Tranzistow v4.8.0 [32-bit & 64-bit Windows VST 2.x + Linux Standalone / Experimental VST 2.x]

http://www.hrastprogrammer.com/tranzistow/

Note: This is not user manual but a brief Tranzistow introduction and description. I simply don't have enough free time to describe all Tranzistow features and parameters because I have to do other job for a living :-(

Some comments on Tranzistow:

"I will just say that it's certainly impressive the amount of work you've put into your projects, both the synths and the music! Incredible amount of output musically, and so I know you are passionate about it. And I must say, it's very excellent work! ... I see you did get some inspiration by my Solaris, which is a very nice compliment ... Let me say again how impressive your music production is! It's a lot of work, and beautifully done. I know when I first was making the Scope plugins, each time I would sit and play, sometimes for hours, through certain sounds that just seemed to inspire a musical phrase or a dream idea ... And so much of what is now being produced out there is, to be honest, boring to me. Just running a 16 step sequencer and adjusting filter cutoffs and so on - there's nothing 'skillful' about that, and there's so much of it going on now. But your music is so much more - very deep soundscapes! And the Solaris is very much that type of instrument, with a focus on pad sounds and textures, so I hear what you've done with your Tranzistow synth and I'm quite amazed!" John Bowen (the creator of Solaris synthesizer, http://johnbowen.com/)

"I wanted to reaffirm my total appreciation of the DSP programming work you have done with Tranzistow. I have many great synths (hardware, software, analog, digital) but nothing quite compares to the organic qualities that I have been able to coax from your brilliant synthesizer. You have done great work with Tranzistow, make no mistake about it. I can't thank you enough for this brilliant addition to my musical life."

"You can add me to the list of people who consider your Tranzistow one of the best software synth available. It is such a pity it is so obscure, more people should play with it. You deserve much more exposure. I really hope some company will acknowledge your talent and offer you the chance to implement your ideas in hardware."

"Modal should hire HrastProgrammer. I tried wavescaning on his Tranzistow synthesizer, it is by far the best software implementation i've heard so far. You can actually hear that crunchy graining thing where so many others fail."

"Tranzistow is slowly becoming a go-to synth for me. Looks like an 80s NASA control interface, sounds like a dream."

"Tranzistow is a monster. The guy programmed it all in assembly language and some Delphi, even John Bowen praises his work. Check it out! Be warned ... Tranzistow is hard core, haven't even figured it all out. It can do every form of synthesis. The programmer is nothing short of a genius."

"Somebody gave me the URL to these synths because of how bat shit ugly they look. When I took the time to actually dive into them, I found a couple of real gems that turned out to be my favorite synths of everything I've ever owned; bought or otherwise."

"Hrast is, for me, one of the best dev with Gol, Urs and Dmitry from Diversion and Richard from Synapse."

[*] What is Tranzistow?

In global, Tranzistow (this is not a mistake - it is not "Transistow") is a virtual software synthesizer with 4 or 8 independent multi parts. Although it is (to some extent) inspired by Alesis Andromeda, Waldorf Q, John Bowen Solaris, Oberheim Xpander, Waldorf Microwave II/XT and Yamaha DX series, it has a unique sound and tons of unique features not found on any other synthesizer. Various forms of synthesis are supported including Additive, Virtual Analog/Modeling (VA), Wavetable (with classic, crossfade and full wave interpolation modes), two completely independent FM engines, Ring Modulation, Oscillator Sync, Vectors, Rotors (as in Solaris), various filter models (SVF 12/24dB, 6dB to 24dB Transistor Ladder with fully configurable poles, 24dB Diode LowPass Ladder, ...), Filter Chaining, Comb Filters, WaveShaping, freely drawable Contour Generators, exhaustive FX section including Chorus/Flanger, Phaser, Delay and Reverb units, etc, etc.

The following Windows VST 2.x versions currently exist:

(*) Tranzistow v4/32 ... 32-bit, 4 parts with SSE3 engine and 4-way multithreading
(*) Tranzistow v4/64 ... 64-bit, 4 parts with SSE3 engine and 4-way multithreading
(*) Tranzistow v4+/32 ... Tranzistow v4/32 with Additive+FM/GPU engine
(*) Tranzistow v4+/64 ... Tranzistow v4/64 with Additive+FM/GPU engine

(*) Tranzistow v8/32 ... 32-bit, 8 parts (4 main + 4 sub) with SSE3 engine and 4-way multithreading
(*) Tranzistow v8/64 ... 64-bit, 8 parts (4 main + 4 sub) with SSE3 engine and 4-way multithreading
(*) Tranzistow v8+/32 ... Tranzistow v8/32 with Additive+FM/GPU engine
(*) Tranzistow v8+/64 ... Tranzistow v8/64 with Additive+FM/GPU engine

(*) Tranzistow x8/64 ... 64-bit, 8 parts (4 main + 4 sub) with AVX2 engine and 4-way multithreading
(*) Tranzistow x8+/64 ... Tranzistow x8/64 with Additive+FM/GPU engine

The following Linux standalone and experimental native VST 2.x versions currently exist:

(*) Tranzistow v4/32 ... 32-bit, 4 parts with SSE3 engine and 4-way multithreading
(*) Tranzistow v4/64 ... 64-bit, 4 parts with SSE3 engine and 4-way multithreading
(*) Tranzistow v8/32 ... 32-bit, 8 parts (4 main + 4 sub) with SSE3 engine and 4-way multithreading
(*) Tranzistow v8/64 ... 64-bit, 8 parts (4 main + 4 sub) with SSE3 engine and 4-way multithreading

(*) Tranzistow x8/64 ... 64-bit, 8 parts (4 main + 4 sub) with AVX2 engine and 4-way multithreading

All versions have 64 voices (4 parts) or 128 voices (8 parts) total polyphony in multithreading mode and 24-voice total polyphony in singlethreading mode (8 parts are not available in singlethreading mode). Of course, the real polyphony (= maximum polyphony which can be achieved under real conditions) depends greatly on the complexity of the sounds and the speed of the CPU. 64-bit SSE3 versions have 5-8% bigger polyphony compared to 32-bit versions. AVX2 versions have 40-60% bigger polyphony compared to 64-bit SSE3 versions, depending on the patch structure.

I wanted to achieve the maximum possible sound quality and the maximum possible amount of features, so Tranzistow can use quite a lot of CPU, but there are various tips to lower CPU consumption in cases where maximum quality is not really needed, especially if working with 96kHz sample rate (which is the recommended sample rate for Tranzistow).

I developed Tranzistow for myself in the first place. My goal was not to (try to) emulate existing (hardware or software, analog or digital) synthesizers (although I was inspired by some of them) but to create something I can actually use in my own music, to master DSP/Assembler/Vector/SSE/AVX/AVX2/GPU/OpenCL programming, to refresh my mathematics knowledge and, last but not least - to have fun. As a result, the synthesizer is mostly unconventional, some features (which most users take for granted) could be missing or could look rather strange, user interface may not be everybody's cup of tea because it was designed according to my habits/needs, etc.

[*] Conditions of Use

(*) Tranzistow is my private project and I developed it for myself in the first place. So, if you don’t like it then don't use it, simple as that ;-) I really don't have the time and patience to argue about something someone doesn't like, or whether "it is or isn't analog enough" etc.

(*) I've done my best to make it 100% compatible with the VST specification but if it still doesn't work (or doesn't work well) under some VST hosts then - bad luck, sorry. There are tons of various VST hosts out there and the exploration of all sorts of VST hosts quirks is almost "mission impossible".

(*) The synthesizer is "feature finished" - no new features will be added so, please, don't ask for a new feature, user interface changes, VST3 support, additional VST automation support, VST preset management, etc. I also don't have any plans to support AU, AAX, RTAS or any other platform beside VST. But, if you want to finance the development for any of the aforementioned feel free to contact me.

(*) Only 32/64-bit Windows VST 2.x, Linux standalone and experimental Linux native VST 2.x versions are currently available. No other versions (VST 3.x, Macburger, AU, RTAS, AAX, iCrap, dumbphones and similar) are planned.

(*) You agree that your name appears somewhere on the screen stating that you are a registered user.

(*) It can be installed/used on a reasonable number of computers. By "reasonable" I mean a few computers you own, not the whole studio shared by a few artists or a classroom, for example.

(*) I am not responsible for anything you do with this software - use it at your (and only your) own risk!

(*) You may not alter this software in any way, including changing or removing any messages.

(*) You may not decompile, reverse engineer, disassemble, modify, distribute, rent or resell this software, or create derivative works based upon it.

[*] Technical Details

(*) Requirements: 32/64-bit Windows VST 2.x host and SSE3/AVX2 capable 32/64-bit CPU.

(*) The complete low-level audio/processing/control/MIDI/FX engine is written in 100% Intel x86/x64 assembler utilizing SSE3/AVX2 instructions for vector processing - that's the main reason why there are 4 (and multiples of 4) elements of each module (oscillators, filters, etc.) because I am processing 4 vector elements in parallel.

(*) 32-bit integers and single-precision floating point numbers are used for all processing as a proof that, if done properly, you don't need 64-bit and double-precision to achieve a high-quality sound. The only place where I had to switch to double-precision is the sample engine because I wanted sample oscillators to be able to handle large files with a high resolution.

(*) User interface and all other high-level tasks are written in Borland Delphi and Free Pascal / Lazarus.

(*) The engine is optimized for 96kHz sample rate. Other sample rates can be used, of course, but 96kHz is the only officially supported sample rate. Most high-end hardware synthesizers (like Solaris, for example) use 96kHz sample rate as well, so I didn't want to make compromises here. On today's computers it doesn't have much sense to intentionally degrade a sound by using lower sample rates just to save some CPU, and 192kHz is really an overkill, especially with 2x and 4x oversampling.

(*) Some internal wavetables are modeled to sound similar to various wavetables from existing hardware synthesizers, but they are generated in a different way and they are not copies of wavetables from those synthesizers. They are also of much higher quality compared to usual wavetables (due to highest-quality wavetable generation algorithms in Tranzistow).

[*] Installation

Just unpack files from the ZIP file into your 32-bit and/or 64-bit VST directory and that's it - your VST host should recognize it and you will be able to use it.

[*] Demo Version

Demo version is, more or less, a fully functional Tranzistow with the following limitations:

(*) No multimode and no multiprocessing - it is monotimbral, singlethreaded and only Part #1 with 16 voices of polyphony is available. External Control Interface is disabled as well.

(*) It loads the default patch on every start, so you'll have to manually reload all patches every time you open/reload a particular project.

(*) No ability to process external inputs through Tranzistow FX units, so it cannot be used as a standalone FX processor. External inputs can still be processed through the voice engine, though.

(*) Occasional noise and clicks/pops now and then, together with occasional missed notes. This is just a slight annoyance which won't have a big impact on the sound and won't disturb the "demonstration purpose" of the synthesizer, but will prevent its serious usage because you don't want to have clicks/pops and missed notes in your tracks, do you?

(*) Limited to 48kHz sample rate on Linux.

Demo downloads are here:

http://www.hrastprogrammer.com/hrastwerk/download/TranzistowDemo.zip

http://www.hrastprogrammer.com/hrastwerk/download/TranzistowLinuxDemo.zip
(see Linux section at the end of this document for more info about Linux version)

[*] Beyond Demo

After trying the demo, some of you could perhaps want to have a registered version without the limitations mentioned above. Tranzistow is not a free software and I put a lot of my own time, knowledge and energy into this project. Unfortunately, I don't have a business which can sell it to you. So, it is not available for purchase until someone takes over the commercial side of this project.

But, I would be very happy to give a free Tranzistow license (either Windows or Linux or both) to anyone under the following conditions:

(*) That you created a bank of at least 200 of your own original Tranzistow/Diodow patches.
(*) Or you made some other valuable Tranzistow contributions like full manual, exhaustive tutorials, etc.

Basically, you have to prove that you already spent a significant amount of time with Diodow or Tranzistow demo, that you really want to use Tranzistow in the future and that I get something valuable in exchange. I reserve the right to reject all license requests which don't fully meet the above conditions. For example, just tweaking my patches and giving them new names doesn't really qualify as "original patches" ;-)

I made a mistake in the past and gave a free Tranzistow license to some people who did exactly nothing with Tranzistow and I got exactly nothing in exchange. They couldn't even spend a couple of minutes and write a few words about Tranzistow on the forums! So, it didn't make any sense to continue with this practice and I won't make such mistakes anymore. Make something with Tranzistow or Diodow and you'll get a license, be sure about that.

There is another way to obtain a registered copy but I want to keep this private and anyone who wants to know more about it can contact me by mail.

[*] Final Words

Tranzistow is a synthesizer for pure electronic artists, sound programmers, synth lovers and creators of true electronic music, it doesn't pretend to be anything else. So, if you need beautiful, unique, powerful and most complex synthetic electronic sounds of the highest quality - then you are at the right place. It isn't really aimed at keyboard players and, although it can be used for such duties, there are much easier ways to obtain the usual keyboard sounds. It is like buying a ultra-capable 4x4 offroad vehicle and use it for everyday driving - isn't it simpler and much more convenient to use a small city car to go shopping?

[*] Patches / Presets

The Tranzistow package (both demo and registered versions) comes with over 3500 of my own patches which can be easily browsed through by clicking on the "Browse" button located on the "Patch/FX" page. Those patches illustrate the way I work and the way I use synthesizers for, so they probably don't appeal to everyone. They also demonstrate a huge range and variety of sounds Tranzistow can make and cover most of Tranzistow functionality, albeit some features are touched very lightly and just in a few patches. Unfortunately, I ran out of time while preparing Tranzistow for production and couldn't allocate more time for patch creation :-(

Tranzistow is a hugely complex and powerful synthesizer which looks and feels more like a fully equipped Land Rover Defender with V12 diesel engine than a BMW Mini or Mercedes C-Klasse. So, most of those patches are very complex and many of them have extralong envelope tails and/or use unison which can eat quite a lot of CPU, although none of them use more than 80% of one CPU core on my i7-4770K @ 4.4GHz with 16 voices of polyphony, 96kHz sample rate and 2x oversampling (this is 64 extremely complex voices across 8 multimode parts on a quad core system at 96kHz, or 128 voices at 48kHz, all with 2x oversampling). There are some advices on how to optimize CPU usage later in this document. Of course, with less complex voices CPU usage will be much lower.

Regarding patch names: I had to name them somehow and you will find everything here. Some names do have a connection to the sound itself and some don't, so if you find something like "analog", "Moog" or "whatever" it just means that a particular sound reminded me somehow of "analog", "Moog" or "whatever". It doesn't mean it sounds exactly like "analog", "Moog" or "whatever" because I couldn't care less if it sounds like "analog", "Moog" or "whatever", neither I tried Tranzistow to be emulation of "analog", "Moog" or "whatever". Furthermore, due to artistic freedom, many names are combinations of words which are not necessarily syntactically or semantically correct, and which don’t necessarily have any meaningful sense to anyone except me.

(*) Here is another bank with ~28000 Yamaha DX7 patches I reworked for Tranzistow QFM engine:

http://www.hrastprogrammer.com/hrastwerk/download/TranzistowQFM.zip

(*) Great bank of Tranzistow patches (and some other goodies like user waves, contours and samples) made by Nicolas Jaussaud (aka Yuli Yolo):

http://www.hrastprogrammer.com/hrastwerk/download/TranzistowYY.zip

(*) Diodow bank with 100 patches made by Ed Ten Eyck (www.edtaudio.com) can be used with Tranzistow as well:

http://www.hrastprogrammer.com/hrastwerk/download/DiodowEDT.zip

Quite a few of EDT patches are using sample engine (mostly in "LA Style") which I didn't even touch in my own patches.

[*] Building Blocks & Modules

Starting on the next page ...

[*] Main Oscillators (Osc/Main Page; Osc #1 ... Osc #4 Sections)

There are 4 main oscillators based on Additive, Wavetable and Virtual Analog/Modeling (VA) engine. Those can go from 0Hz to 24kHz and are fully antialiased with a powerful, fat bass and clean & crispy highs. Every oscillator has its own internal sync oscillator, so all of them can be synced at the same time if needed. All of them can be frequency modulated by any other audio/modulation source inside the synthesizer. Additive part of the engine can produce up to 1024 harmonics and automatically removes harmonics over Nyquist when pitching up, and adds harmonics when pitching down. Wavetable part of the engine can work without any crossfading/interpolation between individual waves (Wave and Wave+Pulse/Ramp modes, this is "classic" wavetable mode with integer wave index as found in Waldorf Microwave, for example), with crossfading between individual waves (XWave and XWave+Pulse/Ramp modes), with interpolation between individual waves ([Wave] and [Wave+Pulse/Ramp] modes) as well as with both crossfading and interpolation between individual waves ([XWave] and [XWave+Pulse/Ramp] modes). Crossfading (XWave) modes are also used when pitching up/down for smooth transitions when adding or removing harmonics. Basic waveform of the oscillator can be shaped into pulse/square and ramp/triangle waveforms, which can then be combined with the basic waveform (in Pulse/Ramp modes). Inside each oscillator there is a thermal drift module for a more analog sound. Main oscillators can use a lot of CPU power in wave/crossfade and wave/interpolation modes.

In addition to built-in waves/wavetables, there are 27 user banks with 100 waves each (User00.wave ... User99.wave files in TranzistowWaves, TranzistowWavesA ... TranzistowWavesZ directories). User00.wave from the main bank (TranzistowWaves) contains all Ensoniq SQ waves while all others are free to use and will be loaded automatically if one exists. The format of user wave files are out of the scope of this document.

Furthermore, there are four user-definable wavetables per patch, each containing from 1 to 128 individual waves with size of up to 256 samples, bandlimiting and interpolation of missing waves. All internal and user waves can be used here. Those wavetables are accessible through WT #1 ... WT #4 buttons after activating them by clicking on the "Osc" button/section from "Osc/Main" page.

[*] Auxiliary Oscillators (Osc/Aux Page; AuxOsc #1 ... AuxOsc #4 Sections)

There are 4 auxiliary oscillators based on a different Virtual Analog/Modeling (VA) engine. Those can go from 0Hz to 24kHz and are fully antialiased with a powerful, fat bass and clean & crispy highs, just like main oscillators. Every aux oscillator has its own internal sync oscillator, so all of them can be synced at the same time if needed. All of them can be frequency modulated by any other audio/modulation source inside the synthesizer. They are also partially connected to additive and wavetable engines, so some features of main oscillators are present here as well. Basic saw waveform of the oscillator can be shaped into pulse/square and ramp/triangle waveforms, which can then be combined with the basic waveform (in Pulse/Ramp modes). Inside each oscillator there is a thermal drift module for a more analog sound. Aux oscillators use less CPU power and are thus recommended when full additive/wavetable functionality is not needed.

[*] Sine Oscillators (Osc/Main & Osc/Aux Pages)

Both main and auxiliary oscillator modules can be switched to sinewave module with 4 parallel sine oscillators which can be synced and frequency modulated just as Main/Aux oscillators and have the same set of basic features (but without thermal drift). Sine oscillators are antialiased by definition and use little CPU power. They are very useful with frequency modulation synthesis or just to beef a sound.

[*] SuperSaw Oscillators (Osc/Main & Osc/Aux Pages)

Both main and auxiliary oscillator modules can be switched to SuperSaw module with 4 parallel fully bandlimited SuperSaw oscillators which can be synced and frequency modulated just as Main/Aux and Sine oscillators and have the same set of basic features, including thermal drift module. Those don't have some advanced features like wavetables, morph, etc. but they are much more CPU friendly and can be used when advanced functionality is not really needed. Furthermore, bandlimiting filter can be freely adjusted so you can have variable amount of aliasing, ranging from no aliasing at all down to the zero point and terrible aliasing like in the good old days of lo-fi digital synthesis :-)

Each SuperSaw oscillator consists of up to 16 individual (antialiased) saw sub-oscillators (this is 64 saw sub-oscillators per voice) detuned against each other for extremely thick sound very popular in Trance genre, but perfectly usable in all other EM genres as well. Various detune type/level modes are provided and all sub-oscillators are synced individually for a huge sound. Depending on the number of sub-oscillators, SuperSaws can use quite a lot of CPU power. It is very important to note that Tranzistow will always allocate enough CPU for the maximum number of sub-oscillators among all oscillators, so it is much more effective to have 4 SuperSaw oscillators with 4 sub-oscillators each than one SuperSaw oscillator with 16 sub-oscillators and other three oscillators set to zero.

[*] HyperSaw/HyperWave Oscillators (Osc/Main & Osc/Aux Pages)

This is the result of my experimentations with very high sample rate for oscillators. Those oscillators are not bandlimited but instead work at much higher sample rate internally. Hypersampling rate can be adjusted and, by default, on 44.1kHz with 2x oversampling they work at 1.4MHz while on 96kHz with 2x oversampling they work at 2.3MHz. The main advantage of such configuration is that sweeping the frequency of the master oscillator with oscillator sync is continuous while on the other oscillators it is somewhat quantized which is noticeable on very high frequencies. Not that I ever sweep master oscillators (and especially not at very high frequencies), because it sounds "meh", but anyway - if it is needed then HyperSaw oscillator is the best candidate for this. The disadvantages are high CPU usage as well as some aliasing at lower sample rates and very high oscillator frequencies.

HyperWave oscillators use waves and wavetables instead of a saw wave to generate the sound in the same way as on the legendary PPG Wave synthesizers.

[*] FormantSaw Oscillators (Osc/Aux Page)

Those are a special sort of oscillators which generate sound through some kind of formant simulation. Basically, formants are resonant spectrum peaks and they sound similar to the combination of frequency modulation and bandpass resonant filters. Number of formants per oscillator can be set from 1 to 8 and they can be morphed from zero to full frequency giving them somewhat "throaty" and "vocal" quality. Formant oscillators have an internal feedback loop and respond to sync and FM as well.

[*] Oscillator Sync (Osc/Main & Osc/Aux Pages)

As said before, each Main/Aux oscillator has its own internal virtual sync oscillator, so all of them can be synchronized at the same time if needed. Oscillator sync is fully antialiased. In addition to internal sync oscillators, each oscillator has a "classic" inter-oscillator sync module where any (slave) oscillator can be synchronized to any other (master) oscillator, similar to classic analog synthesizers but much more powerful because you can have up to 8 sync master/slave oscillators per voice. I still recommend using virtual sync oscillators wherever possible because they are less CPU demanding. SuperSaw/HyperSaw/FormantSaw oscillators can also act as masters and slaves.

[*] Sample Oscillators (Osc/Main Page; Osc #1 ... Osc #4 Sections)

There are 4 high-quality sample oscillators per voice implemented as sub-modules inside main oscillators - they share common pitch/frequency parameters together with the same outputs, so they can be used as audio/modulation sources in tandem with the regular oscillators + sample output can be further processed through Tranzistow engine as any other audio source inside the synthesizer. Sample oscillators can be upsampled up to 32x, with up to 4 antialiasing filters and two interpolation modes (linear and high quality), so a required balance between a sound quality and CPU usage can be achieved for a particular application. The result is that you can transpose a sample up for several octaves without audible/measureable aliasing artifacts, even at lower quality settings and low CPU usage.

The engine can load 8/16/24/32-bit PCM and 32-bit FP mono/stereo WAV files. To load a sample click on the oscillator section name (Osc #1, Osc #2, Osc #3, Osc #4). In case of stereo samples - if you load a sample into oscillators 1 & 3 then a left channel will be loaded, otherwise a right channel will be loaded. To remove the sample press and hold Ctrl key and click on the appropriate oscillator section name. The oscillator section name will have an asterisk on front of it if a sample has been loaded into the connected sample oscillator. Thermal drift is not available with sample oscillators but FM and sync are fully supported. Those are mostly useful with short samples (cycles) but if applied creatively they can be used with longer samples as well.

Furthermore, sample engine contains a GrainStatic module. This is an extension which uses sample oscillators to produce varius granular effects and came as a result of my research in granular synthesis field. Both granular and sample oscillators can be mixed inside a single voice. FM and sync are implemented for granular oscillators as well.

[*] Oscillator Frequency Modulation (Osc/Aux & Mixer/FM Pages; Osc FM & AuxOsc FM Sections)

All Main/Aux oscillators can be frequency modulated by any audio or modulation source, including the oscillator itself (so FM feedback is possible). Each oscillator is actually one FM operator (either modulator or carrier) with two independent FM inputs. Tranzistow frequency modulation works as "Linear FM" aka "Yamaha-style FM" which is in fact a phase modulation but people usually associate FM with "Yamaha FM", so I will stick to this terminology. Real exponential FM is also possible through audio rate modulation matrix. Furthermore, phase locking has been implemented for all oscillators. It is not very useful alone (because phase is not advancing and thus the particular oscillator is not producing any sound) but if used with FM it will, more or less, act as phase distortion (as used by Casio, for example) opening a new array of possible sounds.

[*] QFM - Advanced Frequency Modulation (QFM/1 & QFM/2 Pages; OP #1 ... OP #8 Sections)

In addition to oscillator FM, Tranzistow has an advanced FM engine called QFM which consists of 8 operators grouped into two 4-OP FM modules (OPX and OPY), so only one module can be used if only 4 operators are needed to conserve CPU power. Those modules work in parallel with the existing oscillators - all operators can be processed through the rest of Tranzistow engine and used as audio/modulation sources, just like all other Tranzistow audio/modulation sources. QFM engine is totally independent of Main/Aux oscillator FM modules, so all of them can be used at the same time. Each QFM operator is an advanced version based on Yamaha FM operators and has all the usual FM parameters with lots of additional features. Operator matrix for both OPX (OP 1..4) and OPY (OP 5..8) modules is freely configurable meaning that you can use everything as FM sources - including other operators, oscillators, filters, etc. Furthermore, there is an additional fixed operator routing matrix (in 4x4-OP and 8x8-OP modes) where each 1..4 operator (OPX module) can be routed to each 1..4 operator and each 5..8 operator (OPY module) can be routed to each 5..8 operator, all with configurable amounts. Each operator has its own (configurable) level curve and can modulate itself so you can have feedback on all of them. There is a set of 32 DX7 and 8 DX9 algorithms - when you choose some of them the operator matrix is configured to match the appropriate algorithm, but you are free to reconfigure it afterwards. And each operator has its own 10-stage loopable envelope which can ba used as the regular modulation source as well. Operator waveform is not fixed so you can use not only sine wave but all other waveforms available in Tranzistow, including waves, contour generators, rotors, etc.

I converted thousands of Yamaha DX7 patches into Tranzistow format. Due to various differences (operator level scaling, interpolation, much higher waveform quality, etc.) between Yamaha and Tranzistow FM engines those patches won't necessarily sound the same. And my goal wasn't to try to match the original sound because I wanted to develop my own engine with its own sound, so it has been tweaked according to my needs and according to what I consider a good FM sound. The possibility to use DX7 patches was a real bonus and most of them will sound very similar to the originals but some will need a little tweaking. Of all patches I tried I didn't find a single one which wasn't useful after importing, even without tweaking anything. And last but not least - QFM editor must be activated by holding Shift while selecting Osc/Main or Osc/Aux page. Although both set of pages (Osc/Main+Osc/Aux and QFM/1+QFM/2) cannot be active at the same time all modules are still active and you can use Main/Aux oscillators together with QFM operators without problem. QFM Mode and Algorithm selectors are located on alternate hidden Osc/Main page.

[*] Rotors (Osc/Main & Osc/Aux Pages)

Four parallel units of the same kind can be combined into a single sound source called "rotor" which can then be used as an oscillator. Just like in regular oscillator where phase is rotating through the cycle, in a rotor phase is rotating between 4 sound sources. Depending on the amount of interpolation this rotation can be smooth (crossfading between sources) or rough/sharp/harsh. Because rotor is acting like an oscillator to use it you have to change Main/Aux oscillator waveform to a rotor. Those have [1,2,3,4] after the name - for example, "Osc [1,2,3,4]" is a rotor consisting of four main oscillators while "LFO [1,2,3,4]" is a rotor of four LFOs. In addition to rotors, Main/Aux and Sine oscillators can directly pass another sound sources through themselves, like another oscillator, filter output, ring modulator, etc. This allows building complex configurations of various sound sources, including feedback from the output back to the input.

[*] Mixer & Balancer (Mixer/FM Page; Mixer #1 ... Mixer #4 Sections)

After oscillators and operators there goes a mixer. Audio routing inside Tranzistow consists of 4 mono paths combined into 2 stereo paths (lower and upper) at the end of the voice chain (before FX units). Those 4 paths are routed in parallel from oscillators through filters to the amplifier:

Osc1 => Filter1 => PostFilterMixer1 & Amplifier1
Osc2 => Filter2 => PostFilterMixer2 & Amplifier2
Osc3 => Filter3 => PostFilterMixer3 & Amplifier3
Osc4 => Filter4 => PostFilterMixer4 & Amplifier4

In this configuration it is not possible to "cross the paths" e.g. you cannot, for example, route Osc1 to Filter2 etc. To overcome this you can turn on the balancer which allows routing between paths 1 & 2 as well as between paths 3 & 4 (this is the same configuration as in my Diodow synthesizer and is used when importing Diodow patches):

Osc1+2 => Balancer1+2 => Filter1+2 => PostFilterMixer1+2 & Amplifier1+2
Osc3+4 => Balancer3+4 => Filter3+4 => PostFilterMixer3+4 & Amplifier3+4

But it is still not possible to route audio from paths 1 & 2 to paths 3 & 4 and vice versa. To allow for completely free audio routing there is a dedicated mixer with 4 paths and 4 audio inputs per each path:

Mixer1 => Filter1 => PostFilterMixer1 & Amplifier1
Mixer2 => Filter2 => PostFilterMixer2 & Amplifier2
Mixer3 => Filter3 => PostFilterMixer3 & Amplifier3
Mixer4 => Filter4 => PostFilterMixer4 & Amplifier4

Furthermore, the mixer can be combined with balancer for additional flexibility:

Mixer1+2 => Balancer1+2 => Filter1+2 => PostFilterMixer1+2 & Amplifier1+2
Mixer3+4 => Balancer3+4 => Filter3+4 => PostFilterMixer3+4 & Amplifier3+4

Everything else is variable and (almost) everything (be it audio or control signal, all MIDI controllers included) can modulate (almost) everything, up to 4x oversample rate (depending on how things are configured inside a particular patch).

[*] Vectors (Mixer/FM Page; Mixer #1 ... Mixer #4 Sections)

As part of the mixer vectors allow mixing 4 sound sources in the X-Y plane e.g. crossfading between 4 sound sources in the X-Y plane. There are 4 vectors inside a mixer - one for each mixer path.

[*] Noise (Mixer/FM Page; Noise #1 ... Noise #4 Sections)

After the mixer there are 4 noise generators which can be individually mixed and filtered (so all forms of noise can be generated, from white through pink to brown). Noise generators can be synched to all Main/Aux oscillators for variety of spectral/formant-like sounds (similar to "Synched Noise" oscillators on Clavia Nord Lead series of hardware synthesizers). Noise sync seed is freely programmable.

[*] Ring Modulator (Mixer/FM Page; RingMod #1 ... RingMod #4 Sections)

There are 4 ring modulators on-board, each of them with two inputs (both inputs accept the signal from any audio or modulation source). One input can be switched to unipolar which basically turns a ring modulation into amplitude modulation. Ring modulators are located outside the audio path so, in order to use them, mixer must be activated or they have to be routed directly to filters through side inputs.

[*] Driver (Filter/Amp Page; FilterB #1 ... FilterB #4 Sections)

The driver is a 4-way parallel waveshaper located before filters. It can be used not only to drive the input signal, but to shape and/or distort it using one of many drive curves per each individual waveshaper. Usually, there is no need to drive filter inputs over 25% (= 0.25) because various filter stages have (by default) nonlinear transistor characteristics ("Transistor" driver/feedback/shaper curves) and with higher input levels the signal starts to saturate. Of course, this is a very nice feature (and I was inspired by my Andromeda analog synthesizer while implementing this) but for cleaner signals don't push levels prior to the filter too much.

[*] Filter A Module (Filter/Amp Page; FilterA #1 ... FilterA #4 Sections)

Filter A module consists of 4 parallel State Variable Filters (SVF) of the ZDF (Zero-Delay Feedback) kind, all of them can provide a separate lowpass / highpass / bandpass outputs (and all combinations of them at the same time). Those filters are resonant, of course, and can be driven into selfoscillation. Each of them consists of two stages, 12dB and 24dB, with configurable distance/routing, separate feedback units and separate outputs (both outputs can be used simultaneously). They go up to 24kHz and although they can go down to 0Hz there really isn't much sense to go below 20-30Hz, otherwise you can experience excessive output levels and similar artifacts with some filter configurations. Many feedback curves are available to shape the feedback and they can be either symmetrical or asymmetrical in order to generate different harmonics. One of many nice SVF characteristics is that (in case of a lowpass filter) they don't attenuate frequencies below cutoff when resonance is increased - basically, they preserve bass and have a very powerful sound.

There are 5 SVF types in Tranzistow and some of them can use quite a lot of CPU. The basic filter type is similar to Waldorf Q 12dB/24dB and Alesis Andromeda 12dB, so it is my usual choice for most sounds. FilterC / FilterH have much different topology and can have much different (I would dare to say - more analog) sound compared to the basic filter. They are mostly identical with FilterC having an internal clipper while FilterH having additional soft limiter/saturator to keep feedback under control. FilterQ is a totally different topology with different resonance characteristics and it's own unique sound. FilterX is a very aggressive and screamy diode based SVF model with a really unique sound. Both FilterQ and FilterX use quite a lot of CPU to keep things under control, so two modes of operation are provided: 12dB only (FilterQ/12 and FilterX/12) and both 12/24dB (FilterQ/24 and FilterX/24). Separate 12dB/24dB configurations are provided for other filter types as well. With all those filters - the key is to combine various feedback amounts/curves with internal limiters for a broad range of filter characteristics. All filters A have a side input for direct signal routing. Side inputs don't go through mixer/balancer and aren't affected by gain settings.

Lowpass/highpass/bandpass outputs of both stages can be set independently, so various filter configurations can be achieved (for example, Yamaha CS-80 configuration with 12dB resonant highpass filter in front of a 12dB resonant lowpass, and many more). All individual A filters can be chained together - the chain is always as follows: F1=>F2=>F3=>F4=>F1 with freely configurable levels between stages, so various combinations are possible, up to the gargantuan 96dB cascaded filter.

[*] Filter B Module (Filter/Amp Page; FilterB #1 ... FilterB #4 Sections)

Filter B module consists of 4 parallel transistor based ladder filters and 2 parallel diode based ladder filters (located on paths 1 & 2 only) of the ZDF (Zero-Delay Feedback) kind, all of them resonant and driveable into selfoscillation. Transistor ladder is inspired by Oberheim Xpander filters so all poles are fully configurable and dozens of various filters types and characteristics can be easily created: lowpass, highpass, bandpass, bandstop/notch, phaseshift, 1-/2-/3-/4-pole = 6/12/18/24dB, etc, etc. Diode ladder is lowpass only with stronger resonance and a very juicy sound which reminds me on some modular filters. It can be set to 6dB/12dB/18dB/24dB where 18dB/24dB modes are "well behaving" while 6dB/12dB ones are much more agresive and loud. Both transistor and diode ladders can be used at the same time and crossfaded/combined together. They go up to 24kHz and although they can go down to 0Hz there really isn't much sense to go below 20-30Hz, otherwise you can experience excessive output levels and similar artifacts with some filter configurations. Many feedback curves are available to shape the feedback and they can be either symmetrical or asymmetrical in order to generate different harmonics.

Due to the design, ladder filters attenuate frequencies below cutoff - if such behavior is not desirable for some sounds there are compensation and normalization units inside to overcome this, as well as the ability to pass only feedback signal through the feedback unit or both input signal + feedback signal. As with SVF - the key to push ladder filters to the maximum is to combine various feedback amounts/curves together with configurable poles for a broad range of filter characteristics. All filters B have a side input for direct signal routing. Side inputs don't go through mixer/balancer and aren't affected by gain settings. B filters use quite a lot of CPU as well, especially diode ladder one.

All individual B filters can be chained together - the chain is always as follows: F1=>F2=>F3=>F4=>F1 with freely configurable levels between stages, so various combinations are possible, up to the gargantuan 96dB cascaded filter.

[*] Comb Filter Module (LFO/Comb Page; Comb #1 ... Comb #4 Sections)

Comb filters are basically feed-back or feed-forward modulated delay lines. They don't actually damp any part of the signal but add a delayed version of the input signal to the output, creating peak and holes depending on the frequency and feedback. The maximum comb delay is 1 second and the delay buffer can be shaped for more strong or a more soft sound, depending on the particular application. All comb filters have a side input for direct signal routing. Side inputs don't go through mixer/balancer and aren't affected by gain settings. Beside side inputs, comb filters have additional input/output settings, so filters A/B can be routed directly to them and/or they can be routed directly to filters A/B. Among the other things, this is great for various per-voice chorus/flanger effects etc. Comb filters also have a lot of use in physical modeling because they can create abstractions of various string and wind instruments. Beside CPU usage, comb filters are large memory consuments because there are 4 of them per voice and there are 64 or 128 voices across all parts = 256 or 512 comb filters in total, each needing a 1-second buffer of floating point values up to 4x oversample rate (= 384kHz @ 96kHz sample rate). For this reason, comb buffers are allocated on-demand, after comb filters were activated or after a patch using comb filters has been loaded (but once allocated - they remain allocated for a particular part until you restart Tranzistow).

Warning: Beware of levels - comb filters can produce extremely high output level!

All individual comb filters can be chained together - the chain is always as follows: F1=>F2=>F3=>F4=>F1 with freely configurable levels between stages, so various combinations are possible.

[*] Filter FM (Mixer/FM Page; Filter #1 FM .. Filter #4 FM Sections)

All filters (A, B, Comb) can be frequency modulated by any audio or modulation source, including filters themselves. Contrary to oscillator FM, each filter has only one FM input.

[*] Post-Filter Mixer & Amplifier (Filter/Amp Page; Amplifier #1 ... Amplifier #4 Sections)

Post-filter mixer combines output from all filters together with pre-filter input and an additional post-filter side input. The signal then goes through a simple but effective 4-way parallel compressor (aimed mainly at keeping output level under control) to the amplifier which sets the final level of each audio path, pans both lower and upper stereo paths left/right and balances them between lower and upper stereo effect groups. It is controlled by amplifier modulation source which is usually amplifier envelope (Env #4) and is also responsible for setting the output volume and DC correction of the voice output.

[*] Shaper (Filter/Amp Page; FilterB #1 ... FilterB #4 Sections)

The shaper is a 4-way parallel waveshaper located after the filters. It can be used not only to limit the output signal (because waveshapers are basically compressors/limiters in their own way), but to reshape, saturate and/or distort it using one of many saturation curves per each individual waveshaper. Shaper can be located either before post-filter mixer (so only filter output go through the shaper) or after post-filter mixer (so filters + pre-filter input & post-filter side input go through the shaper).

[*] Voice Engine (Osc/Main & Patch/FX Pages; Osc, AuxOsc, Modules, Voice & Control Sections)

Voice engine activates/executes all modules and routes audio/modulation signals between them based on parameters set inside a patch. It combines outputs from all individual voices into a single 4-mono/2-stereo audio stream and sends it to the effect processor. It is also responsible for audio input and MIDI messages handling, including unison mode where more than one voice is used per note with configurable detuning and modulation drift. Voice engine supports 3 levels of audio quality: Single Processing (no oversampling), Double Processing (2x oversampling) and Quad Processing (4x oversampling). Oversampling can be activated for audio/synthesis engine only, of for both audio/synthesis and FX engine. On the other hand, the engine supports audio reduction where audio quality can be degraded either by lowering the resolution of the signal (bit reduction down to just 1 bit) or by dividing the (over)sample rate by a given factor (down to 1/1024 for totally quantized and disharmonic audio - great for industrial sounds, for example). Reduction works on audio generators only (Main/Aux oscillators and QFM engine) meaning that FM, ring modulators, filters, waveshapers, amplifiers and other audio processors are not affected. In many cases (especially when working with 88.2/96kHz sample rates and/or oversampling) audio generators (over)sample rate can be reduced by a factor of 2 without any audible or measurable audio artifacts but with lower CPU consumption. Voice engine contains the dedicated declicker unit for envelopes and filters. Clicks can occur under various circumstances: short envelope times combined with mono mode and/or free-running oscillators, voice stealing, playing speed and style, etc. Declicker can minimize those clicks in most cases. Use it carefully (otherwise it can create clicks when there aren't any) and don't forget that not all clicks are (always) bad ;-)

I addition to audio signals, voice engine generates all modulation/control signals at the rate set inside a patch (from up to audio/oversample rate down to sample rate / 4096; 4 to 8 is the optimum for most applications) including MIDI modulation (approx. 333x per second), note-on modulation and the modulation of FX parameters (from sample rate down to sample rate / 4096). Although sources modulate destinations through various modulation matrices (and almost everything can modulate almost everything) there are 4 fixed parallel control signal lanes which are used for direct modulation and the modulation of rate/time parameters.

[*] Low Frequency Oscillators (LFO/Comb Page; LFO/AuxLFO #1 ... LFO/AuxLFO #4 Sections)

Low frequency oscillators (LFOs) are similar to audio oscillators but their purpose is to act as modulation sources, so the parameter set is completely different. Although they are designated as "low frequency" they can go well into the audio range, up to 24kHz depending on the control rate. But, keep in mind that they aren't antialiased so they can generate tons of unwanted harmonics if you decide to use them as audio sources. There are two sets with 4 LFOs in each set: main and auxiliary (8 in total). Main LFOs are always active while auxiliary ones can be activated on demand to conserve CPU, otherwise they are exactly the same. Beside using the built-in shapes, LFOs can directly pass another sound sources through themselves, just like Main/Aux oscillators but quantized at the control rate. LFOs can be synchronized to VST host tempo and all patch LFOs can be synced together as well (so all LFO modulations per chord are synchronized, great for chord sweeps). Because those "Voice LFOs" are tied together their rates cannot be individually modulated or synced to other modulation sources, although they do react to MIDI realtime messages (Start, Cont and Measure).

[*] Envelopes (Env/Main & Env/Aux Pages; Env/AuxEnv #1 ... Env/AuxEnv #4 Sections)

There are two sets with 4 envelopes in each set: main and auxiliary (8 in total). Main envelopes are always active while auxiliary ones can be activated on demand to conserve CPU, otherwise they are exactly the same. Each envelope has 10 stages (8 from note-on and 2 after note-off) with time/level parameters, configurable curves and loop points from each stage to any other stage for building very complex modulations including the simulation of complex multistage LFOs. Beside using the built-in curves, each envelope stage can directly pass another sound sources through itself, just like Main/Aux oscillators but quantized at the control rate (see "Env Sequence" patch). Envelopes can be synchronized to VST host tempo as well.

Env #4 is the "Amplifier Envelope" and it controls the overall duration of the voice. It can be routed to other modulation destinations just as all other envelopes but it controls the voice activity no matter what destination it is routed to. Amplifier modulation source is usually (and by default) set to Env #4 and, although it can be set to any other source, Env #4 still acts like amplifier envelope and controls the voice duration.

All envelopes have an internal sub-envelope path with a separate set of levels. This parallel second envelope is not smoothed and is not under the sequence control but, otherwise, it is identical to the main one and all other parameters (like times etc.) are shared between them. So, Tranzistow envelopes are actually dual-envelopes.

In addition to the regular editor, each Main/Aux envelope has a built-in graphical editor which can be activated by clicking on the appropriate envelope section. It enables easier editing because you can draw the envelope with the mouse and see how it actually looks in time. To work with a particular stage you can also press and hold keys 0..9 before and while using the mouse. This is especially useful if some or all stages are closed (after patch initialization, for example) because you cannot access them directly with the mouse in this case.

[*] Lags (Spec/Mod Page; Lag #1 ... Lag #4 Sections)

A lag unit is like the attack stage of an envelope - it can lag the sound a bit but can also be used to implement glide/portamento. There is no dedicated portamento module and Tranzistow is lacking somewhat in this regards because I am not a big glide/portamento fan and rarely use such sounds.

[*] Modulation Matrix (Matrix/16 & Matrix/32 Pages; ModMatrix #1 ... ModMatrix #32 Sections)

All modulation routing between various sources and destinations goes through one of 5 modulation matrices. This main/fast one is processed at the control rate (from up to audio/oversample rate down to sample rate / 4096) and has 32 slots grouped into 8 groups of 4 consecutive slots. Each slot has one modulation source, two separate destinations, one modulation curve/shape and various other parameters for precise control of a particular modulation routing. To conserve CPU power only groups with at least one active slot (one or both destinations assigned) are processed. So, don't leave unnecessary gaps between slots and always allocate them sequentially from the beginning. There is also the possibility to run some modulation slots at full audio/oversample rate while the others are processed at the regular control rate. This allows processing various audio rate modulations like real exponential oscillator FM etc.

[*] MIDI Modulation Matrix (MIDI/Mod Page; MIDIMatrix #1 ... MIDIMatrix #16 Sections)

This slow modulation matrix is processed at the MIDI rate (approx. 333x per second) and has 16 slots grouped into 4 groups of 4 consecutive slots. Each slot has one modulation source, two separate destinations, one modulation curve/shape and various other parameters for precise control of a particular modulation routing. To conserve CPU power only groups with at least one active slot (one or both destinations assigned) are processed. So, don't leave unnecessary gaps between slots and always allocate them sequentially from the beginning.

[*] Note-On Modulation Matrix (Note/Mod Page; NoteMatrix #1 ... NoteMatrix #16 Sections)

This modulation matrix is processed at note-on only and has 16 slots grouped into 4 groups of 4 consecutive slots. It's main purpose is to process modulations which have sense at the beginning of the note only - like velocity, keytracking etc. Each slot has one modulation source, two separate destinations, one modulation curve/shape and various other parameters for precise control of a particular modulation routing. To conserve CPU power only groups with at least one active slot (one or both destinations assigned) are processed. So, don't leave unnecessary gaps between slots and always allocate them sequentially from the beginning.

[*] Note-Off Modulation Matrix (Spec/Mod Page; OffMatrix #1 ... OffMatrix #4 Sections)

This modulation matrix is processed after the note-off MIDI message has been received and has 4 modulation slots. Like the main/fast modulation matrix, it is processed at the control rate (from up to audio/oversample rate down to sample rate / 4096) as well. Each slot has one modulation source, two separate destinations, one modulation curve/shape and various other parameters for precise control of a particular modulation routing. To conserve CPU power this matrix is processed only if at least one slot is active (one or both destinations assigned).

[*] Direct Modulation (Spec/Mod Page; Direct Modulation Section)

Beside modulation matrices, some sources (like envelopes, LFOs, etc.) can modulate six most important parameters (Main/Aux oscillators pitch, the cutoff of filters A & B and comb filter frequency) directly over four fixed control lanes. Sources on one lane can modulate destinations on the same lane only. For example, LFO #1 can directly modulate the pitch of oscillator #1 but cannot directly modulate the pitch of oscillators #2, #3 and #4 (modulation matrix has to be used for this instead).

[*] Modifiers (Spec/Mod Page; Modifier #1 ... Modifier #4 Sections)

Each of the 4 modifiers can perform various mathematical/logical operations on two input control signals and two constants. The results of those operations can then be used as modulation matrix sources. Furthermore, an modifier can be used as input to another modifier for building more complex modulations.

[*] Sync Modifiers (Spec/Mod Hidden Page; Modifier #1 ... Modifier #4 Sections)

There are 4 sync-modifiers per voice which run at audio/oversample rate and, beside additional LFO/envelope/etc. syncing (over the existing functionality built into the modulation engine), they can also be used to sync oscillators to various modules and even MIDI controllers. Contrary to the virtual sync oscillators and interoscillator syncing, sync-modifiers are not antialiased and can create all sorts of digital artifacts which can be desirable or undesirable, depending on the particular case. They don't sync at the end of the cycle as regular oscillators (because many sync-mod sources don't even have a cycle, only oscillators, LFOs, envelopes and lag processors do) but at the crossing from negative to positive values (MIDI controllers cross at the middle of their range). For this reason, you can have more than one sync points during the same cycle, when using wavetables, for example, which is not possible with the virtual sync oscillators and interoscillator syncing.

[*] Rate/Time Modulation (Spec/Mod Hidden Page; Modifier #1 ... Modifier #4 Sections)

Various rate/time parameters (like LFO rates, envelope times for all segments, chorus rates, delay times, etc.) can be modulated in realtime over four fixed control lanes, each with a separate ModFactor parameter: #1 for LFO #1 / AuxLFO #1 / Env #1 / AuxEnv #1 / Lag #1, #2 for LFO #2 / AuxLFO #2 / Env #2 / AuxEnv #2 / Lag #2, etc. Note: All envelope / lag / delay times are translated into rates internally (for example, 4s envelope time translates into 0.25Hz envelope rate).

The formula for rate modulation is: ModulatedRate = Rate * (1 + RateModulation * ModFactor)

Example 1: LFO #1 Rate=1Hz, ModFactor1=1000, Amount=1 => The resulting LFO #1 rate will be 1001Hz
Example 2: LFO #4 Rate=1Hz, ModFactor4=1, Amount=-0.9 => The resulting LFO #4 rate will be 0.1Hz

Example 3: Env #1 Time=0.5s (Env #1 Rate=2Hz), ModFactor1=1, Amount=-0.75 => The resulting Env #1 time will be 2s (0.5Hz rate)

[*] Contour Generators (Osc/Main Page; Contours Button/Section)

In addition to predefined waves/wavetables, LFO shapes, envelope/lag curves, filter/drive/shaper and other curves/shapes, there are 100 freely drawable contour generators (with built-in waveform/spectral graphical editor) per patch, each generator with up to 8192 points (although contours with over 1024 points don't have much practical sense). This provides a whole new level of versatility because they can be used as additional Main/Aux oscillator waveforms, LFO shapes, envelope curves, filter curves, modulation curves, etc. If used as main oscillator waveforms they can be bandlimited (only contours with 4 points and up) or non-bandlimited, with freely configurable min/max window parameters (the same as built-in waveforms). All contour generators have various resynthesis options and capabilities. Among the other things, WAV files can be loaded, resynthesized and used as the source material. Unfortunately, I ran out of time while preparing Tranzistow for production and, except "Contour 00" example patch, couldn't create more patches exploring this extremely powerful capability.

[*] FX Engine (Patch/FX Page)

Tranzistow contains a full-fledged effects engine with a dedicated chorus/flanger, phaser, delay and reverb FX modules. Except in case of chorus/flanger, all other modules contain two parallel stereo FX units which operate on lower and upper stereo audio paths, and both of them can be arranged in parallel or serial and everything in between. Phaser/Delay/Reverb units have two sets of lowpass/highpass filters (one at the input, one at the output) and can be connected in arbitrary order. FX engine can work at the sample rate or at 2x or 4x oversampling rate. The order of modules inside the FX chain is fully configurable.

[*] Chorus (Chorus/FX = Shift+Spec/Mod Page; Left/Right Chorus #1 ... Left/Right Chorus #4)

Stereo chorus/flanger FX unit contains 4 delay lines and 8 LFOs per channel - especially great for strings and pads but perfectly usable for many other sounds out there. By default, it is located at the end of the voice chain, on lower stereo path only and before Phaser/Delay/Reverb effect units, but can be positioned anywhere inside the FX chain if desired. The times for all delay lines and rates for all LFOs are freely configurable, together with LFO shapes and many other parameters. Left and right delays 1 & 2 as well as left and right delays 3 & 4 can be arranged into a so-called "matrix" configuration for more smooth and spacious sound.

[*] Phaser (Patch/FX Page; Phaser #1 & #2 Sections)

The phaser is a combination of several all-pass filters working in parallel. This generates a strongly colorized signal with a "spacey" and "swooshy" character, depending on center, spacing and feedback parameters. Up to 16 stages can be activated for both lower and upper units. Each phaser unit contains an dedicated LFO with configurable shape, rate and phase (both rate and phase separate for left and right channels). Phaser LFOs are syncable to VST host tempo and can also modulate the times of delays #1 and #2 enabling them to act as additional chorus/flanger units if needed. Those LFOs can even modulate reverb times for a very special reverb sound. Setting wet amount for both phasers #1 & #2 to zero will turn the whole phaser module off to conserve CPU.

[*] Delay (Patch/FX Page; Delay #1 & #2 Sections)

The delay is an effect which produces echoes of the input signal and is the most usable effect of all effects (at least to me). Those echoes can be routed back and added to the input through an feedback unit for much denser delays. Both parallel stereo delay units can be individually synced to host tempo, crossed between and arranged into a so-called "matrix" configuration for more smooth and spacious sound (a "poor man's" reverb without using the dedicated reverb unit). Setting wet amount for both delays #1 & #2 to zero will turn the whole delay module off to conserve CPU.

[*] Reverb (Patch/FX Page; Reverb #1 & #2 Sections; Reverb/FX = Shift+PatchFX Page)

The reverb effect adds ambience to a dry audio signal making it more spacious and 3-dimensional. Both parallel stereo reverb units contain and internal early reflection sub-unit with up to 16 taps, internal feedback-delay sub-unit with up to 16 taps and internal all-pass filter sub-unit with up to 16 taps. There are tons of various parameters to configure all of them for any particular application. Groups of 4 successive feedback-delay sub-units can be arranged into a so-called "matrix" configuration for more smooth and spacious sound.
There is also an additional auxiliary stereo reverb embedded into reverb #1. Setting wet amount for both reverbs #1 & #2 to zero will turn the whole reverb module off to conserve CPU.

Ed Ten Eyck (www.edtaudio.com) on the Tranzistow reverb:

"The Reverb sounds very smooth and spacious to me. At first the Endelverb preset reminded me of the Eventide Blackhole reverb because of it's smoothness. It would be excellent on it's own as an effect plugin!"

[*] FX Modulation Matrix (Spec/Mod Page; FXMatrix #1 ... FXMatrix #4 Sections)

This modulation matrix is processed at FX control rate (from sample rate down to sample rate / 4096) and has 4 modulation slots for modulating various FX parameters. Each slot has one modulation source, two separate destinations, one modulation curve/shape and various other parameters for precise control of a particular modulation routing. Only MIDI sources have sense and are available here. To conserve CPU power this matrix is processed only if at least one slot is active (one or both destinations assigned).

[*] MIDI Implementation

Tranzistow has a full MIDI implementation and all MIDI control messages (including control change messages, channel aftertouch, poly aftertouch and pitch bend messages) as well as note-on and note-off (release) velocity can be used as sources in all modulation matrices. There is no awkward "MIDI learning" and similar VST idiosyncrasies - MIDI works in the same way as used to work on hardware synthesizers for ages.

[*] Multitimbral Operation & Multithreading

Except in demo version, Tranzistow is always operating in multitimbtral mode with 4 or 8 multimode parts, depending on the configuration. Parts 5 to 8 can be accessed by clicking on Part #1 ... Part #4 buttons while holding Ctrl key. By default, it is operating in singlethreading mode meaning that all parts are executing on the same CPU core. But it can also operate in multithreading mode where each of the 4 parts (or 8 parts in 4 main+sub pairs) can be executed on a separate CPU core utilizing up to 4 CPU cores without any additional multithreading support from VST host. This feature must be activated in Tranzistow.ini (this file should be located in the Current User directory or, if not found there, in the same directory where Tranzistow32/64 DLLs are located):

[Engine]
Multithreading=0 | 1 | 2 | -1 | -2 ; Default: 0
...

(*) Multithreading=0 ... All parts are running inside a single thread (4 parts total).

(*) Multithreading=1 ... Each part runs inside a separate thread + there is an additional thread for mixing all parts (4 parts total).

(*) Multithreading=2 ... Each part runs inside a separate thread and thread for the first part is used for mixing (4 parts total).

(*) Multithreading=-1 ... Each main+sub part pair runs inside a separate thread + there is an additional thread for mixing all parts (8 parts total = 4 main + 4 sub).

(*) Multithreading=-2 ... Each main+sub part pair runs inside a separate thread and thread for the first part pair is used for mixing (8 parts total = 4 main + 4 sub). This is what I mostly use when working with Tranzistow.

And last but not least, Tranzistow multimode allows sub-parts inside a single multi part/patch. The basic concept behind this is: Tranzistow voices are very complex and you don't really need all modules for a single voice in most cases. So, it would be very nice if some modules from the same voice could respond on different MIDI channels. As described before, there are 4 mono paths per voice and each mono path can be assigned to a separate MIDI channel (assignable relative to the part MIDI channel) giving you the possibility to have different sounds from the same patch. Of course, this is not the same as full 16-part multi-mode but was much easier to implement into the existing engine and doesn't really consume any additional resources. This enhancement won't extend the available polyphony because paths assigned to different MIDI channels aren't disabled while playing (internal engine paralelism doesn't allow this), they are still executing but are muted after post-filter mixer. All this opens a whole new world of interactions between various paths/parts/modules, something not possible with any other synthesizer out there. The same as with most other Tranzistow features, for that matter ;-)

[*] Arpeggiator (Alt+Patch/FX Page)

Each multimode part has it's own independent arpeggiator. It is always synced to host tempo, responds to MIDI Start/Stop/Continue realtime messages and has the usual set of features: Mode, Clock, Gate, Range, Notes, Pattern (16 fixed rhythm patterns + 1 user-definable), Reset and Sort.

[*] User Interface

Tranzistow user interface is, in general, designed and optimized for touch-screens, and works great with touch-screens, so it has just a few basic elements: sections, buttons, lists and sliders. To access various parameters through lists (LFO shapes, modulation sources/destinations, etc.) just click on the parameter and a list will pop up. All sliders have an edit box above them where you can enter the value directly. Double-click on the edit box will show the calculator (result is returned back to the edit box). All sliders also have a vertical mode where you can move a mouse vertically outside the slider area and use the whole display height for a single slider. Right-click on the parameter will reset it to a default value. Double-click on the parameter name (or right-click on the parameter slider/list while holding Ctrl key) is "undo" - it will reset a particular parameter to the previous value (undo buffers are unlimited). Press and hold Ctrl key while moving a slider to activate the "microscope" mode for precise parameter adjustment.

Many parameters have extended ranges e.g. they allow values outside the regular slider range. If a value falls outside the slider range then a slider will be colored red. You can enter such values directly into the edit box or you can press and hold Alt key and move a mouse outside the slider area. In case of "clocked" parameters (LFO rates and phaser/delay times with Clocked parameter activated) numbers represent notes e.g. 0.0625 = 1/16 note, 0.125 = 1/8 note, 0.25 = 1/4 note, 0.5 = 1/2 note, 1 = whole note, etc. To simplify entering of such parameters you can enter a number and press / (divide) key. For example, 4 followed by / will give you 0.25.

Tranzistow has a huge number of parameters (almost 4800 per patch) and there simply isn't enough space on main pages for all of them. That's why there are hidden parameters on hidden pages (accessible by double-clicking on the page button) and alternate hidden pages (accessible by clicking on the page button while holding Alt key). Furthermore, a few shifted pages exists as mentioned in QFM, Chorus and Reverb sections.

In addition to the on-screen GUI elements, a few keyboard shortcuts are available as well: F2 = Save, F3 = Load and F9 = Init. Those shortcuts must be enabled in Tranzistow.ini:

```
[Editor]
Shortcuts=1
...
```

Shortcuts could or could not work, depending on how is your VST host using the above keys.

User interface us fully scalable so it can be accommodated to large displays which are more and more common today. Scaling can be enabled in Tranzistow.ini:

```
[Editor]
Scale=150
...
```

Scale parameter is in percents and the default is 100 (meaning the original 100% scale). The user interface is optimized for 1280x720, 1280x768 and 1280x800 resolutions so the alignment of various elements can be slightly different with scales other than 100. The lowest possible scale is 10. If you set Scale=0 then user interface will be scaled automatically according to the current screen resolution.

[*] Patches & Presets

Tranzistow doesn't use VST preset/patch system at all. I simply don't like it and it doesn't work in a way I am used to work. Each Tranzistow patch is a separate file which can be loaded from disk, saved to disk, organized into directories and manipulated like any other file. The only parameters saved by the host are patch file locations, MIDI channels and the polyphony for all parts. For easier browsing through patches there is a separate browser page accessible by clicking on the "Browse" button located on the "Patch/FX" page. On the same page there are 3 more patch buttons: "Init" to initialize the patch to default settings, "Load" to load the patch from the file and "Save" to save the patch to the file. Holding Alt key while clicking on the "Save" button will act as "Save As" e.g. the save dialog will pop up, so you can set the file name and directory where you want the patch to be saved.

[*] Randomizer

Tranzistow has a full-blown randomizer with a parallel user interface (activated with Alt+Init) where you can set min/max ranges for all parameters you want to randomize. It mainly generates various industrial drones/noises and that's what I like about it and what I am using it for ;-)

The patch can be randomized from randomizer editor or directly from the regular editor using Ctrl+Init combination (or Ctrl+F9 if shortcuts are enabled). There are two types of randomization: absolute and relative (parameters are randomized relative to the current values). The type and intensity is configurable through "Randomizator" parameter: positive = absolute, negative = relative. All randomization parameters are saved together with the patch so you can recall all settings later. If you want to load a patch without loading randomization parameters then press and hold Alt key while loading. Randomization parameters from the current patch will be preserved in this case = easy way to copy randomization parameters from one patch to another.

[*] VST Automation

By default, Tranzistow doesn't support VST automation because I don't like it, I don't use it and I prefer to control everything over MIDI. But you can turn it on using Tranzistow.ini file:

```
[Editor]
Automation=1
...
```

I don't plan to provide any additional automation features beside this.

[*] External Control Interface

External Control Interface (ECI) is the module which enables the direct communication between Tranzistow user interface and external/hardware MIDI controllers. Tranzistow has a built-in native support for Behringer BCR2000 which makes creating/editing patches a breeze. Everything you can do using the standard Tranzistow screen editor can be done through ECI and BCR2000. The only thing you have to do is to transmit the appropriate system exclusive file I created for Behringer BCR2000 (this must be done only once). Various other nice features have been implemented as well, like knob anti-jittering, curves and 4 levels of knob precision for fine granularity of parameters etc.

External Control Interface must be enabled in Tranzistow.ini:

```
[Editor]
ExternalControl=1
...
```

Detailed ECI description is part of a separate document.

[*] Hardware

A few notes about PC configuration:

(*) Intel SpeedStep (EIST) should be disabled in BIOS.

(*) Windows power plan should be set to "Max. performance" with minimum and maximum processor state set to 100%.

(*) It is recommended to disable hyperthreading in BIOS.

[*] CPU Usage Tips

(*) Turn off modules you don't use.

(*) Use auxiliary oscillators instead of main oscillators if only saw/pulse/triangle waves are needed.

(*) Use sine oscillators if only a sine is needed. Four sine oscillators (or 8 if both Main/Aux oscillator modules are set to Sine) are a rudimentary additive engine which use very little CPU.

(*) Use FM because you can get very complex sounds with moderate CPU usage.

(*) Use main oscillators without crossfading and/or interpolation between individual waves where possible.

(*) Once you turn them on - use all oscillators, filters, etc. All 4 units inside the same module are working in parallel, so you won't gain any advantage in using just one, two or three of them.

(*) Use SuperSaw oscillators instead of unison and use all 4 SuperSaw oscillators per module. It is much more effective to have 4 SuperSaw oscillators with 8 sub-oscillators each than 2 SuperSaw oscillators with 16 sub-oscillators and other two oscillators unused.

(*) Remove redundant modulations and modulations with source/destination set to "None" or amount set to zero.

(*) Don't leave unnecessary gaps between modulation slots and always allocate them sequentially from the beginning.

(*) Don't put slow modulation sources in fast modulation matrix. MIDI controllers should be put into a MIDI modulation matrix while velocity, keytrack and similar sources should be put into a note-on modulation matrix.

(*) Turn off oversampling because in most cases (especially when working with 88.2/96kHz sample rates) oversampling is not really needed.

(*) Use reductor because in many cases (especially when working with 88.2/96kHz sample rates and/or oversampling) audio generators (over)sample rate can be reduced by a factor of 2 without any audible or measurable audio artifacts but with lower CPU consumption.

(*) Use delay instead of reverb if full ambience is not really needed.

[*] Hosts Quirks

Some quirks noticed on various VST hosts:

(*) Reaper: Check "Disable saving of full plug-in state", otherwise it will save a patch to the file on every single parameter change! Such automatic saving is totally unnecessary with Tranzistow because it will ask you to save a patch on exit if some modifications have been made and the patch hasn't already been saved. When I change a parameter on a hardware synthesizer the patch doesn't get saved immediately - it's only when I press "Store", "Save" or whatever button that a patch will be saved. I don't see a single reason that a software should behave differently. Sometimes I really don't get those VST hosts stupidities.

[*] Practical Tips from Yuli Yolo

(*) Contours :

Alt + mouse draw will snap to zero.

Set size to 1024 for import a single cycle from Serum, Icarus or other audio editor (2048 samples), very useful for custom Osc waveforms and LFO/Env Shape contours.

(*) Portamento:

Set Lag to Glide, Level = -1, Offset = +1 and in the modulation matrix assign Lag to Pitch, Amount = +1.

(*) Mono/Legato:

Set Declicker to Voice On/Release.

Set Env #4 to Legato and Env #1/2/3 as well, if needed.

Turn on Declick for filters.

Set polyphony to 1

(*) Pitch Bend:

1 Octave +/- => Set Amount to +1, Min to -0.5 and Max to 999999.

2 semitones +/- => Set Amount to +0.12, Min to -0.5 and Max to 999999.

(*) LFOs:

You can use various modulation curves and LFO Shapes.

For example, set Analog Saw LFO to modulate Filter Cutoff and in the modulation matrix assign Exp5 curve.

(*) Envelopes:

You can loop an envelope (so it will act like an LFO) and modulate the time of each segment or assign differents contours to differents stages.

(*) Oscillators and LFOs:

For free running phase set StartPhase to -1.

OscRange is very useful when a source like envelopes and LFOs modulate the Pitch.

(*) VST Hosts:

If you use Vienna Ensemble Pro don't forget to decouple all to avoid "autosave presets". The automation system works perfectly.

[*] Additive+FM/GPU Engine

Additive+FM engine is a separate engine running on a graphics card (GPU) in parallel with the main Tranzistow engine. It contains 4 multi parts with 16 additive voices (for 64 voices in total), each voice with 4 additive oscillators, each oscillator with 240 dual-harmonics and internal virtual sync oscillator. Dual-harmonic is a combination of two harmonics where the second one is PW phase-shifted in order to have pulse-width at additive level (so, basically, there are 480 harmonics per oscillator * 256 oscillators = 122880 harmonics processed per sample). Each dual-harmonic has 8-stage loopable envelope (61440 envelopes processed per sample). All oscillators are fully bandlimited which means that you can, for example, sweep a saw wave and the engine will automatically remove harmonics over Nyquist as you pitch up and add harmonics as you pitch down. To avoid sudden jumps when removing and adding harmonics (which can be quite hearable in some cases) there is a built-in XFade functionality which crossfades between old and new harmonic window.

More or less, this is additive Tranzistow engine ported to GPU with a difference that Tranzistow works with up to 1024 dual-harmonics and it cannot calculate each harmonic in realtime because there isn't enough CPU power to do this together with everything else Tranzistow does at the same time. So, I am precalculating a lot of things in advance on Tranzistow and store calculated data in memory - this process is invisible to the user but needs a lot of RAM. Today's machines have enough RAM, anyway, so in many cases it is better to use more RAM when possible in order to save some CPU for other things. The other limitation is that Tranzistow harmonics don't have envelopes - you cannot control how individual harmonics change over time, which is quite a big difference between those two engines.

Each additive oscillator has it's own 136-band spectral filter (with 8Hz-20kHz range) meaning that 256 filters are running at the same time processing data for a total of 256 * 136 = 34816 bands per sample. This filter can be used for everything, from 136-band EQ via various resonant/non-resonant lowpasss, bandpass, highpass, formant (multiple bandpass) to some completely crazy and yet unheard filters. Furthermore, additive engine has many advanced FM capabilities. Four successive harmonics are grouped into one FM operator, so each additive oscillator (4 per voice) supports up to 60 operators - this is 240 operators per voice. Each operator supports two separate FM inputs with feedback capability, and with 4 harmonics (each with separate Index, Multiplier and Level parameters) per operator much more complex waveforms are possible. Operators can be connected in unlimited number of ways and can be freely combined with individual harmonics (for example, you can use harmonics 1..224 for additive synthesis and harmonics 225..240 as 4-operator FM) and harmonics can even be used for additive and FM duties at the same time.

Additive oscillators are sub-modules inside auxiliary oscillators - they share common pitch/frequency parameters together with the same output and they can be used as audio/modulation sources in tandem with the regular oscillators + additive output can be further processed through Tranzistow engine as any other audio source inside the synthesizer.

The complete 64-voice Additive+FM engine (4 parts, 16 voices per part, 4 oscillators per voice = 64 voices / 256 oscillators in total across 4 parts, each oscillator with 240 dual-harmonics, 60 FM operators and 136-band spectral filter, each dual-harmonic with 8-stage loopable envelope, etc.) is optimized to the maximum and, running at 96kHz sample rate with 240-samples VST buffer, consumes under 55% of the available power on my 1.2GHz HD7970 graphics card with fan speed set to minimum to keep fan noise very low.

Additive oscillators have various resynthesis options and capabilities. Among the other things, WAV files can be loaded, resynthesized and used as the source material. All those features greatly expand the available sound palette and simplify the initial sound generation. Such complex and powerful Additive+FM engine has never been invented/implemented before, as far as I know :-)

Note: Although it is finished and thoroughly tested, Additive+FM/GPU engine is not yet available to the public. Owners of registered Tranzistow version could contact me with GPU details and I will send them a test program to check if the engine runs on their systems.

[*] 32/64-bit Linux Standalone and Experimental Linux Native VST 2.x Versions

Starting with build 16.08.2016/1, Tranzistow is available on Linux in the form of 32-bit and 64-bit standalone audio applications as well as experimental Linux native VST 2.x libraries. Due to various reasons, experimental Linux native VST versions are implemented as client/server combo where the client is a 32/64-bit VST .so library and the server is a separate 32/64-bit standalone Tranzistow synthesizer application.

In order to use Tranzistow you need 32-bit or 64-bit Linux with at least GTK2 version 2.8 installed (GTK3+ is not supported yet). I recommend 64-bit Ubuntu Studio distribution as the best one currently available, IMHO. There is no Tranzistow installation per se - just unpack the ZIP archive into some folder and that's it. Tranzistow use ALSA, Jack or PortAudio/PortMIDI to communicate with audio and MIDI hardware, so you must have correct driver and firmware installed for the sound card you intend to use. You need libasound2 as well, although I suspect it is already installed together with ALSA. By default, Tranzistow will connect to "hw:0,0" audio device and "hw:0,0" MIDI device, but this can be changed through configuration/INI files. If you intend to use Jack then you must have Jack installed and configured properly, of course. The same with PortAudio/PortMIDI.

Tranzistow will search for Tranzistow.ini configuration file in /home/<UserName>/.config/Tranzistow folder (replace <UserName> with your Linux user name). If it doesn't find one there, it will search in a folder where Tranzistow application has been copied and run from. You can open Tranzistow.ini using a regular text editor and configure the following sections/options:

[Audio]
Driver=ALSA | Jack | PortAudio ; Audio driver to use (default: ALSA)
Device=... ; ALSA audio device to use with Tranzistow (ALSA only, default: hw:0,0)
SampleRate=... ; Sample rate to use (ALSA only, default: 44100)
BufferTime=... ; Size of cyclic audio buffer in milliseconds (ALSA only, default: 20)
FrameTime=... ; Size of one audio frame in milliseconds (ALSA only, default: 10)
Resample=0 | 1 ; Turn automatic ALSA resampling on/off (ALSA only, default: 0)
Format=16 | 24 | 32 ; Set the format / number of bits for audio data (ALSA only, default: 16)
Oversample=0 | 1 ; Turn internal Tranzistow 2x oversampling on/off (default: 0)

[MIDI]
Device=... ; ALSA MIDI device to use with Tranzistow (ALSA only, default: hw:0,0)
Tempo=... ; Fixed tempo in BPM or 0 for automatic synchronization to incoming MIDI clock (default: 125)
TempoSmooth= ... ; Smooths out tempo changes when synchronizing to incoming MIDI clock (default: 0)

[Jack]
ClientName=... ; The name of Jack client (default: Tranzistow)
ServerName=... ; The name of Jack server (default: None)

[PortAudio]
AudioLibrary=... ; The name of PortAudio library (default: libportaudio.so)
AudioDevice=... ; The name of audio output device (default: None = default device)
InputDevice=... ; The name of audio input device (default: None = output device)
AudioBufferSize=... ; Size of audio buffer (default: 512)
MidiLibrary=... ; The name of PortMIDI library (default: libportmidi.so)
MidiDevice=... ; The name of MIDI input device (default: None = default device)
MidiBufferSize=... ; Size of MIDI buffer (default: 256)

Note: PortAudio/PortMIDI device names are not the same as ALSA device names!

Other configuration options are the same as described in previous sections.

To avoid unnecessary ALSA resampling, it is much better to match the requested sample rate to be the same as driver-supported one. Additional internal 2x oversampling can be turned on to achieve 88.2/96kHz quality on systems where higher sample rates are not available. Of course, CPU usage will be doubled in this case.

Configuration example:

```
[Audio]
Device=hw:1,0
SampleRate=48000
BufferTime=10
FrameTime=5
Oversample=1

[MIDI]
Device=hw:2,0,0
Tempo=0
TempoSmooth=0.5

[Editor]
Scale=125
```

Note #1: Upper/lower cases are important in all file names and paths because I am not really a fan of lowercase-only names. If an operating system has been designed with case sensitivity in mind then both upper and lower case should be used in my opinion.

Note #2: Configuration file should be writeable because Tranzistow will use it to store various data like window position, patch names/paths and MIDI channels for all parts, etc.

Linux version will sound, look and work more-or-less exactly like the Windows one, with the following exceptions:

(*) No parameter automation in either standalone and experimental native VST versions.
(*) Standalone, ALSA only: Only one (main) stereo audio output is currently available.
(*) Standalone, ALSA only: No audio inputs are available yet.
(*) Demo version only: Sample rate is limited to 48kHz (resampling should be turned on for higher sample rates).

Everything else is the same and almost everything written in this document applies to Linux version as well. All patches are interchangeable between versions too. BTW, despite the equal look and functionality on Linux and Windows, Tranzistow is a native Linux GTK2 application which doesn't use Wine or WineLib at all.