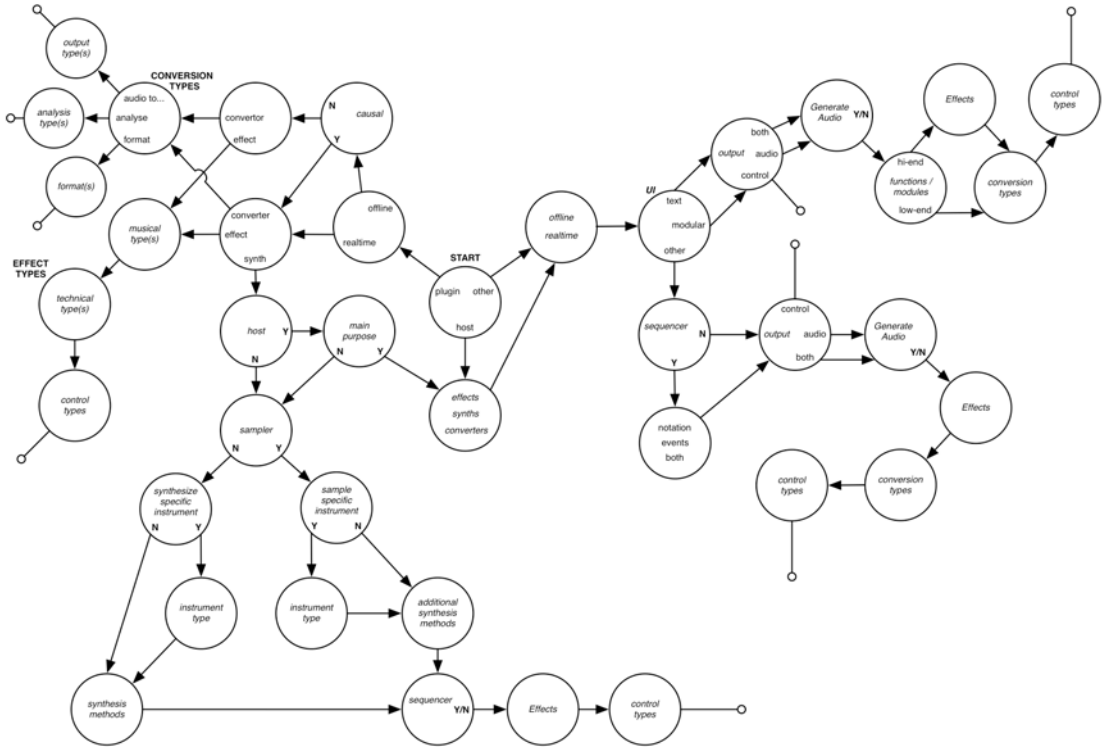
APPROACHING DIGITAL AUDIO

a technical audio software classification



Toine Heuvelmans

MA Sound & Music Technology

2005 – 2006

EUROPEAN MEDIA MASTER OF ARTS

Approaching Digital Audio
a technical audio software classification

© 2006 Toine Heuvelmans
www.toine.org
toine@toine.org

Graduation Thesis

Associated Project:
OFFline, a different audio editor

Graduation Teacher:
Jorrit Tamminga

European Media Master of Arts
Sound & Music Technology

Utrecht School of Music & Technology
Utrecht School of the Arts
www.hku.nl

CONTENTS

INTRODUCTION

As in many fields, the invention of the personal computer had a major impact on the music industry. Software emerged which enabled composers and producers to sequence and mix audio like never heard before, and generate and modify audio in ways they couldn't dream of. Music production was made more accessible to the common man, and from everywhere in the world audio software entered the market. Some applications tried to incorporate an entire studio, others were more aimed at just one thing.

Nowadays there's an incredible amount of audio software, covering almost every possible topic of audio processing. For a developer it has become quite hard - but I believe not impossible - to come up with something new, not in the last place because he can't get a clear overview of what has already been done before. Main reason for this is probably that most classifications for software are aimed to satisfy the user, not the developer. These classifications are quite general, and sometimes even incorrect.

A user tends to search for a certain piece of software by using terms that describe what an application does with a sound audibly. Best results however for finding out if something is not yet done, is to categorize by what an application does with a sound technically. For instance, it might be that there's not yet any equalizer that uses the Fourier Transform to get result, but it might just sound better (...?) than the commonly used techniques.

It might sometimes be hard to classify a certain piece of software technically, either because it is too varied or complex, or because the developer won't share his techniques. However, one can make a very detailed classification, without getting vague on terminology. Keep in mind that a classification on its own can't tell you what has been done. It must be combined with a database to get an overview off all existing software. With this, a developer can get much more certain if a technique or combination has already been used before. Also, the uniqueness of an application comes more to light: it might for instance be an audio editor, but unlike other audio editors, this one edits sounds only in the frequency domain.

I am such a developer, and I try to find out if my current project (OFFline) is a gap in the market. Therefore I created a technical classification in the form of a deterministic system, which I will explain in this writing.

1. COMMON CLASSIFICATIONS

Before explaining the technical classification of audio software, let's take a look at a few classifications that already exist. Some elements of these classifications are based on principles that I also use in the technical classification, or can add to it to improve its quality.

1.1 Purpose

The most common classification describes the general purpose of an audio application. This purpose depends on what an application can do with digital audio. That can be one or more of the following:

Audio...	The application can...
Generation	... produce its own audio material
Modification	... modify either self-generated or incoming audio
Conversion	... convert audio to a different storage type or representation
Recording	... capture external audio
Playback	... play a recorded sound file
Editing	... change the start- and stop time and duration of a sound file
Sequencing	... order audio events in time

Of course you won't find an application that says it's audio-generating-modifying-recording-and-playing. A few general groups are invented having different combinations of these events.

Effect

This kind of application often only modifies audio, and with that it is the simplest kind of audio software (not meaning the effect algorithm can't be complex). It is almost always found as a plug-in, letting a host application handle all I/O. Effects are the most common piece of audio software, because of their simplicity.

Effect: Tape Delay

Synthesizer

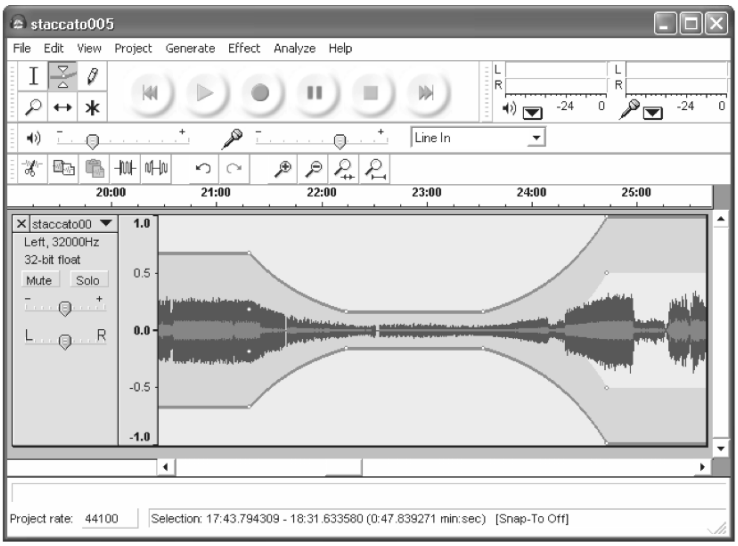
Like an effect, a synthesizer is specialized in only one thing: generating audio. Because it generates its own audio, it doesn't need any input. Therefore it can be found as a plug-in as well as a standalone application.

Sampler

A sampler can load (or sometimes record) sound files and play them back. Often it is found among synthesizers, because both produce audio.

Audio Editor

This is a standalone application that can do various things with sound files. It can record new files, modify and edit them, convert them, and of course play them back.



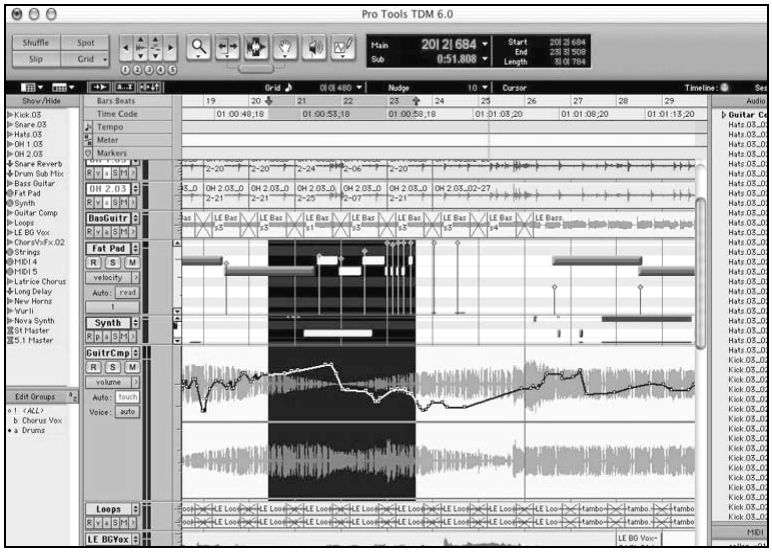
Audio Editor: Audacity

Sequencer

Originally a *sequencer* only handles the recording, sequencing, editing and playback of *control* data. Often it is confused with a DAW (see below) that does the same for *audio* data.

DAW

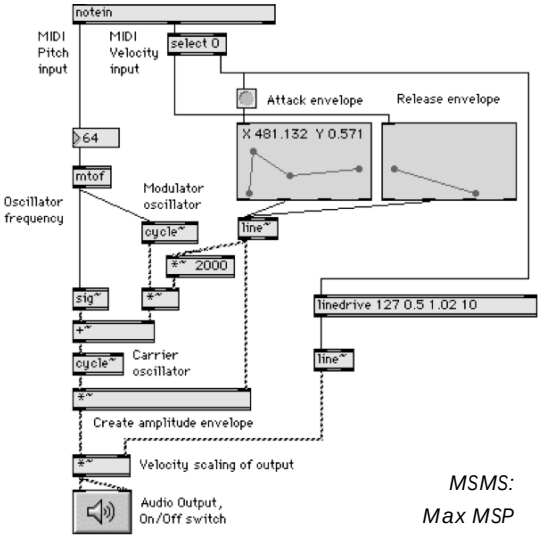
A Digital Audio Workstation (DAW) is the most advanced piece of audio software, being capable of generating and modifying audio (often using effect- and synthesizer plug-ins), and recording, sequencing, editing, conversion and playback of audio files. It is often used as the core of a modern studio.



DAW: Pro Tools

Modular Software Music Studio

These are applications in which every event (generating, recording, modifying, converting audio) is represented as a module. Modules can be connected to each other in various ways (often with virtual wires), making a process chain. The modules often do their calculations real-time and are easy to reconnect, and therefore it is often used in interactive installations and live performances.



MSMS: Max MSP

Audio Programming Language

These are applications in which one has to write a script that generates or modifies sound. It often has the ability to record sound, and in many cases it can load or save a sound file.

Audio Analysis

Analysis of audio almost always means the audio is converted to a visual representation, for instance to see which frequencies are present. The audio can be either a loaded file, or recorded audio. The audio data actually remains the same.

This classification is not a standard for what a certain application must be capable of and what not. A developer may choose to develop a Modular Software Music Studio that processes sound files offline, and perhaps is capable of sequencing the files. Or an effect might combine its incoming audio with self-generated audio.

1.2 Plug-ins

Effects and synthesizers can - as described above - often be found as a plug-in. This means the input and output (both audio and control data) are handled by a host application. Audio editors, DAW's and Modular Studio's are most suited to host plug-ins, using it for generation and modification of sound files and audio streams.

Plug-ins can be found in all kinds of types, being synthesizers and effects, but also analysis plug-ins displaying for instance a RMS meter, or more advanced plug-ins that can mix channels or that are specialized in panning.

A few standard formats exist for audio plug-ins, some having more options than others (like the channel mixing). The most popular standards are VST (*Virtual Studio Technology*), AU (*Audio Unit*), RTAS (*Real Time Audio Suite*) and DX

(*DirectX*). Applications can also have their own plug-in formats (RTAS is only for Pro Tools). Not every format is supported on every operating system, and therefore the plug-in format is often an important criterion when users search through audio software.

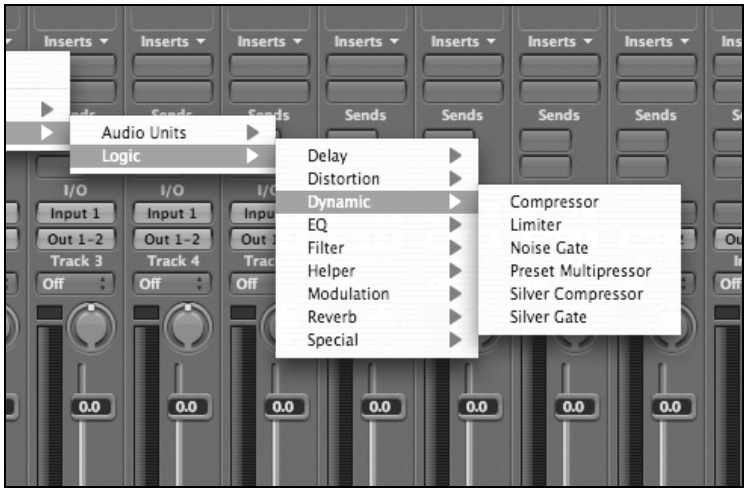
Effects

If effects are categorized, than it's almost always done on what kind of effect the effect will generate. Many effects are specifically designed to solve a certain problem or to supply a certain demand in sound editing. Other effects live up to their name, and create special effects; sometimes so extreme the original sound can't be distinguished anymore.

Here is a list of the most common classes of effects (not official names):

Filter / EQ	Suppress or boost certain frequencies
Reverb	Room simulation / spatial effects
Delay	Echo / time delay effects
Dynamics	Compress / limit audio
Distortion	Deformation of audio
Modulation	Slowly changing delay / filter / ...

These do not cover every effect, and some plug-ins combine multiple effects. More specific categories exist, but many effects covered by these categories often also fall in one of the above.



Effect classification in Logic Pro

Synthesizers and Samplers

Synthesizer plug-ins are categorized in various ways, sometimes by synthesis technique (FM, granular, physical modelling...), sometimes by instrument it tries to simulate (strings, brass, organs, analog synthesizer simulations...). Samplers are distinguished by the sample format(s) they support (Akai, Giga, SDII...).

2. TECHNICAL CLASSIFICATIONS

For a developer a classification like the above described purposes isn't very useful, because one doesn't know if an application is for example 100% an audio editor, or perhaps 10% modular studio. It doesn't tell the developer without doubt what the application can and can't do with audio. More useful would be when the possibilities of an application in terms of generation, modification etc. were specified.

It is useful to see if there are for instance already many reverbs if a developer wants to create one himself. However it might be he invented or adapted a technique that makes his reverb far more efficient than existing ones. One can't tell this difference only from the generally used classification. It also doesn't matter if this reverb is a VST or an Audio Unit; the algorithm is the same.

More important for a developer is to know what techniques and approaches are used to generate and modify sound, what control is offered, and in what form this all is presented. Combined with elements of the above described classifications the developer can get a clear overview of how a certain application works, if it has been done in other ways, and what kind of software is technically related.

I molded this together into a deterministic system, that can for instance be implemented in a website to create a database of audio software. The choices I made are described in this chapter; later on I will describe how these are placed in the system.

2.1 Form

I start off separating software looking at what software form it has, because this form has great consequences for the further properties an application can have. The most common kind of audio software (strength in numbers) is a plug-in, as described above. As the name indicates, a plug-in is part of a bigger application, a host. A plug-in nearly always has only one function, being sound generation, modification, conversion or recording. This function extends the capabilities of the host application: without a host a plug-in is useless. The host supports all the audio in- and output, and handles control signals et-cetera. The way a host offers its audio- and control data to a plug-in is often (but not always) standardized, making it possible to use a single plug-in in different hosts.

An application can of course be not extendable, not supporting plug-ins. We can therefore split audio software in three groups: plug-ins, hosts and other applications. Sometimes it happens that a plug-in can also host plug-ins. Depending on the hosting being its main purpose, we group it either with plug-ins or with hosts.

2.2 Time

A second factor that gravely influences the possibilities of an audio application is how it handles time. There are two ways in which audio software can process sound over time. The way the software does its processing, depends on the way the sound is provided.

Real-time

In a real-time context, the computer must respond to live voice or music "as fast as required". In audio technology, this implies there should be no (perceptible) delay at the receiving side. The term *real-time* can only be used for software that processes live sound, which cannot be predicted. If an application processes sound files whilst playing, it can't be called real-time.

Offline

Audio software that processes sound offline does the calculations as fast as possible. The time it takes to calculate may be longer or shorter than the original length of the sound. In order to do this, the audio that is being processed must first be stored.

The advantage of offline processing compared to real-time processing is that the entire sound is known by the application. This means the application is capable of using all this information in a calculation. If a system uses future information next to present and / or past information, the system is called non-causal. A simple example of a non-causal computation is normalization, where the maximum value of a sound file is required. Real-time systems can only handle present and past information, and can therefore only be causal.

2.3 Control

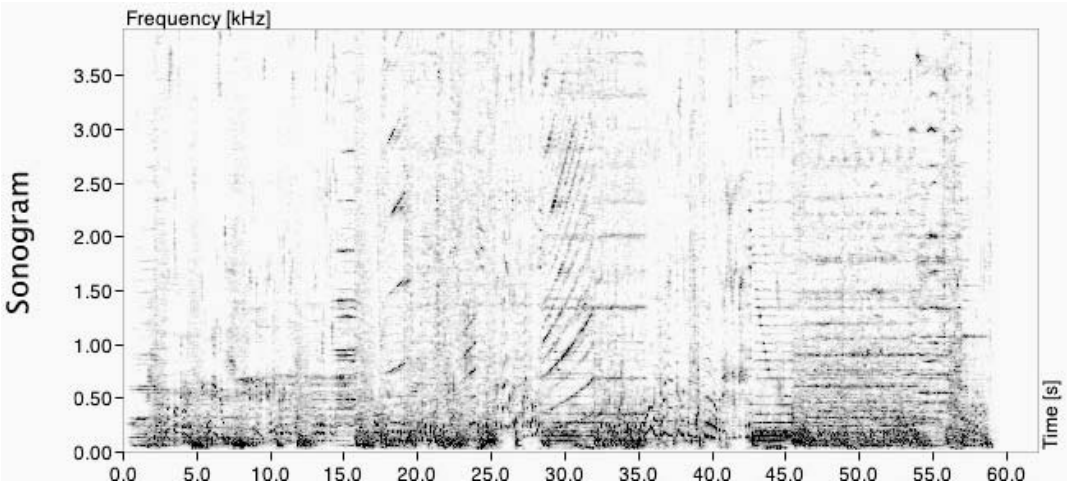
Audio software produces audio, but can also produce control signals, for instance to control an external hardware synthesizer. Many applications can also receive these control signals, to control a software synthesizer, or for instance to change parameters of an audio effect.

These control signals are of lower frequency than audio, and can also contain note information for a synthesizer, or other information. There are a few popular

standards for this kind of control signal, of which the two most famous at this moment are MIDI and OpenSound Control.
Applications can however use their own kind of control signal that is only known internally. An example might be an audio editor with a custom type of effects, of which the parameters are being controlled by automation tables.
Sometimes audio software uses an analysis of an (extra) audio stream to control a parameter. Think of a side-chain bus, of which the envelope of the incoming audio is often used to control the amount of compression. Beside side-chain, there are also things like network audio, a network connection that passes audio between two or more computers.

2.4 Conversion

Beside generating and / or modifying sound, audio software can also convert sound. A conversion does not affect the audible character of a sound, either because it only analyses the sound, or because it only changes the way the sound is stored.
In the first case, the sound file is used to create a different representation. This can be a visual representation for measuring and analysis of the sound (spectrogram, VU meter, phase plot), or a different kind of signal (control signal), for instance a pitch tracking of a sound, resulting in a signal containing note information.
In the second case, the digital format of a sound is changed. This can be a sample rate change, or more technical, a change in how the bits are ordered. If this is a radical change, the audible character of a sound might also change (a sound at a sample rate of 11025 Hz sounds quite bad compared to the more general 44100 Hz).
A change in channels (mixer) or in panning, where the audible character is also maintained, is however regarded as a modification, because the audio signal values are changed in these processes.



Sonogram in AudioXplorer, displaying frequencies with their amplitudes over time

2.5 User Interface

The user interface can also have quite an influence on the possibilities of audio software. For instance, sequencing using a modular studio application is hard to imagine. I distinguish four kinds of different user interfaces in audio software:

Sound files

Applications that display sound files or pieces of audio that you can select, move, cut or play back. This can be DAW's (sequencing part), audio editors, samplers and analysis software.

Modules

The modular software music studio has a totally different way of working than all other software. Each module can be connected and reconnected to other modules, and between them audio and control data is passed. This approach is less suited for editing and sequencing of sound files, because there's no timeline on which the audio can be placed.

Script

Scripting audio and audio events is not very common, but there are applications that use this approach, like CSound and SuperCollider. They give an immense control over how the sound will be created, because the user must specify exactly how it is constructed (which requires a certain level of know-how). It is possible to sequence audio files with a script language, but it is a bit more elaborate having to provide start time and duration for each file.

Other

Other applications often use combinations of knobs, sliders, keys, matrices, scores etc. to give the user control over what he does with audio. Sometimes even more interesting user interfaces are found, like editable frequency curves in SPEAR.

Combinations of these interfaces are common, for instance in SuperCollider, a script application with which the user can also script his graphical user interface with knobs and sliders around his script.

2.6 Effects

As with common classifications, effect- and synthesizer software requires more detailed specification, describing what techniques are used. We already have a classification for effects that describes what purpose the effect has. Combining this with a specification on what kind of technique is applied, the developer can get to know quicker if his idea has been used already. Some of these techniques

match the general classification described in chapter 1, because the category is simply named after the generally used technique. I distinguish 11 categories in which an effect can be placed very easily. The categories describe exactly what should be in it.

Filters

Filters modify the frequency of a signal by rejecting, retaining or emphasizing certain partials. Filter functions are functions in the time domain. This makes it different from the general class of filters (ch. 1), which can also hold filters that use spectral techniques.

Delays

Delays simulate echo-like acoustical phenomena by changing time information of a signal.

Modulators & Demodulators

These modify or vary parameters (amplitude, frequency or phase) of a signal by using another signal. This second signal can be low frequency (control) or high frequency (audio).

Nonlinear Processing

This is clearly a different name than those in chapter 1. It stands for every effect that creates intentional or unintentional harmonic or inharmonic frequency components that are not present in the input signal. Many effects from the distortion category use nonlinear processing.

Spatial Effects

Spatial Effects involve artificial spatial cues for he brain to place sound events in space. This actually means things like room simulation, what reverb effects use.

Time-Segment Processing

These effects process sound segments to achieve effects like time stretching and pitch shifting, or time shuffling and granulation. The first two change playback speed (of either the individual segments or the overall timeline), the last two change the order and organization of the segments.

Time-Frequency Processing

As mentioned with filters, there are effects that use spectral techniques. This means they use a time-frequency representation of the signal to transform the sound. These effects require three steps: analysis, transformation and re-synthesis.

Source-Filter Processing

This might sound bit like time-frequency processing. The effect divides a signal into a source signal (exciter) and a spectral envelope (resonator or time-varying filter), transforms these individually, and joins them again in the re-synthesis. It can also be that the envelope of one sound is joined with the source signal of another sound.

Spectral Processing

A sound is analyzed to obtain alternative frequency representations (being not time-frequency) which can than be transformed to produce the new sound.

Warping

With warping the time- and / or frequency axis is deformed to obtain interesting audio effects.

Bitstream Signal Processing

This generates a single bit representation of audio signals to be processed (not often used).

Advanced effects often use a combination of these approaches to get complex sound effects. Therefore one must not see this as a classification, but more as a specification.

2.7 Instruments

When a software synthesizer is designed to generate the sound of an existing acoustic instrument, it is often categorized in groups you would find in a symphony orchestra, like woodwind, brass, strings and percussion. If it is a different instrument, like a guitar or piano, it is just called after that. I believe when adding a bit of technical background of the instrument the categorization can be far more effective. In a good musical instruments encyclopedia one can find the scientific classification of musical instruments (the *Hornbostel-Sachs system*), based on the principle of sound-generation, which includes five groups:

Aerophones	Sound is generated by an air-column
Idiophones	Sound is generated by the material itself
Membranophones	Sound is generated by a stretched membrane (skin or other)
Chordophones	Sound is generated by a stretched string
Electrophones	Sound is generated by electronic means

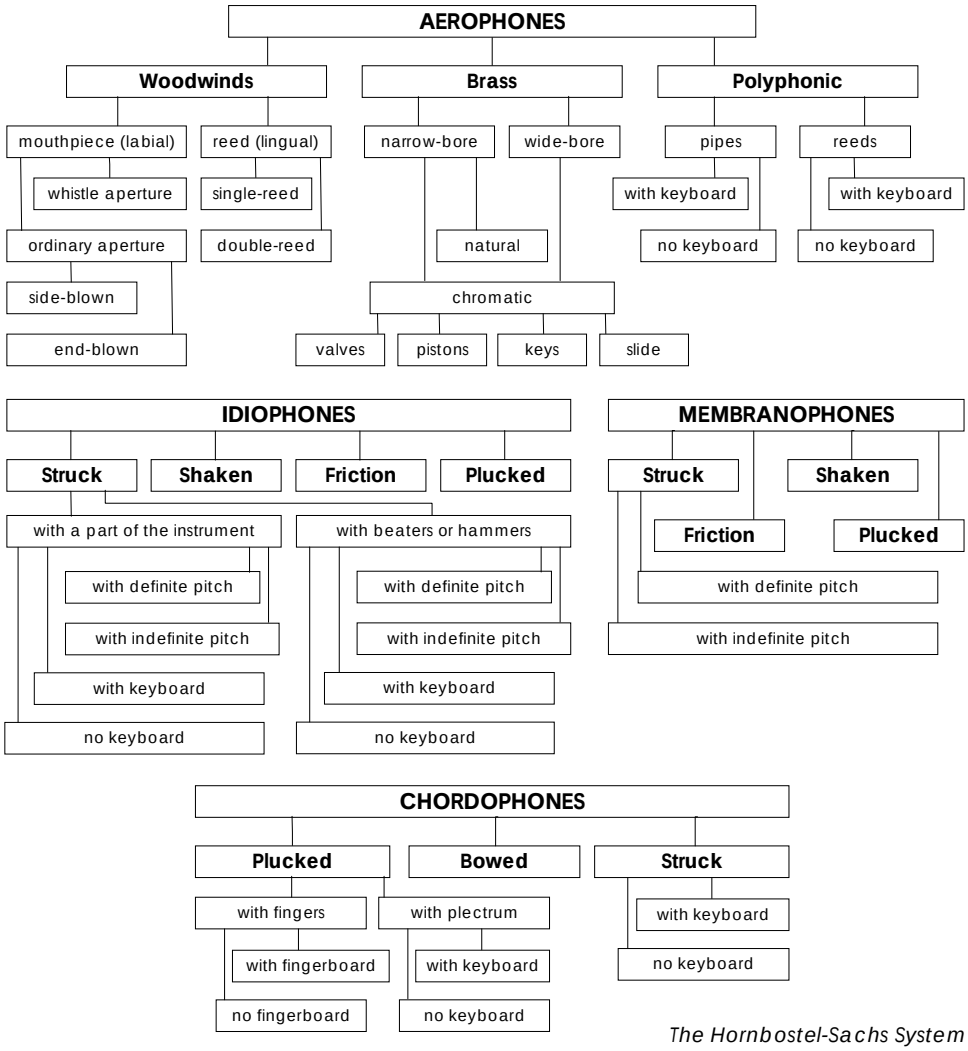
This almost matches the above-described symphony orchestra, where aerophones are woodwind and brass, chordophones are strings, and idiophones and membranophones are united under percussion. Electrophones are often

called synthesizers themselves, so software versions of them will often have exactly the same name.

These groups describe almost exactly how the sound is generated, and this makes it easy for a developer to see which instruments might be synthesized with almost the same technique. He will however need just a little more detail to find this out. Materials and the way these are set in vibration give the necessary information to shape a realistic synthesis model. The Hornbostel-Sachs system provides this level of information (and goes even more in detail).

Aerophones can either enclose the vibrating air or not (respectively called *wind instruments* and *free aerophones*). One can also separate these in woodwinds, brass and polyphonic instruments. Differences like mouthpiece or reed and narrow- or wide-bore can provide more detail.

Both idiophones and membranophones can be struck, shaken, rubbed (friction) or plucked. Chordophones can be plucked, bowed or struck. If struck, the item with which the material is struck (beaters, hammers, a part of the instrument) can gravely influence the sound that is generated. This also counts for plucked instruments, which can be plucked with fingers or with a plectrum.



The Hornbostel-Sachs System

2.8 Synthesis Methods

There are numerous methods a developer can use to synthesize a sound. Each synthesis technique has its own character, suitable for a specific kind of sound. Of course one can use different techniques to approach the same sound, but some techniques are faster or more detailed.
Some of these techniques date back a few decennia, and were originally used in analog synthesizers. So if a software synthesizer tries to simulate its analog counterpart, it can often use exactly the same techniques. However analog circuits have characteristics almost irreproducible in digital systems, so the emphasis lies on "tries to simulate".

Additive Synthesis
This technique adds up a number of waveforms of different pitch, each a partial or harmonic of the final sound. Fourier discovered the concept of a sound existing out of multiple sine waves almost 200 years ago. Sounds with not much partials, like that of an organ, can easily be simulated with this technique.

Subtractive Synthesis
Starting with a sound source, like noise, a waveform or any other sound, the final sound is reached by filtering out various frequencies. Where additive synthesis adds partials, subtractive synthesis removes partials. The filter working of our mouth is a good example of subtractive synthesis.

Frequency Modulation Synthesis (FM)
The frequency of a waveform is modulated by a second waveform. With this technique it is easier to create many harmonics. FM produces a characteristic "metallic" sound, suited for sound effects, but less useable for approximating acoustic instruments.

Physical Modelling Synthesis
This flexible technique tries to simulate the physical working and materials of the instrument, and the user's interaction with it. This produces a sound that behaves more "natural" than with other techniques. With minor changes a single physical model can produce a wide variety of sounds. (The model can be designed to synthesize a specific subgroup of the Hornbostel-Sachs system)

These are probably the most popular techniques, but there are a number of other techniques. Not all are as flexible as the above techniques, but can produce great sounds. Wavetable synthesis, granular synthesis and phase distortion synthesis are also quite common, to name a few.

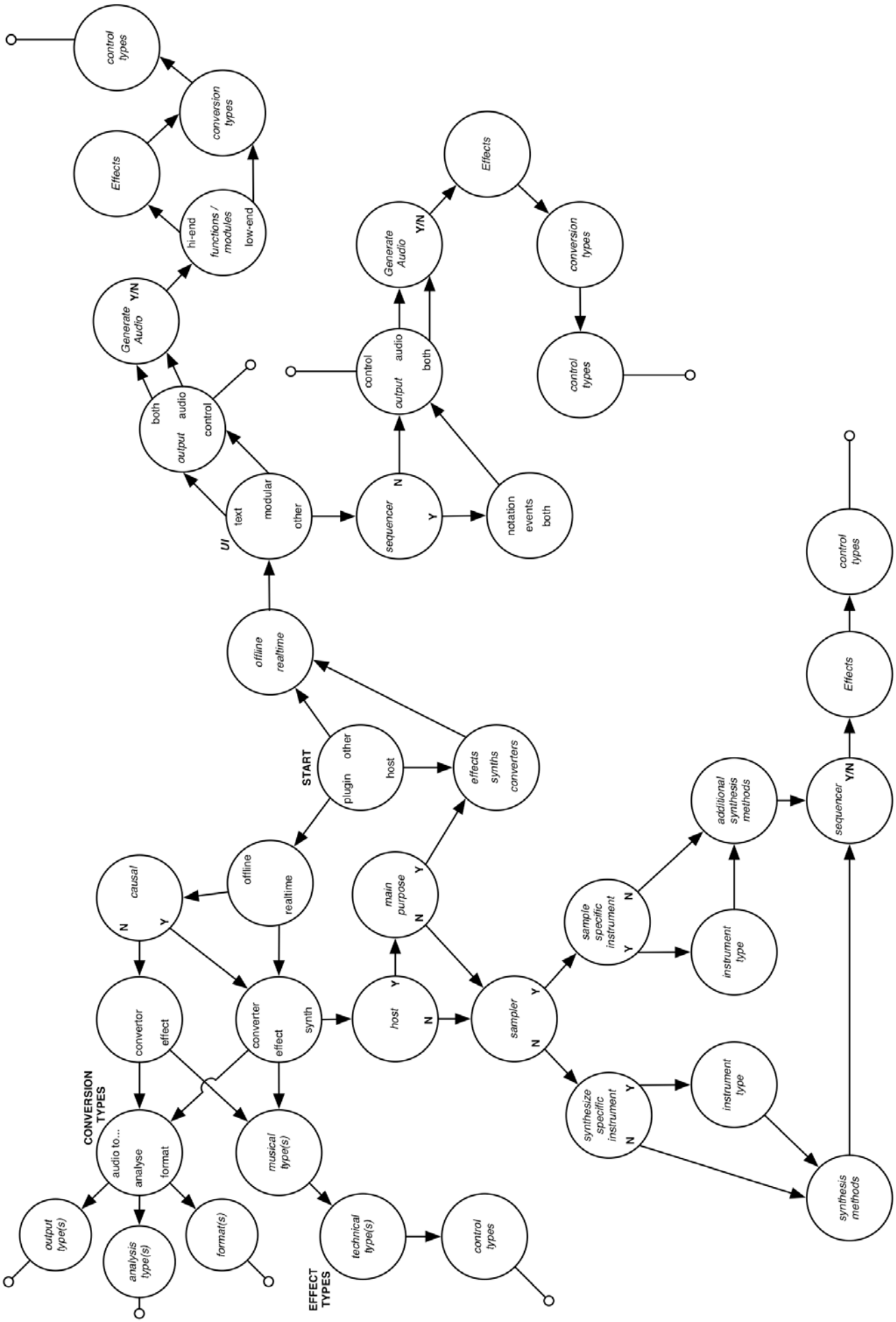
3. DETERMINISTIC SYSTEM

Just like the Hornbostel-Sachs system, all above described classifications can be combined with some extra selections into a deterministic system. Starting at a central point (or at the top in the Hornbostel-Sachs system) each decision leads to a different point in the system, where another selection must be made. At the end of the ride, a series of numbers is produced, describing all choices that are made.

The deterministic system produced from my technical classifications can be seen at the next page. It starts at the center with the selection of form (ch. 2.1). In modules like this one, with multiple outlets, only one thing can be selected (Yes or No, converter or effect or synthesizer...). With *italic* typed modules multiple items can be selected, for instance with effects, where first the musical type(s) must be selected (ch. 1.3) and than the technical types (ch. 2.6). Less surveyable areas like synthesis techniques, where the system might not have included a specific technique, will have an "other..." option where the technique can be described.

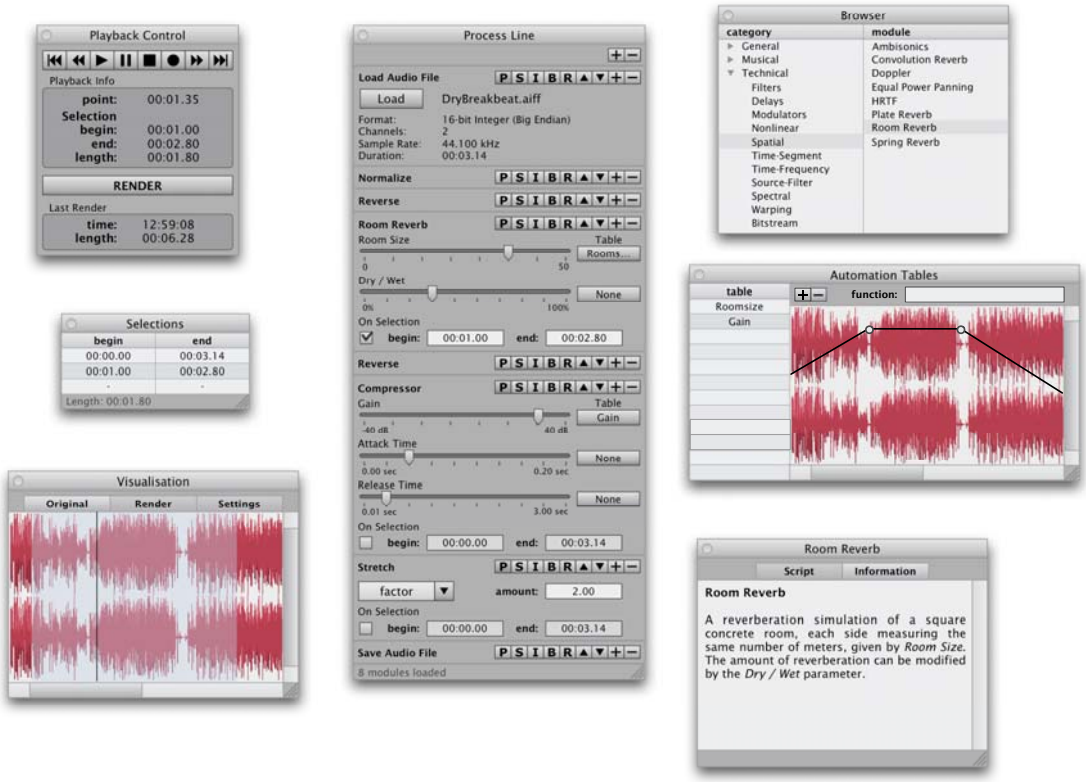
This system can be implanted in a website (like KVR) or application, displaying the right selections after each answer. The numerical series of each application that has been put in the system is stored in a database, and from that an image can be produced displaying each section of audio software, making visible what kind is very common, and what less common.

A possible visualization of the audio software database


START
UI
CONVERSION TYPES
EFFECT TYPES
offline
realtime
plugin
other
host
effects
synths
converters
text
modular
other
output
both
audio
control
Generate Audio
Y/N
hi-end
functions / modules
low-end
Effects
conversion types
control types
sequencer
N
Y
notation
events
both
causal
convertor
effect
converter
effect
synth
main purpose
host
sampler
sample specific instrument
synthesize specific instrument
instrument type
additional synthesis methods
synthesis methods
audio to...
analyse
format
output type(s)
analysis type(s)
format(s)
musical type(s)
technical type(s)

4. OFFLINE

My graduation project is called OFFline. It is an audio application that – as the name implies – works totally offline. Everything that happens with a sound file (load, cut and paste, effects, save...) is displayed as a module on a stack. Selections can be made in a window with different kinds of visualizations. Automation tables make it possible to modulate parameters. The user can write own modules using a script language. OFFline is designed to be able to run on most operating systems, like Mac OSX, Windows and Linux.



I try to find out if and which applications come close to the way OFFline works. Therefore I first need to know what kind of application OFFline is, so let's see what the common classifications and my deterministic system say.

Purpose

OFFline can do various things with sound files, but cannot sequence them. It is not a DAW, but probably an audio editor. I described every operation as being a module, but the main purpose of Modular studio's isn't working with sound files. The script language with which the user can write effects is not the main interface

in the application, so OFFline can't also be called an audio programming language (the script language on its own can).

Plug-ins

OFFline does not (yet) support popular plug-in formats because these are hard to incorporate in the system. It is nonetheless possible to script (and externally save) new modules to extend the abilities of the application, so the modules in OFFline can be seen as plug-ins.

Effects can be almost anything the user invents, ranging from simple filters or delays to hi-end modules like octave stackers or vocoders.

Synthesizers can be found as signal generators. These can also range from very simple sine or noise generators to a VOSIM system or multi-channel samplers.

Summarized, it is a cross-platform audio editor that can host own plug-ins, it has all kinds of effects, and it can generate audio. Beside price and file formats this is about it with what you can usually query a database like KVR.

Now we know a little bit about what kind of application it is, and we can find out what applications might be the same. I selected the following applications by querying the KVR online database just on audio editors. It also returned some applications that were clearly DAW's, but these applications are most likely audio editors:

Name	Platform	Host	Generate
Acoustica *(Acon)*		DX	No
Amadeus II *(HairerSoft)*		VST / AU	Yes
Audacity *(Audacity)*		LADSPA / (VST)	No
DSP-Quattro *(i3)*		VST / AU	No
GoldWave *(GoldWave)*		DX	Yes
Peak *(BIAS)*		VST / AU	No
QuickAudio *(Sion Software)*		VST	No
Sound Forge *(Sony)*		VST / DX	Yes
Sound Studio *(Freeverse)*		AU	Yes
Soundtrack Pro *(Apple)*		AU	Yes
Wave Editor *(Audiofile Engineering)*		AU	Yes
WaveLab *(Steinberg)*		VST / DX	Yes
OFFline		Own	Yes

The only other cross-platform audio editor is Audacity. It does not have signal generators, but it does host plug-in standards, where OFFline only has its own format. The database does not give enough information to deduce what

applications are like OFFline. It does give some extra information for each application, but probably the developer is free to write here whatever he wants, which makes it inconsequent information. After reading all this information (and information on each application's website) one can however get a better overview of which applications are more alike, but this is a tedious job. It did reveal some information useful for me, like the fact that Audacity, like OFFline, has the ability to script effects, GoldWave has somewhat the same for generators and Soundtrack Pro, Sound Forge and WaveLab can script tasks in the environment. Furthermore, the amount of non-causal effects in general is quite low, where in OFFline these can be present in unlimited amounts.
To be sure I also queried the KVR database on modular environments, but this did not return any application that could edit sound files, only real-time applications.

Now for the deterministic system: OFFline is a host that supports own effect and synthesizer plug-ins. These plug-ins work offline, and the rest of the application is also offline. Its user interface is mainly module based; sound file visualization is only used for selections that can also be typed in (so you can do without it). OFFline can generate audio, and its modules are hi-end. The possible effects are unlimited. Conversion can be in the form of visualizations, audio to control signal and format change. The automation tables are the only form of control the application has.

Immediately visible when using this system is that all of the above selected audio editors go a different way than OFFline. Where in OFFline the main interface is module based, all other applications rely on a sound file display as central point. Apple's Soundtrack Pro and Audiofile Engineering's Wave Editor do have a list where all applied effects can be seen and changed, but this does not have the most important role in the user interface.
The applications KVR returned on modular environments, which follow somewhat the same path as OFFline using the deterministic system, are all real-time, and none can be seen as a sound editor. This probably makes OFFline the only modular sound editor.

The deterministic system is more detailed for small software like plug-ins, where one can truly focus on the techniques used. For stand-alone applications like OFFline, one can't go as deep as that, because the application is too varied. Luckily there aren't as much stand-alone applications as plug-ins, so these can be overseen more easily.

5. CONCLUSION

Common audio software classifications are often not well specified and the limits are not clearly given. They try to categorize software by purpose, to let the user quickly find what he needs. Software may however have multiple purposes, or be written for a whole new purpose. Furthermore, these classifications include other topics that enable the user to select only the type of software he can run, like what operating system it runs on, what format it is, or what formats it supports.
For a software developer who wants to find out if a specific kind of software has already been made, these classifications often don't give enough information. This can be because they aren't specific enough, or because the given information is useless for the developer.
The developer has more benefit from a technical classification, giving more detail on how an application works, and what abilities it has. This classification can however best be combined with a few common classifications where possible, to get the most information out of it. The smaller the application, the better can be described what it does, because it often does only one thing, using just one or a few techniques.
Such a technical classification can best – like all classifications – be connected to a database that keeps track of all audio software that's gone through the system. This system is a deterministic system, where choices are made that describe what the application can or has. With the data from the database a visualization can be created that displays what kinds of audio software are very common, and where there's still a gap in the market.

By using the technical classification I created, I found out that my graduation project (OFFline) can be called a modular sound editor, and with that it is one of a kind, and thus a gap in the market. No other sound editor I found from an existing database (KVR, that says it is the biggest in its kind) was modular. Further details on techniques used can't always be specified for a big application like a sound editor, but the bigger the application, the fewer there often are. A developer can therefore often narrow a search result to a number of applications that can be overseen.

SOURCES

KVR Audio
Online Audio Software Database and wiki
http://www.kvraudio.com

Answers.com
Definitions and translation
http://www.answers.com

Websites as graphs
Visualizing html tags using traer physics' *random arboretum*
http://www.aharef.info/2006/05/websites_as_graphs.htm

Bibliography

UDO ZÖLZER (2002), ***DAFX – Digital Audio Effects***, 1st edition, John Wiley & Sons

ALAN V. OPPENHEIM, ALAN S. WILLSKY, IAN T. YOUNG (1996), ***Signals and Systems***, 2nd edition, Prentice Hall

Wikipedia

Audio editor
Audio editors (cat.)
Audio programming language
DAW
MIDI
Modular software music studio
Music sequencer
Open Sound Control
Real-time computing
Software sampler
Software synthesizer