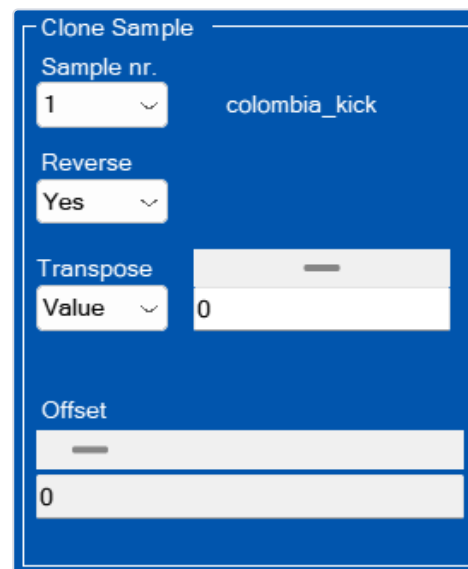
Loop generator



The Loop Generator interface has a blue background with white text. It features a 'Repeat Offset' section with a slider and a text box showing '1000', and a 'Repeat Length' section with a slider and a text box showing '1000'.

Put the **Loop Generator** into a slot to get a seamless loop. *Repeat Offset* is where the loop starts, the rest of the sample is the loop. The end of the loop is crossfaded automatically, so the loop start can be anywhere. In the sample view the loop is drawn darker and its start marker can be dragged.

Clone sample



The Clone Sample interface has a blue background with white text. It includes a 'Sample nr.' dropdown menu showing '1' and the text 'colombia_kick', a 'Reverse' dropdown menu showing 'Yes', a 'Transpose' section with a slider and a text box showing '0', and an 'Offset' section with a slider and a text box showing '0'.

Clone Sample reads an instrument that is rendered before this one - forward or reversed, from an offset, and **transposed** by up to one octave down or up. The transpose can be modulated by V1-V4 for pitch bends and vibrato.

Bytebeat (experimental)

Bytebeat

Formula

Examples

Gain

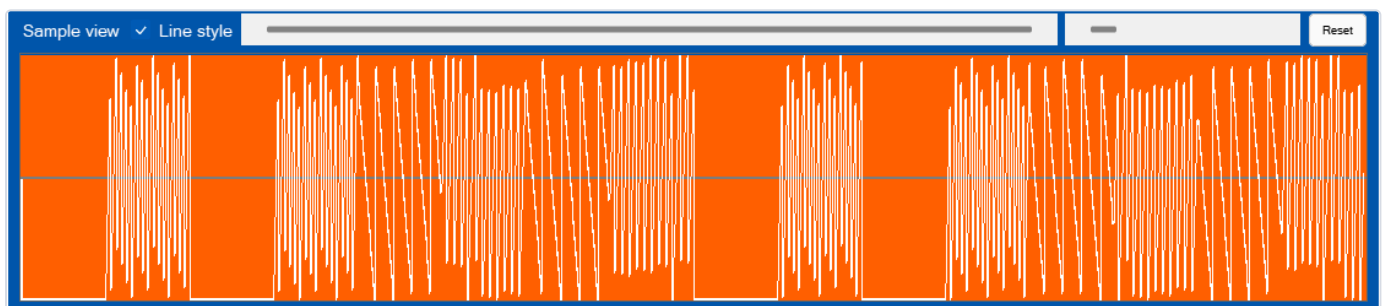
Value

The **Bytebeat** function makes the sample from a single formula over t , the number of the sample: $t*(42 \& t \gg 10)$ is a complete melody.

- **Formula:** type it and press Enter. An error is shown below the formula, and the slot gets an orange circle.
- **Examples:** classic formulas to start from.
- **Gain:** the volume, a fixed value or modulated by V1-V4 (128 = unchanged).

Numbers are decimal or hex (0x2A). The operators are those of C: $*$ $/$ $\%$ $+$ $-$ $<<$ $>>$ $< <=$ $> >=$ $==$ $!=$ $\&$ $^$ $|$ $\&\&$ $||$ $?:$, the signs $-$ $\sim$ $!$ and parentheses.

The formula is computed with **unsigned 16 bit** numbers (values wrap at 65536, a division by zero gives 0); the low byte of the result is the sample. The Amiga computes exactly the same samples. Needs aklang2asm 1.17.

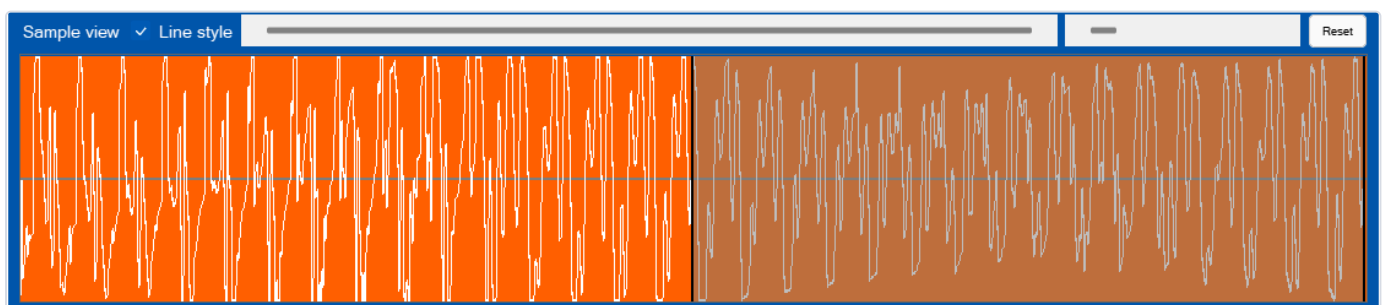


The sample of $t*(42 \& t \gg 10)$.

Sample length and sample view

Samplelength (Hex/Dec)

The sample length is set in hex or decimal (Enter) or with the slider. Shorter samples render faster and need less chip memory.



Sample view: mouse wheel zooms around the mouse, dragging scrolls, *Reset* shows the whole sample, *Line style* switches the drawing. The darker part is the loop; a play head runs while the sample plays.

Playing

Keyboard

Octave:

1

2

3

Keys Z..M, Q..P play the instrument

The computer keyboard plays the selected instrument like in ProTracker. The lower row plays the octave chosen with the **Octave** buttons, the upper row one octave higher. Holding a key keeps a looped sample playing. The preview sounds like Paula (no interpolation) with low latency.

The keys work by **position**, so the layout is the same on every keyboard:

Lower row	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
QWERTY	Z	S	X	D	C	V	G	B	H	N	J	M
QWERTZ	Y	S	X	D	C	V	G	B	H	N	J	M
AZERTY	W	S	X	D	C	V	G	B	H	N	J	,
Upper row (+1 octave)	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
QWERTY	Q	2	W	3	E	R	5	T	6	Y	7	U
QWERTZ	Q	2	W	3	E	R	5	T	6	Z	7	U
AZERTY	A	é	Z	"	E	R	(	T	-	Y	è	U

The rows continue to the right (**,** **L** **.** ... and **I** **9** **0** **0** **P** on QWERTY). Notes above B-3 are ignored, like in ProTracker.

Imported samples

Import Sample (32767 total max)

futurekickshort2

8E8

2280

Load

Clear

Total

27453

✓ ADPCM (4 bit, half the size)

total 93614

91K / 512K

Real samples can be mixed into instruments with the **Imported Sample** function. There are **12 import slots**: choose a slot (also with the mouse wheel), **Load** a WAV or RAW file, type a name and press Enter to rename it, **Clear** removes it. WAV files can be 8 to 32 bit or float, mono or stereo; they are mixed to mono and converted to 8 bit (up to 65535 samples). All imports together may use up to 32767 bytes in the executable (*Total*).

ADPCM stores a sample as 4 bit ADPCM: half the size, slightly lossy. The preview plays it exactly as the Amiga decodes it, and the patch keeps the 8 bit original, so the switch can be turned off again without loss.

The **chip bar** next to *total* shows all samples against the chip memory budget; click it to switch between 256K, 512K, 1M and 2M.

Sequencer and Sample Mixer

Sequencer

Pattern

< 00 >

Speed

< 03 >

	Track 1	Track 2	Track 3	Track 4
00	C-3 01 40	---	C-3 05 20	---
01	---	---	---	---
02	---	---	C-3 05 14	---
03	---	---	---	---
04	C-3 01 40	C-3 02 30	C-3 05 20	---
05	---	---	---	---
06	---	---	C-3 05 14	---
07	---	---	---	---
08	C-3 01 40	---	C-3 05 20	---
09	---	---	---	---
0A	---	---	C-3 05 14	---
0B	---	---	---	---
0C	C-3 01 40	C-3 02 30	C-3 05 20	---
0D	---	---	---	---
0E	---	---	C-3 05 14	G-3 0D 18
0F	---	---	---	---

Click a cell to enter notes, Esc to stop

Editing row 05, track 2 (orange row, white cursor).

The sequencer holds **16 patterns** of 4 tracks × 16 rows in ProTracker style: C-3 01 40 = note, instrument, volume (00-40 hex). **Pattern** < > selects the pattern, **Speed** < > the speed (01-1F ticks of 1/50 s per row). Patterns are saved in the .akp.

Click a cell to start editing:

- Note column: the tracker keys enter the note with the **instrument selected at the top left** and play it.
- Instrument and volume: hex keys 0-9, A-F.
- Del clears (in the note column the whole cell), arrows / Tab / PgUp PgDn / Home End move, Esc stops editing.

Sample Mixer

Pattern

00

Speed 03, 16 rows

= 15904 samples

Instruments:

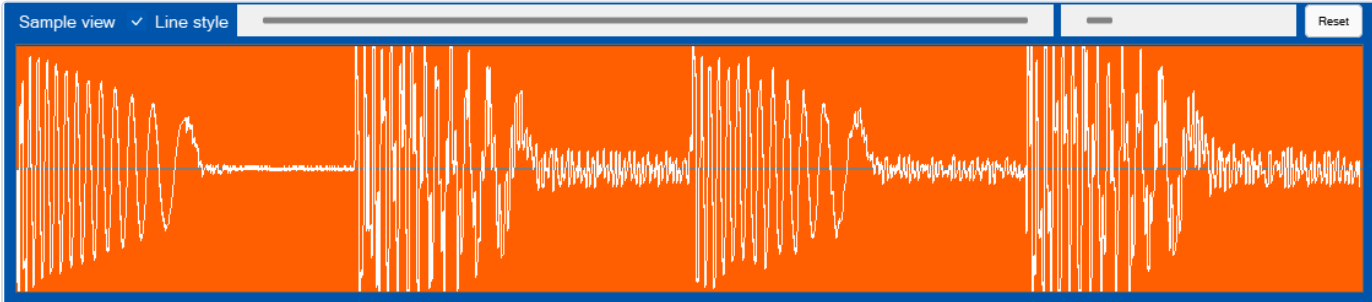
01 02 05 0D

Fit length

Sample Mixer

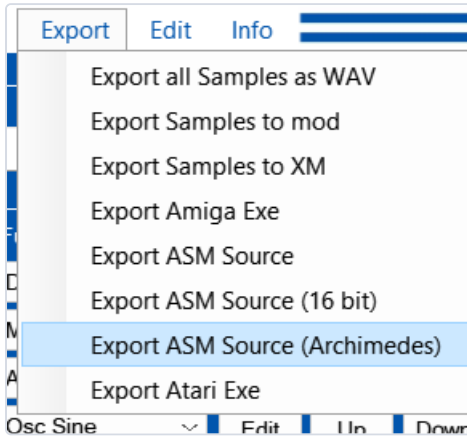
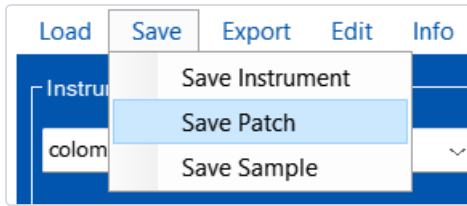
The **Sample Mixer** function mixes a pattern into the sample of an instrument - drum loops, chords, arpeggios, layered sounds. Choose the **Pattern**; the panel shows speed, rows and the resulting length plus the instruments used. **Fit length** sets the sample length to the pattern length.

Each note plays its instrument once, the pitch is relative to C-3. Only instruments **before** the mixer instrument can be used, because they must already be rendered; later ones are shown in red.



The mixed result of the pattern above: kick, snare, closed hihat and a pling as one sample.

Saving and exporting

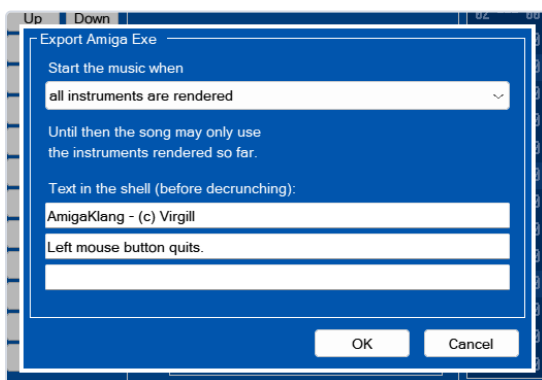


- **Save Instrument / Save Patch** - see *Files*. **Save Sample** writes the current instrument as RAW or WAV.
- **Export all Samples as WAV** writes every used instrument: 8 bit mono at 16574 Hz (the note C-3), with the loop of the loop generator, so samplers and trackers loop it correctly.
- **Export Samples to mod** writes the rendered samples, lengths and loops into a ProTracker module. In an existing module **all sample data is overwritten**, its song stays; a new file name makes a new module.
- **Export Samples to XM** writes the instruments as **16 bit** samples into an Extended Module. An existing file keeps its song and instrument settings, a new name makes a new module. The samples are rendered again in 16 bit, not scaled up from 8 bit. C-5 there is C-3 in ProTracker.
- **Export Amiga Exe** builds a standalone Amiga executable of the song (below).
- **Export ASM Source** writes `exemusic.asm` with the 68000 routine `AK_Generate` that renders all instruments, for your own code. The **16 bit** entry writes `exemusic16.asm` for Platon42's UNZ player (a routine per instrument, 16 bit samples), the **Archimedes** entry `arcmusic.asm` with ARM code.
- **Export Atari Exe** builds an Atari ST program.

All exports are made by **aklang2asm** (Dan/Lemon) and the tools in the `exe_creator` folder; a patch with wiring errors can't be exported.

Amiga executable: render while playing

The Amiga executable renders all instruments with the 68000 code of `aklang2asm` and plays the module with Leonard's **Light Speed Player**. While rendering, a progress bar grows along the bottom of the screen; during playback four VU bars light up per channel.



When the music starts, and the text for the shell.

1. Compose the song in ProTracker with the samples of the patch (see *Export Samples to mod*).
2. **Export** → **Export Amiga Exe**.
3. Choose **when the music starts**: when all instruments are rendered, or as soon as a certain instrument is rendered. The choice is remembered until the editor is closed.
4. Edit the **text in the shell** (up to three lines). The executable prints it before it decrunches, so it can be read while it unpacks. The text is saved in the patch.
5. Pick the `.mod` file. A console window shows the build; `exemusic.exe` appears in the `exe_creator` folder.

Requirements

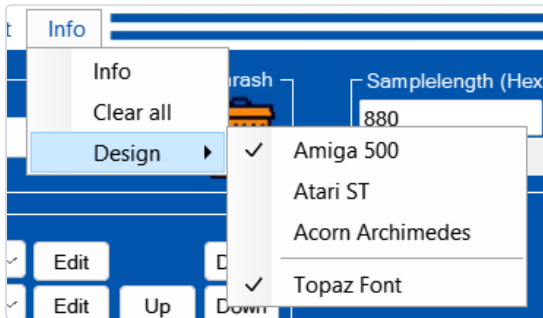
- Runs on every Amiga from Kickstart 1.3 and 68000 on (tested A500 KS 1.3/3.1, A1200). The left mouse button quits - also while rendering.
- The tools are in the `exe_creator` folder of the release: `aklang2asm 1.22` (1.16 is enough without Bytebeat), `LSPConvert 1.33` in `bin\lsp`, `vasm` and `Shrinkler`. The example song (25 instruments) packs to about 10 KB.

Render while playing on the Amiga

With a start instrument, the music starts as soon as that instrument and all before it are rendered. The remaining instruments are rendered while the music plays. Compose the first patterns with the instruments up to that number - sort the instruments so that the intro uses the low numbers - and the precalc time becomes very short.

A one-shot sample gets its silent first two bytes as soon as it is rendered. Instruments that clone, chord or mix it later see these two zero bytes at its start - just like the original AmigaKlang exe.

Designs and window

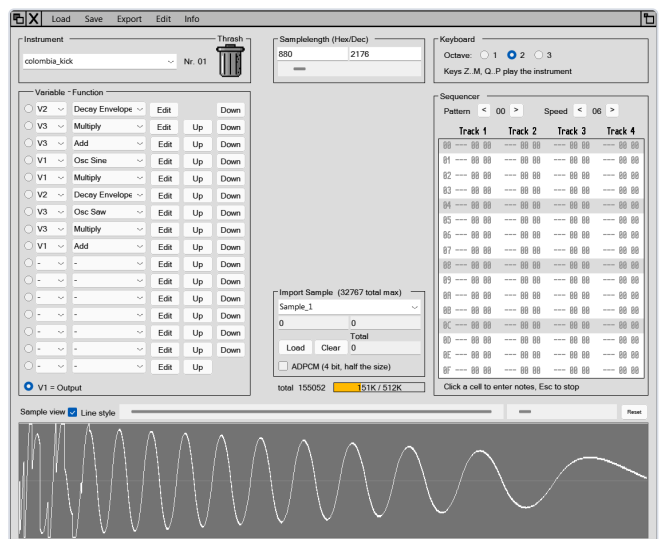


Info → **Design** switches between three looks: **Amiga 500**, **Atari ST** and **Acorn Archimedes** (RISC OS 3), each with its own colours, title bar, bin and mouse pointer. **Topaz Font** uses the Amiga font for the whole editor (the sequencer always uses it). **Info** → **Clear all** deletes all instruments.

The window keeps its proportions when it is resized or snapped to a screen edge. A **maximized** window fills the screen: the editor is centred on the desktop colour of the design.



The Atari ST design.



The Acorn Archimedes design.

Key reference

Keys	Action
Lower / upper letter row	Play the instrument (octave from the <i>Octave</i> buttons, upper row one octave higher)
Ctrl + Z	Undo
Ctrl + Y or Ctrl + Shift + Z	Redo
Enter	Confirm a typed value or name
Mouse wheel over the sample view	Zoom around the mouse
Mouse wheel over a slider or its number	Value up / down, with Shift a large step
Mouse wheel over a list	Previous / next entry: instrument, import slot, mixer pattern, sample number (Clone, Chord, Imported Sample), chord note, oscillator note, filter mode, Bytebeat example
While editing the sequencer (click a cell)	
Letter rows	Note column: enter the note with the selected instrument and play it
0-9 A-F	Instrument and volume (hex, volume 00-40)
Del / Backspace	Clear the field (the whole cell in the note column) and go to the next row
↑ ↓ / ← →	Previous / next row, previous / next field
Tab / Shift + Tab	Next / previous track
PgUp PgDn / Home End	4 rows up / down, first / last row
Esc	Stop editing

Tips

- Watch the chip bar: everything the instruments render must fit into the chip memory of the target Amiga.
- Slow functions (reverb, comb filter, long delays) cost rendering time on the Amiga - prefer short samples and loops.
- Clone, Chord Gen and Sample Mixer reuse instruments that are already rendered, so order your instruments from the building blocks to the combined sounds.
- Settings (window size, Topaz font, chip budget) are stored in %AppData%\AmigaKlang\settings.json.

Credits

Virgill	AmigaKlang
Dan / Lemon	aklang2asm (68000 code generator)
kieran / Bitshifters	ArchieKlang: ARM code generator for the Acorn Archimedes
Leonard / Oxygene	Light Speed Player and LSPConvert, Atari playback
Blueberry / Loonies	Shrinkler executable packer
Gopher / Alcatraz, Hellfire / Haujobb, Bartman / Abyss	node architecture, code improvements, tools

Greetings to Chlumpie, Tecon, JosSs and Teo.