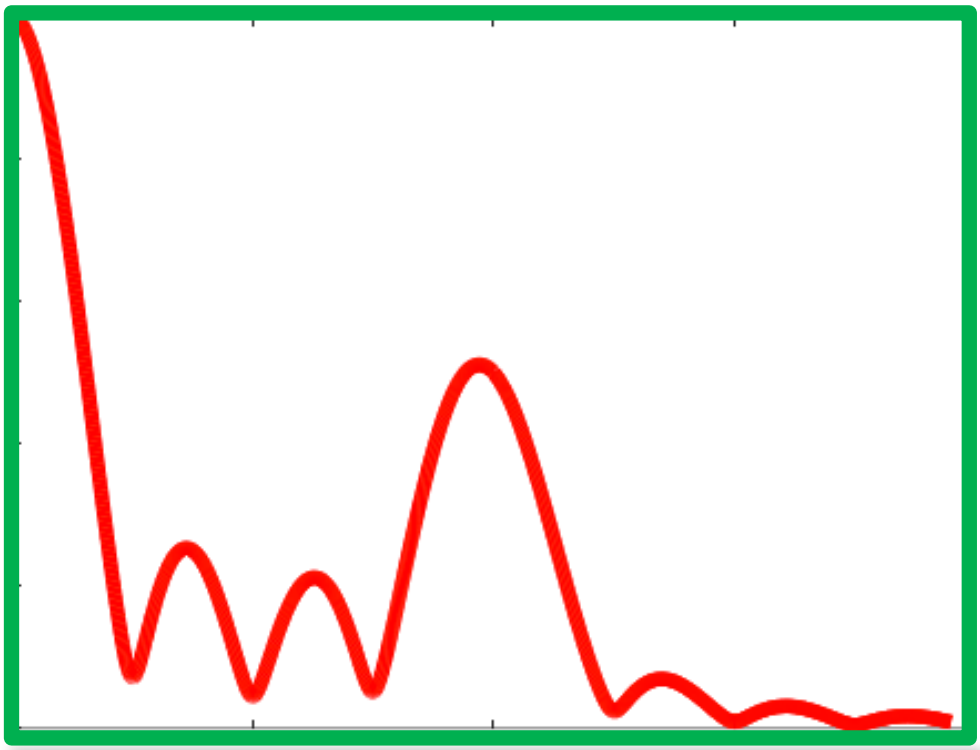
Vosim 66 Handbook

Vosim Sound Synthesis System

for artistic sound design



Heinerich Kaegi March 2021

Content

1. How it began

In the sixties of the last century Werner Kaegi - at that time a young, promising Swiss composer of contemporary music - started integrating tape and electronic music into his work. Mention that there were no computers and the first tape recorders were enormous, expensive machines. To understand the new field of sounds better, Kaegi began to experiment with whatever technical equipment he could achieve: Revox tape recorders, oscilloscopes, simple wave generators, and analogue spectral analyzers. It brought him to some very profound conclusions, he published for the first time in 1967 in his famous book: "Was ist elektronische Musik?".

In short, he stated two Ideas. First: musical sounds are always rooted in the sounds of our languages. And secondly: most of the complicated information in sound signals can be cut away, without loss of its typical sound qualities. This discovery was certainly not obvious and brought him some years later to the Vosim model and the MIDIM software.

The Vosim model describes a specific signal function, the Vosim signal, it's spectral characteristics and the parameters to control it. The Vosim function consists in its simplest form of a sinesquare pulse, followed by silence. The silence represents the part of the signal which is in fact redundant. Of course, it doesn't hold for all kinds of signals, but it does certainly for vowels. And as music is basically built on this speech like sounds, as Kaegi stated, many musical sounds can be created by much simpler signal functions than we would think, watching their shapes.

Very soon Kaegi was aware of the scope of his curious discovery. He received several grants to investigate his ideas more profoundly (Swiss and Dutch national research funds). It finally led to fifteen years of investigations at the Institute of Sonology in Utrecht, Holland (1971-1986), where he became a leading member of the scientific staff in 1973. There he had two DEC PDP-15 digital computers at his disposal and a team of very dedicated collaborators, as Stan Tempelaars, a physicist and Jo Scherpenisse, a digital technician.

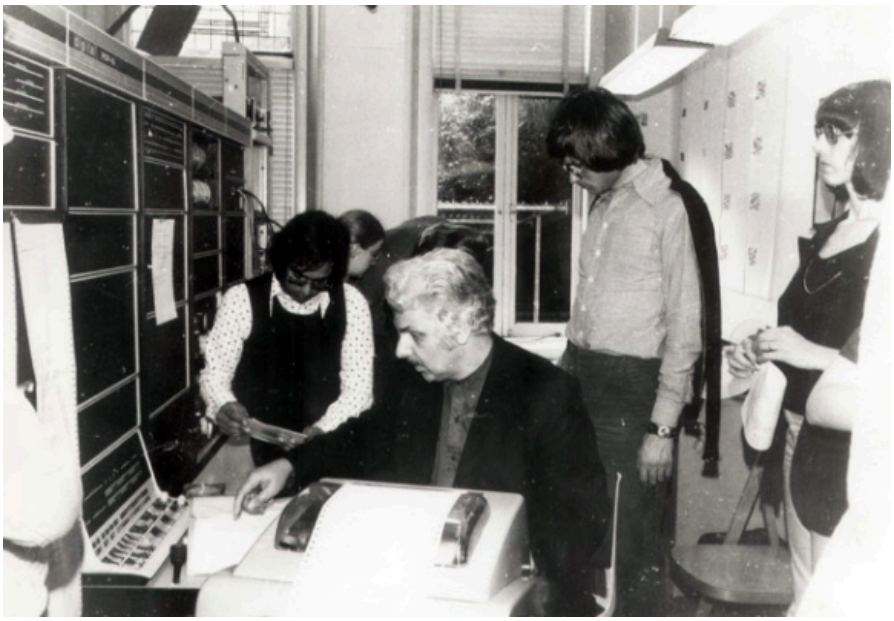


Figure 1 Werner Kaegi with some students at the DEC PDP-15 computer (ca. 1976)

Together they built the first set of microprocessor controlled Vosim generators, developed software to control the signal, and set up scientific measurements to study the perception of the sounds. Kaegi lectured many years about his growing insights into the connections between logic, mathematics, music theory, history, speech and sound synthesis. Composers and researchers from all over the world came to Holland to study at the famous Institute of Sonology. Kaegi published many of his discoveries and was invited to conferences about linguistics and new music research in Europe, USA and Canada.

But finally, Kaegi's interests went back to his roots, music as art. His goal was achieved: to make new sound worlds accessible to the musician by modern technical means. In these years

he created several electronic pieces, all carefully crafted in his own MIDIM language[1]. Other composers followed his path and many MIDIM/VOSIM compositions arose. In 1981 the MIDIM-group was founded, uniting a group of the closest artists, and survived the dramatic end of the famous Institute in Utrecht (1986)[2]. The MIDIM-group[3] held the heritage alive till begin nineties. Kaegi's MIDIM-system (written in Fortran 77 code) was transferred to Atari ST 1040 computers. New pieces emerged and concerts were organized with composers, artists, writers and performers.

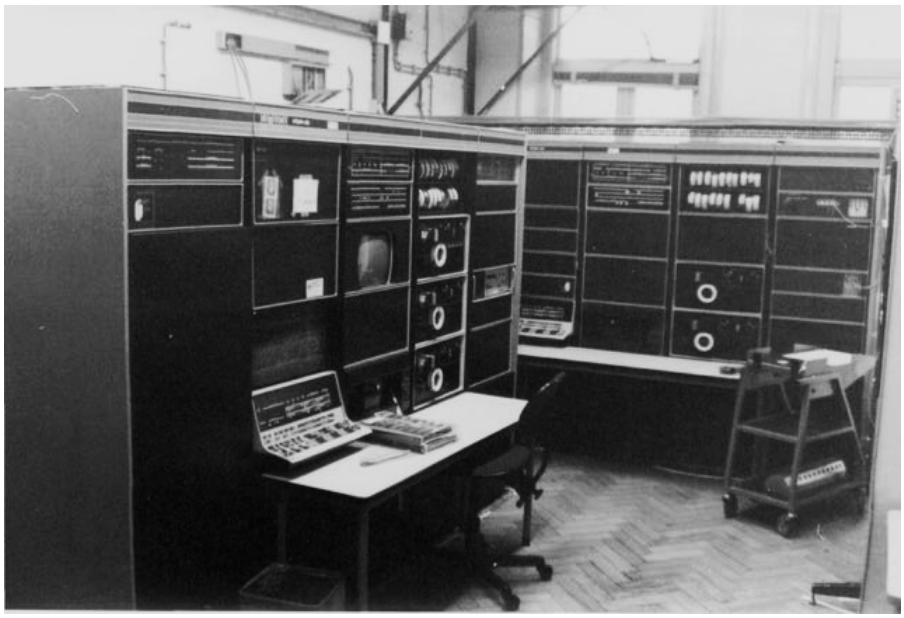


Figure 2 The two DEC-PDP's

As Werner Kaegi's son, I followed the VOSIM-story closely and personally. When Werner did his experiments with speech and music sound synthesis on the DEC computers of Sonology - it must have been around 1978 - I was a teenager. I regularly joined him in the weekends to the Institute. The computer room, where the two PDP-15's were housed, seemed for me more like

[1] MIDIM = Minimum Description of Music. Kaegi gave this name to the software (and the theory behind it) he wrote to control the Vosim generators. The software was written in Fortran 77 and consisted of thousands of code lines. It was used by many students and composers between 1977 and 1986 The term was used by Kaegi before Midi was created and they have nothing to do with each other. He also developed a MIDIL system (Minimum Description of Language) for speech synthesis. Although Kaegi was also a pioneer of speech synthesis in the Netherlands and received international recognition for this research (see publications), the project was halted because the University felt that he was venturing outside his competence.

2 The Institute of Sonology in Utrecht belonged to an interfaculty of the University of Utrecht. In 1986, formally the Institute was not closed but moved from Utrecht to The Hague. But its two most prominent leaders, Werner Kaegi and Georg Michael Koenig left the institute because they could not agree with the new structure within the conservatory of The Hague, where the institute was housed. The main reason for the closure of the Utrecht Institute was that the University no longer wanted to finance Interfaculties. The great flight that collaborations between art and science took later, shows how wrong these kind of narrow-minded decisions were. In the early eighties the Institute of Sonology in Utrecht, was quite famous and attracted around 80 national and international students per year.

3 The MIDIM-group was a collaboration of some young artists/scientists who shared enthusiasm for the MIDIM/VOSIM system. The members were: Jos Janssen, Paul Goodman, Heinerich Kaegi, Pieter Kuipers and Pierre van Berkel and of course Werner Kaegi himself.

an alchemic Faustian laboratory. Temperature stable 18 degrees Celsius, a roaring airco, panels with hundreds of flashing lights and choppy turning tape readers. It fascinated me deeply. While I was writing my first simple Fortran programs on one machine, in the never-ending air-conditioning noise, I heard strange "mama"-sounds coming from the other computer. I couldn't escape the exiting and sometimes frightening impression of a sorcerer teaching an alien creature to learn human speech. Later I assisted my father in the studio and joined the MIDIM-group.

I never forgot these moments and always thought of rebuilding something like VOSIM or MIDIM. When the Coronacrisis was forcing us to keep home, the right moment was there to dive into the code. After one year of programming there is the Vosim66 system, which you can try yourself. GNU Octave was at that moment for me the best choice to be able to make a workable software. It's certainly not the best choice, but as I'm not a professional programmer this was my road. At least, this software is very transparant, open source and Octave is free for everybody.

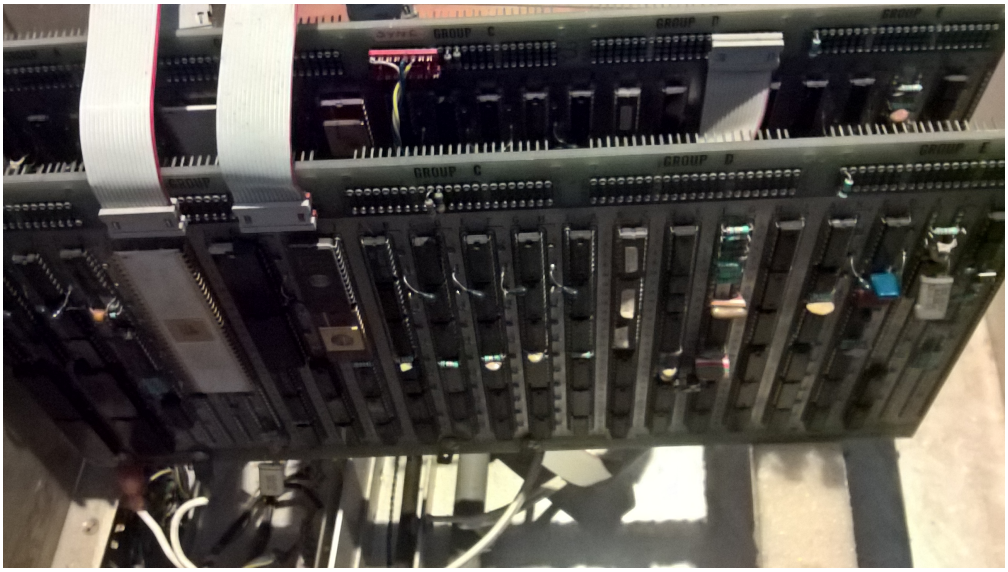


Figure 3 The second generation hardware Vosim generators built around 1985 at the Institute of Sonology

Together with Ludger Hurts (whom I met in 1991 at the Conservatory of Amsterdam, where I gave a MIDIM/VOSIM-course, together with Jos Janssen) I created also many sound examples. We also archived all the old files and adapted them to new file formats. When I contacted some other members of the old MIDIM-group, I made the wonderful discovery that Jos

Janssen (who was an assistant of Kaegi in Utrecht) still kept an old version of the MIDIM-software alive on an Atari-emulator. More pieces of the puzzle came together. We came in contact with Luuk Trip, at the time also student in Utrecht and later professional software engineer, who programmed the first 100% software Vosim-generator. Our mails motivated him to dive into Vosim again, after many years.

Even when the endless memory capacity and speed of modern computers has obviated the need for sound synthesis (all sounds can be sampled and stored in real time), the discoveries of the Vosim model remain valid. So do sounds made with Vosim, we believe. Kaegi -at the time- has made many simulations of acoustic sounds at an early stage in his work. He was one of the first who created synthetic speech. For him that was a way of putting Vosim and the controlling of Vosim (MIDIM and MIDIL) to the test. First, he wanted to demonstrate the versatility of the Vosim synthesis. But even more important for him was to show that digital sound synthesis can have artistic value. That the computer sounds can have enough richness and liveliness and are fully controllable, like in traditional composition. That's certainly not obvious. It needs concepts how to control the computer in a deterministic and for the artist understandable way. I hope that the many sound examples that we have collected -from the past and new creations- can convince the listener of the power of Vosim, also in this time.

The actual software, sound examples and documentation are all work in progress. Obviously, we cannot guarantee that all functionalities are working properly and that all information is complete and correct. What's more, there are many unrealized ideas and plans to make the system more user friendly, more advanced, more precise and more artistically versatile. We appreciate your experience, new wishes and ideas, we would love to hear your new sound creations and we hope you will report problems and short comings.

Heinerich Kaegi, February 2021

2. First things first

Before we go into details of the Vosim model and the controlling of its parameters, we show you in this chapter how to start the program and to make it produce some sound. There are many sounds in the folders we have prepared already which you can easily load into the program.

2.1. Installing GNU Octave

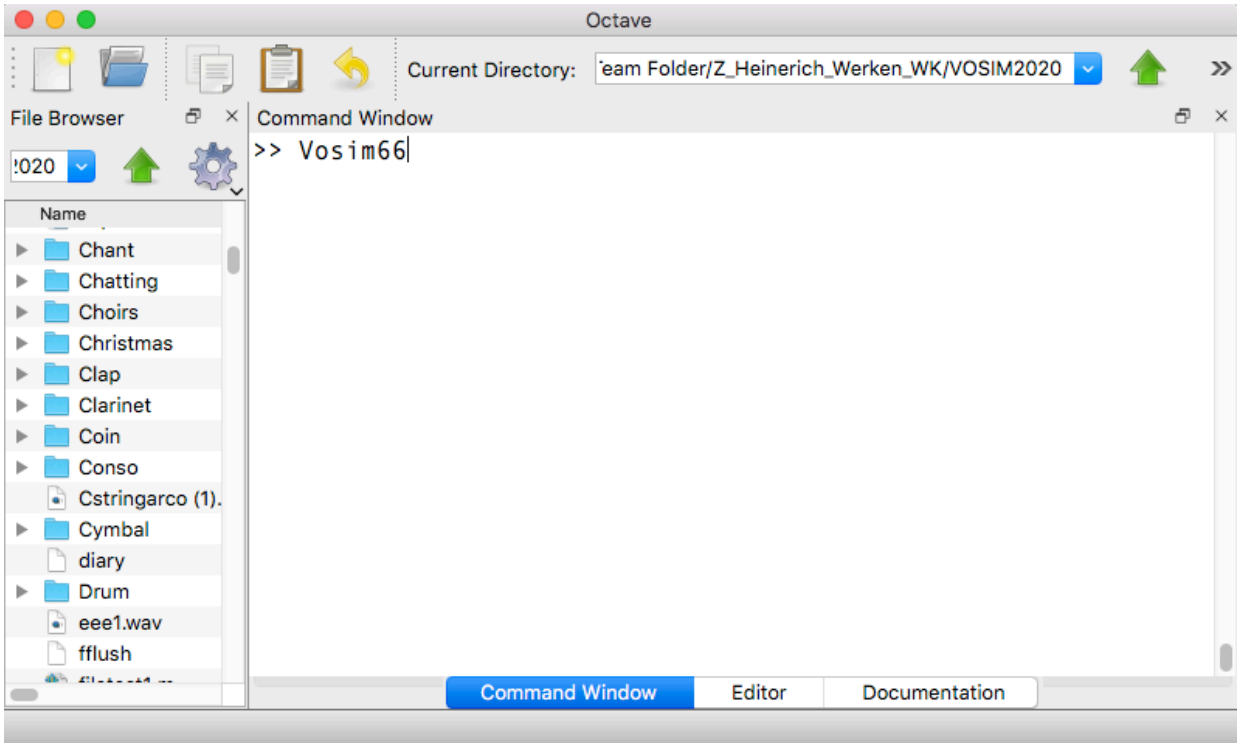
To run Vosim66 you need a completely installed version of GNU Octave with the additional packages *Control* and *Signal*, and for the real time version you need also the package *ltfat*. You can find all information and downloads on https://www.gnu.org/software/octave/. For Windows the necessary packages are a part of the standard download. When you have installed Octave you can determine if these packages are installed by typing >> *pkg list* in the command window.

```
>> pkg list
Package Name  | Version | Installation directory
--------------+---------+----------------------
     control  |   3.2.0 | /opt/local/share/octave/packages/control-3.2.0
       ltfat  |   2.4.0 | /opt/local/share/octave/packages/ltfat-2.4.0
      signal  |   1.4.1 | /opt/local/share/octave/packages/signal-1.4.1
     specfun  |   1.1.0 | /opt/local/share/octave/packages/specfun-1.1.0
>>
```

If you cannot find the right packages installed in Octave, you can try to install them from out Octave by typing >> *pkg install -forge* followed by the package name.

2.2. Starting Vosim66

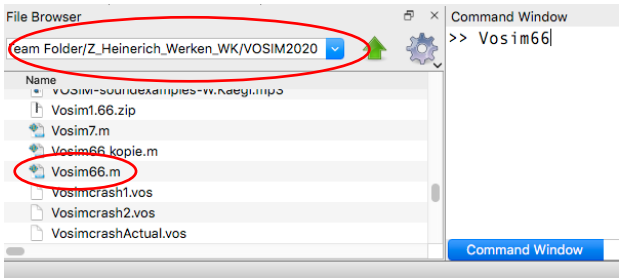
Starting the program is simple. Go to the Command Window of Octave and type *Vosim66.*



The program will announce itself:

```
**********************************
* VOSIM generator Version 1.66 *
**********************************

VOSIM software generator based on the Vosim7 model and MIDIM theory of Werner Kaegi
Created by Heinerich Kaegi January 2021
For info/remarks etc. heinerich@kaegi.nl
```

If the program doesn't start, you probably need to select the right folder on the left of your window. The Vosim66 script-file must be in the list below.



2.3. Using Projects

Vosim66 gives you the possibility to organize your sound projects and all the data in a practical way. The many files the system produces cause easily data loss. We advise you to use project names. Let's suppose you want to create Bassoon-like sounds. You call your project *Bassoon.* Every project has a folder of the project name. When the project doesn't exist, a new folder will be created.

```
Do you want to enter a projectname?(yes or no) yes
Enter project name: Bassoon
This folder doesn't exist jet.
Do you want to keep this projectname and create a new folder?(yes or no) yes
New folder for this project has been created!
All files will automatically be saved in this folder.


MAIN MENU
---------
Project: Bassoon
Loaded file:
Input: 0 seg. Output: 0 seg.
Actual buffer is Inputbuffer

1.Load data
2.Create new sound
3.Assemble segments
```

In case you don't define a project name (which is possible), your files will be stored in the root folder of the Vosim-system.

When you use a project name and you leave the program, without closing the octave window, Vosim66 will ask you to keep the actual project. In case Vosim66 is terminated, but the GNU Octave shell still operates, all the data will be stored. You can simply start Vosim66 again and continue your work without any data loss.

```
**********************************
* VOSIM generator Version 1.66 *
**********************************

VOSIM software generator based on the Vosim7 model and MIDIM theory of Werner Kaegi
Created by Heinerich Kaegi January 2021
For info/remarks etc. heinerich@kaegi.nl


Existing projectname is: Bassoon

Do you want to keep this projectname and all data? (yes or no)
```

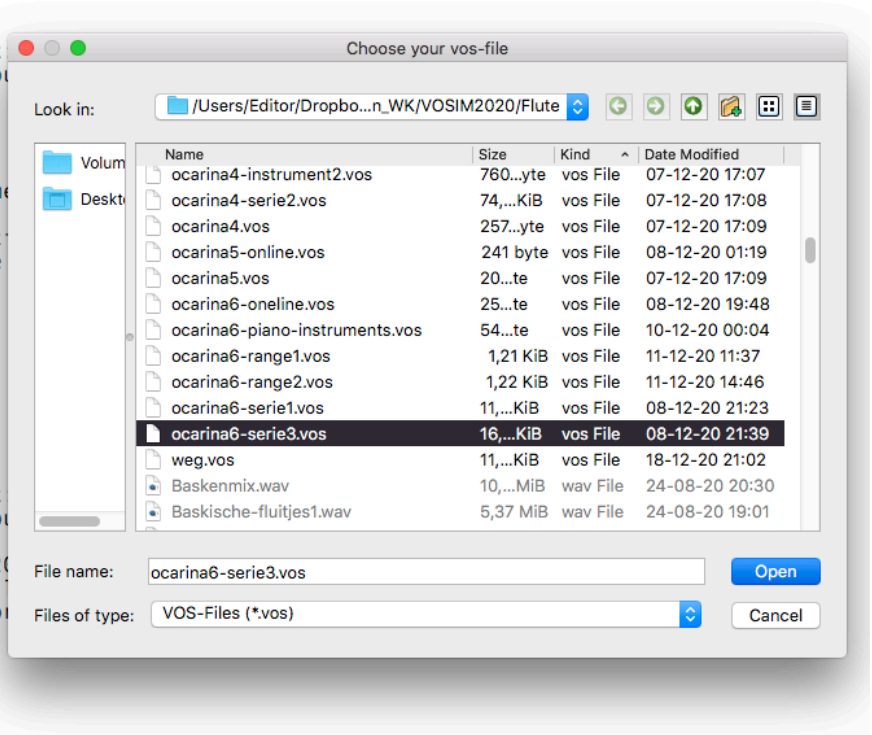
If you choose yes, all the data will be kept in the system buffers. You can immediately continue your work, without the need to reload data or recalculate segments or sound.

2.4. Your first sound

Let's first load a sound and listen to it. It's a good way of checking your sound output system and sound card.

So, choose *1. Load data --> 1. Load Vosim file (2020 standard 1 and 2)*. In the pop-up menu you can look for an existing vos-file.

```
MAIN MENU
---------
Project: Bassoon
Loaded file:
Input: 0 seg. Output
Actual buffer is Inpu

1.Load data
2.Create new sound
3.Assemble segments
4.Play and edit segme
5.Save data
6.Play last calculat
7.Change projectname
8.Create series
9.Exit


--> 1

LOAD MENU
---------
Project: Bassoon
Input: 0 seg. Output
Actual buffer is Inpu

1.Load Vosim file (2(
2.Load old Vosim7 fi
3.Load old Predicato
4.Exit


--> 1
```



In this case we choose *ocarina6-serie3.vos*

You will be offered several different ways to load the data into your buffer. For now we choose option 0, which clears data (when you have loaded sound already) and then enters the

vosim lines into the buffer. As you can see in the screenshot, calculation of the audiosamples is performed immediately after loading, which takes some time for all 65 lines.

```
65 lines found. Load options:
 0=Clear inputbuffer, then load new data
 1=Append new data to old ones
 2=Overlay: old->layer1, new->layer2
 3=Overlay: old->keep layers, new -->layer1
 4=Overlay: old->keep layers, new -->layer2 0

Segmented Vosimdata stored in Inputbuffer (65 segments).

--------------------------------------------------
Calculation of samples (STOP with s-key)
Segmentnr: 7/65   50%
```

Mention that you can stop calculations with the s-key on your keyboard. Of course in this case the audio will not be played. When calculations have been finished the generator will tell you the calculation time, in this case 120.7 seconds and the player will start.

```
--------------------------------------------------
Calculation of samples (STOP with s-key)
Segmentnr: 65/65  100%
Assembling sounds
Layer: 1 Segment: 65/65
Calculation time (s): 120.7
--------------------------------------------------

Audioplayer: p=pause/continue r=restart .=forward ,=backward z/x/c/v=layer1/2/3/4 a=all e=exit

       0    10    20    30    40    50    60    70    80
 74.8s |=====|=====|=====|=====|=====|=====|=====|=o---|-- playing all  |
```

The player offers you several keyboard options:

key	function
p	pause the sound and replay
r	restart from the beginning
, and <	scroll backwards
. and >	scroll forward
z/x/c/v/a	play only layer 1,2,3,4 or all together
e	exit the player

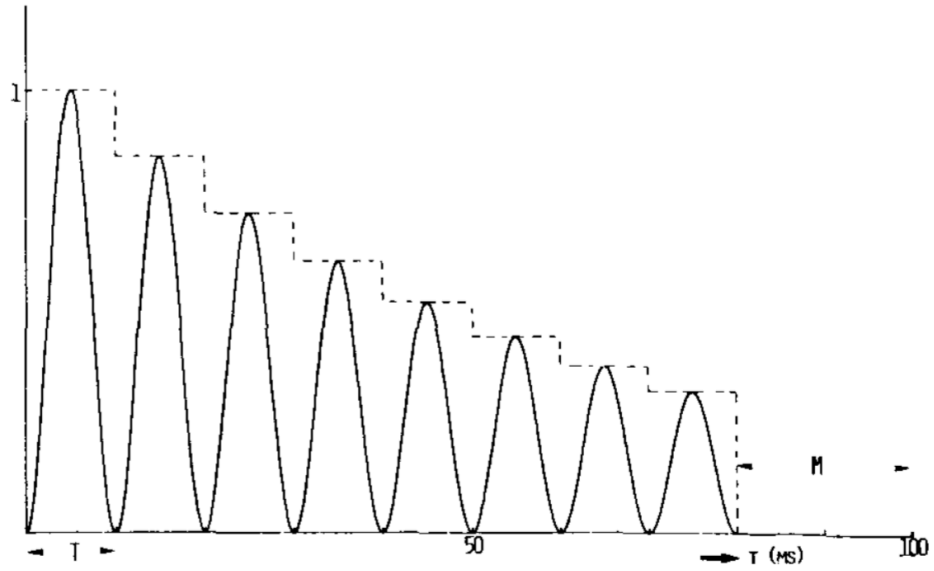
Hitting the e-key exits the player and leads you back to the main menu.

If you haven't heard any sound during the VOSIM player shows the timeline with a moving o, then you have to check your sound card and equipement.

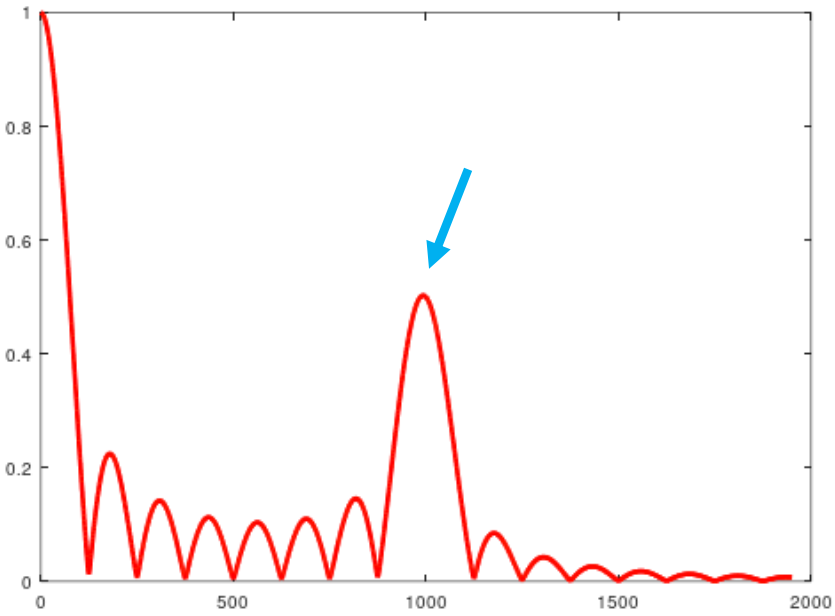
3. The Vosim model in a nutshell

The Vosim model has been documented in several articles. In this chapter we will not go into details, but only give you a very limited summary of how the Vosim signal works and sounds.

The Vosim signal in short can be described as one or many pulses, followed by silence. Together they form "a period" in time (indicated by T'). The period is this part of the signal which is repeated to create the "fundamental frequency f_0" (the main tone you hear, also called the pitch) and it's "harmonics" (f_1-f_n, which produce the sound quality). Let's watch such a period in the figure below.

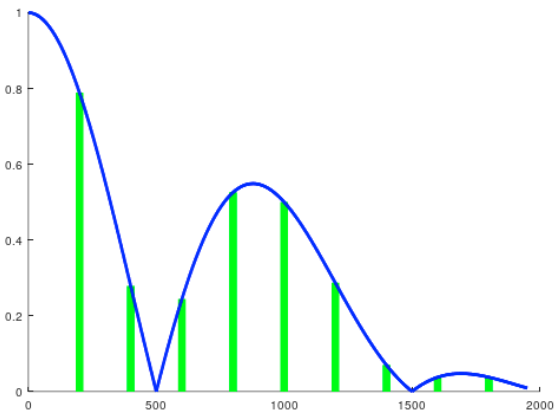


Even when this signal resembles a sinewave, it's spectral characteristics are much richer and more complicated, because of the silence or "delay" (indicated by M) after the pulse train. While a sine wave only has one frequency, we find in the Vosim signal a second very important feature. We call it the "Formant frequency F". This frequency is determined by the width of a single puls T. In the Vosim spectrum we can demonstrate this more clearly.

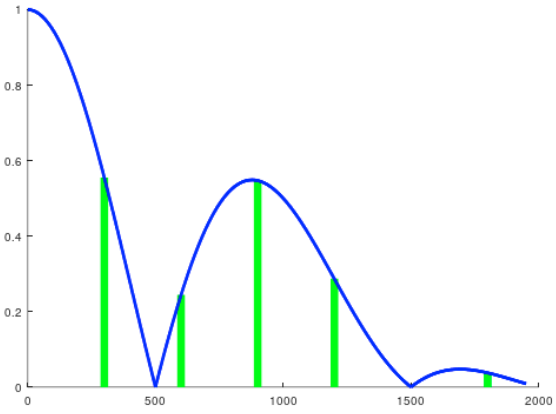


The formant is the maximum at 1000 Hz (blue arrow). The term *formant* is used in linguistics to indicate the main components in vowel sounds. You are now probably wondering where the fundamental and the harmonics have gone. Well, while the shape of the spectrum depends on the fundamental frequency (which is thus characteristic for it's sound quality), the fundamental is a free choice, namely the tone ("the note"). The fundamental and it's overtones/harmonics are peaks under this line, as we demonstrate below (green peaks). Their height is determined by the spectral envelop. Of course, the harmonics are always on the same distances of each other, but they can move around, when the tone changes.
We show you four cases of the same Vosim spectrum, but with different pitches, 200, 300, 400 and 500 Hz. The formant is at 1000 Hz. While the envelop remains the same, the harmonics are moving upwards, following the red line.

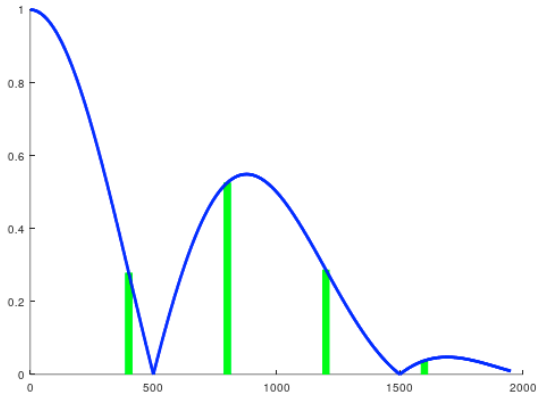
Different pitch in the same Vosim spectrum (N=2, Decay=100%, Formant=1000 Hz)



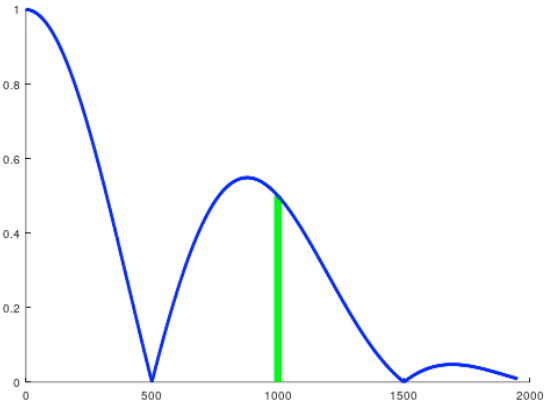
Pitch=200 Hz



Pitch=300 Hz

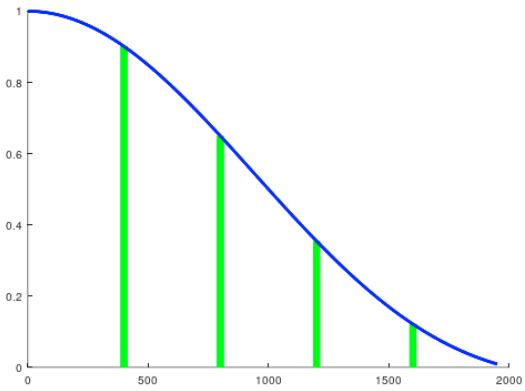


Pitch=400 Hz

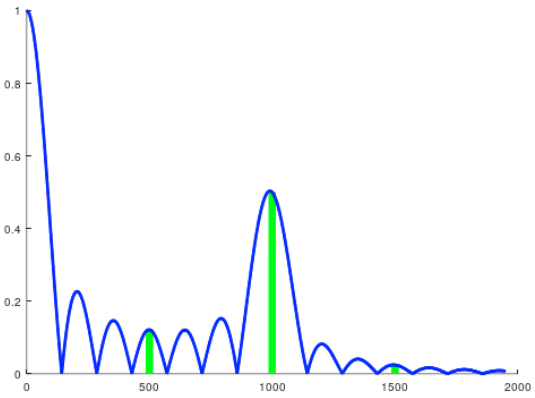


Pitch=500 Hz

Another two important features of the Vosim signal and it's spectrum are the number of pulses N and the way they decay, which is determined by the decay percentage C.



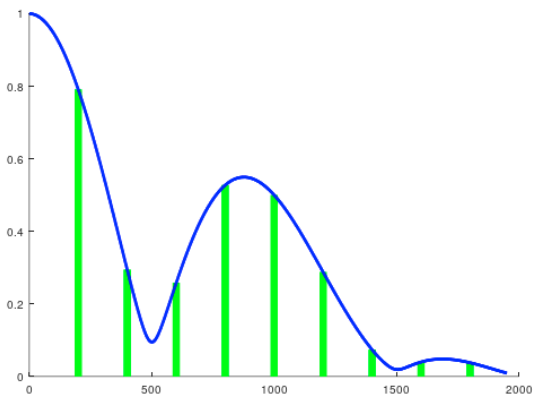
N=1 F=1000 Hz Pitch=400 Hz



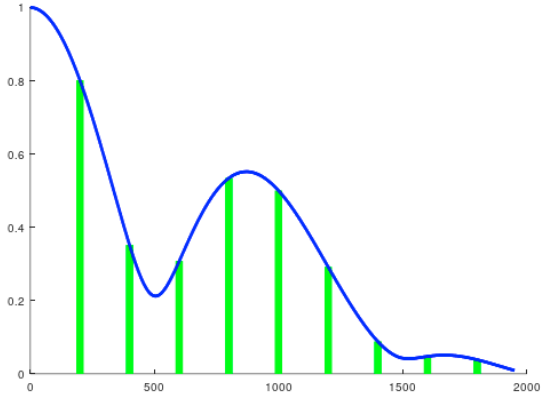
N=7 F=1000 Hz Pitch=400 Hz C=100%

In the two graph you can see how the spectrum changes between N=1 and N=7. For one puls the low frequencies are much stronger than the high ones. When N becomes bigger, frequencies under the formant frequency are muffled and sometimes even suppressed.

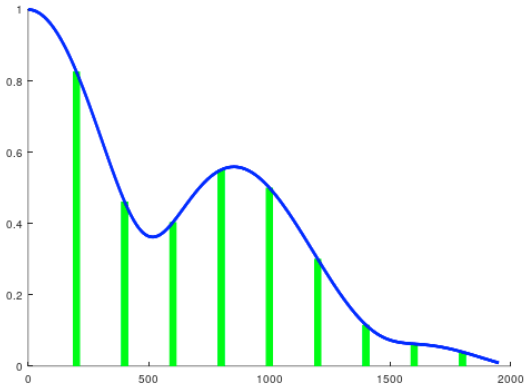
To adjust the strength of the harmonics below the formant, the decay value is useful. The lower the percentage of the decay, the higher these components are, as you can see in the graphs below. Here we have: N=2, F=1000 Hz, Pitch at 200 Hz.



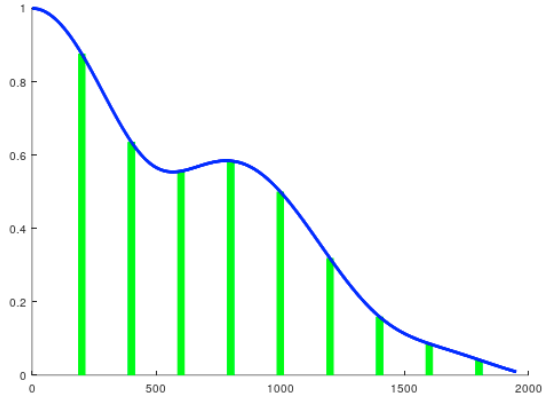
C=80%



C=60%



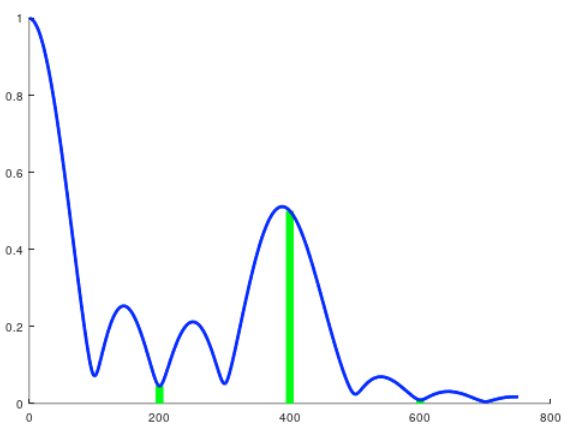
C=40%



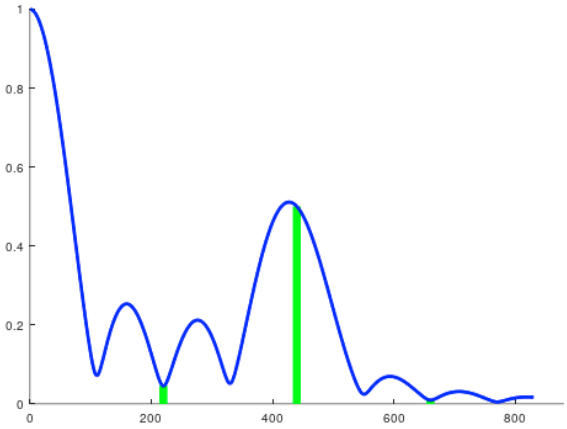
C=20%

The Vosim model is based on two combined Vosim signals, to create two vowel formants. As we know from linguistics a minimum of two formants are needed to generate a clearly recognizable vowel sound. If you add two Vosim signals (from two generators) , their corresponding spectra will be added too. The final spectrum therefore contains two formants (peaks).

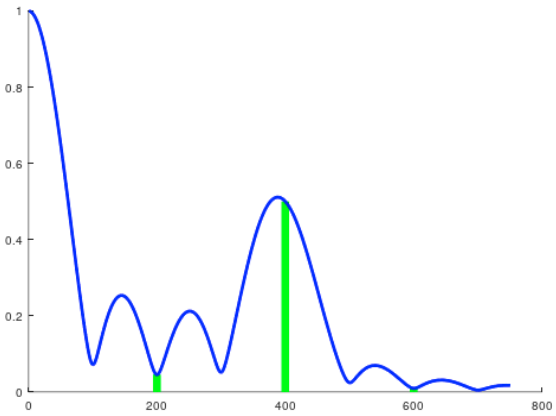
Finally we would like to mention the option in Vosim66 to make the formant frequency dependent on the pitch (the so called "Moving formant"). This means that in the spectrum the harmonic structure remains the same for different tones. 7



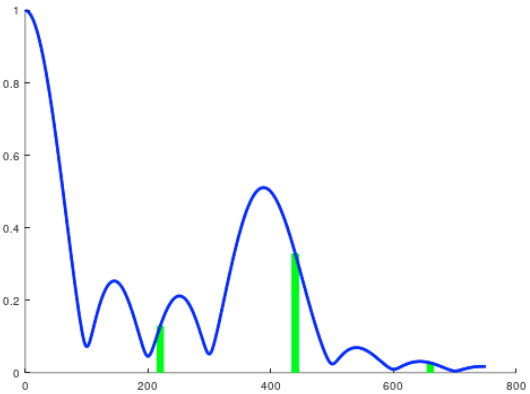
pitch=200 Hz
Moving formant situation



pitch=220 Hz
The spectrum shifts with pitch



pitch=200 Hz
Fixed formant situation



pitch=220 Hz
The harmonics shift upwards

4. Data and concepts

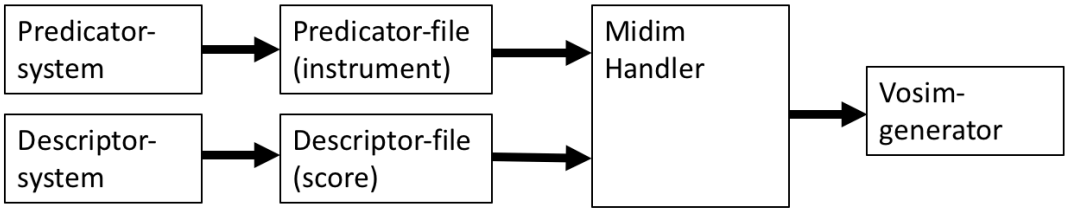
In this chapter we will describe the way data are stored in the software, in data-files and as audio samples. For those who knew the MIDIM-software, we will mention the old Midim-structure and compare it with the new concepts in Vosim66.

4.1. Old MIDIM concepts

Werner Kaegi, when het designed the MIDIM-software not only wanted to store data in buffers and files, but he was very precise in labelling the different data-streams in a conceptually correct way. So, to understand the concepts he used we will describe them first. In the old Midim system, a distinction was made between the following files and associated data arrays:

* Vosim file - sound events
* Predicator file - sound concepts (instruments)
* Descriptor file - scores
* Library file - sound families

In the original system there was hardly any overlap. A predicator - let's say an instrument - could not be heard without a descriptor - the notes. The old structure can be seen in the diagram below.



The predicator-system created a sound concept/instrument, the descriptor was the composers tool to "write" a score. The Midim-handler combined the two and fed it to the generators which made the sound hearable. Nowadays computers are so quick, that most audio systems are calculating faster than they can play. That needs more integrated storage. In the new Vosim66 all sound design is based in the enlarged Vosim-vector (Vosim2020 standard).

A Vosim2020 file and also it's corresponding Vosim-buffer consist of pairs of Vosim lines, of which the odd lines belong to generator 1 and the even lines to generator 2. The pairs form one pair, we call “a segment”. Several segments form together a series of segments.

4.2. New Vosim66 concepts

While in the old MIDIM playing of predicators and/or descriptors was not possible, in Vosim66, you can hear your sound immediately. That means necessarily that some prosodic parameters like pitch and duration (which originally only were found in the descriptor) are now part of the Vosim2020-vector. It also means that functions, which in the old storage were only settled in the predicator to define the sound concept (like "moving formant", Midim function F1), now have been shifted to the Vosim2020 vector.

In the new Vosim66 system the editor is the heart and interacts with the Vosim-buffers. You can adjust the Vosim vector per segment using the keyboard. The segment can be immediately converted to samples and played. The calculation is in most cases so fast that the calculation time is less than the playback time.

This gluing together may for some of you seem confusing at first, but has the great advantage of making the setup and operation of the system much simpler and more flexible. In the old system, the separation between “input” (the input of the data) and “output” (the sound you hear) was very large and it was therefore obvious that you keep the data separate. This was mainly due to the split between software and hardware.

The backside of the simplification of concepts is, that the fundamental understanding of the difference between a sound concept or instrument (which needs to be stable) and the compositional controls (which create the field of freedoms the composer has) is blurred. It's therefore especially important that the user is aware as to which concept a variable belongs. A formant is for example clearly part of the "instrument" or sound concept. Pitch on the other hand is obviously part of the "notes".

The new Vosim2020 segment as we use it in Vosim66 can therefore fulfill various roles. Sometimes it is a predicator, sometimes a descriptor and sometimes a sound. This has resulted in the number of Vosim parameters in the Vosim vector being expanded from 12 to 25. And that number will also be expanded in the future.

In the new structure, all parts have been merged into one vector. It contains parameters from both streams: both “instrument” data and “notes” information. The MIDIM handler, which previously processed the data, applied the so-called Midim functions and performed the domain corrections, is partly merged with the input modules and editor, partly with the "Vosim generator". The modern Vosim generator is nothing more than a function within the Vosim66 software. The original Midim functions are not all included in the Vosim66 system, but will be built in later versions.

By merging the vectors into one advanced vector, the system becomes much more flexible. Which function the Vosim line fulfills no longer depends on the data structure, but on which “input” we use when processing the data.

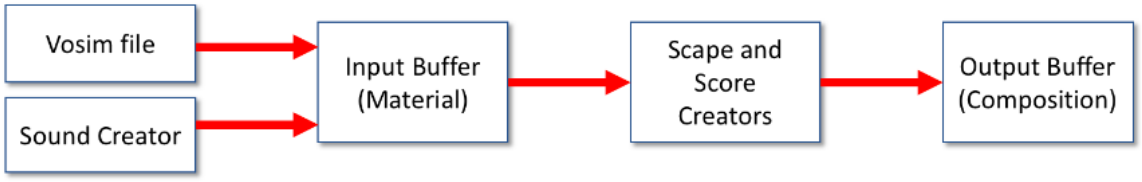
For example, it is possible to merge data from a “score” (descriptor) with a Vosim line and produce new Vosim lines. The “score” then supplies the data of eg pitch and duration, the Vosim line supplies the sound properties.

4.3. Input buffer

Buffers are nothing more than storage tanks for the data, in this case Vosim lines. In Vosim66 are two buffers that store Vosim data: the input buffer and the output buffer. These two buffers have the same structure. Each row consists of a Vosim line, alternately for generator 1 and 2. Pairs of two lines form the segments. The columns contain the vector parameters. In the old system the input buffer would be your predicator, your output buffer would be your Vosim data.

The input buffer can contain n segments of two lines each. The input buffer is intended for storing your first material. That can be:

* An already created Vosim file of several segments that you want to play.
* A new segment generated with the creator (sound creator), a sound, which is used as a starting point for longer series, for example by combining it with a score.
* Three segments, which form an “instrument” consisting of prefix, body and suffix (the old predicator structure, without the stop).
* Several groups of segments, which form a sound library of instruments or sounds.

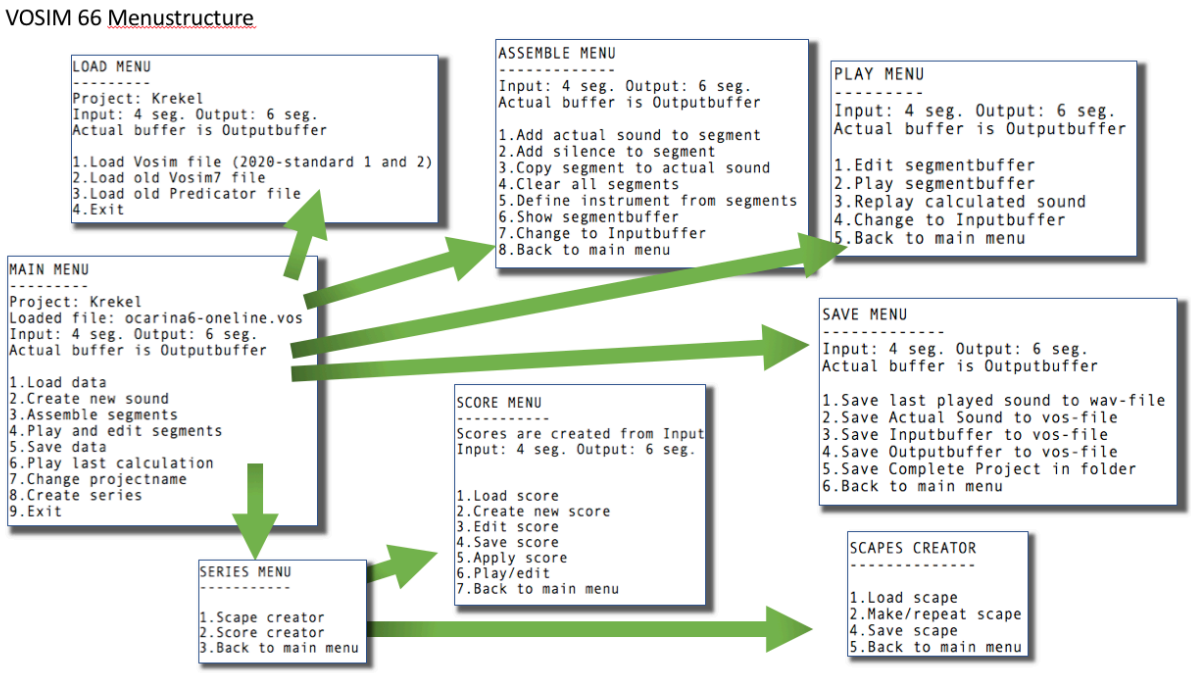


4.4. Output buffer

While the input buffer contains mainly your starting material, like sounds, instruments and patterns, the output buffer is more like a "composition" or a series of changing sounds, based on the material in your input buffer. In general, your workflow goes from the input buffer to the output buffer. Suppose you made several sounds, like instruments with the sound creator and you stored them in your input buffer. Now you want to use them in a changing scape. With the scape creator you define your filter parameters and you apply them to the segments in the input buffer, to create a longer sequence in the output buffer. There your scape can be listened to or further modified with the editor.

5. Menu Structure

Let's first dive into the complete menu-structure of the software. In the scheme below you can see the first and second level menus of the program. From the main menu all functionalities can be reached. Functions with no submenu are not in the scheme.



We will first discuss all the different functions of the main menu, then we will enter more deeply into the different submenus.

1. Loading data

Loading data mainly consists of entering Vosim files. Each Vosim file contains a variable number of Vosim segments of each two generator lines (odd numbered lines--> generator 1, even numbered lines --> generator 2). The 2020-standard Vosim files correspond with the parameters in the two buffers and thus also with the parameters which can be edited in the

editor. As you load older files, they always will be converted to the new standard. Loaded data will be stored into the input buffer.

2. Create new sound

In the sound creator you will be asked to respond a series of questions about the sound you would like to create. To use the sound creator proper knowledge of the Vosim model is needed.

3. Assembling segments

The assemble menu merges some useful functions to erase, change and control the content of your buffers. It also defines the way "instruments" function.

4. Play and edit

The play menu has a central role in the software. It gives access to the editor. Most of the sound design will take place in this editor.

5. Saving data

To store your creations there are several options you can find in the save menu.

6. Using Series

Series are also series of segments. Series can be created in two ways: by creating a score of notes, which then can be processed to a serie of Vosim segments. Or series can be created by the definition of a soundscape. Scapes are series of segments with small variations. All series are created from one or several segments in the input buffer (your sound material) and are stored in the output buffer. The score and the scape menu we will discussed later.

6. The editor

To be very clear: a segment is a code to control two parallel generators. In fact it is a sound-event. The moment you want to play the segment the samples will be calculated and played on your sound card. To control this sound you can manipulate the parameters of this Vosim vector. Several segments form together a continuous sound.

The editor works on the input or the output buffer. Which buffer is used can be seen at the top of the menus (“Actual buffer)”. To use the editor, choose 4 in the main menu and 1 in the play menu. With the option “change to… buffer” you can switch between the two buffers.

```
PLAY MENU
---------
Input: 0 seg. Output: 0 seg.
Actual buffer is Inputbuffer

1.Edit segmentbuffer
2.Play segmentbuffer
3.Replay calculated sound
4.Change to Outputbuffer
5.Back to main menu
```

The moment you are in the editor, you can see the following prompt:

```
[1/3]=>
```

It indicates on which segment you are at the moment (here nr. 1) and how many segments are loaded into your buffer (in this case 3). After the arrow you can enter one or more letters to control the editor.

6.1. Play your segment(s)

You can play your segment with lower case "c" + enter (c=calculate). The system calculates the complete segment and plays it. The longer the segment, the more time is needed to calculate the samples.
There are two modes to play sound in the editor, standard mode and loop mode.
You switch to the loop mode with comma + enter.

```
 [1/3]=> ,
***LOOPPLAYING ON
```

This can be turned off again with dot + enter. In loop mode, approximately 1 second of the sound is played and repeated 10 times after every new command. In this mode you can immediately hear what you are changing.

There are some more commands for playing several segments in one row:
C = plays around the actual segment, one before till one after.
CC=plays the actual instrument, always starting on segment 1,4,7,10 etc.
CCC=plays all segments in your buffer.
r=replay, just plays the tone buffer again, with no calculations or changes.

6.2. Using the keys

Most parameters can be changed in smaller or larger steps using the keys. So, if you want to increase the length of your segment, for example, you press l + enter (lower case L and enter).

The following conventions apply:

- Lower case letters decrease the value, upper case letters increase the value
- Repetitions of the letter (up to three, sometimes four letters) make the step bigger.
- The first time you hit a letter, Vosim66 will show you the value without changes.

6.3. An example

Suppose you have a formant F1 of 2000 Hz. You want to make that 100 Hz lower, then you type t+enter and the system will inform you about the formant

```
[3/3]=> t
Formant1 has value: 2000  (Min: 10 Max: 20000 Add: 1/10/100/)
```

Then you type ttt+enter and the value is reduced to 1900 Hz:

```
[3/3]=> ttt
Formant1 has changed to: 1900
```

If you type TT then, it will be increased to 1910 Hz, so 10 Hz higher.

```
[3/3]=> TT
Formant1 has changed to: 1910
```

The range of available values and the increments associated to the number of repetitions can be seen in the information at the end of the line.

```
[3/3]=> t
Formant1 has value: 2000  (Min: 10 Max: 20000 Add: 1/10/100/)
```

In this case you can enter values between 10 and 20000 Hz. One key hit jumps 1 Hz, two key hits change 10 Hz and finally, three hits jump 100 Hz.

6.4. Setting menus

Some parameters need settings. For example, the output mode. Go first to loop mode with the ,-key to hear immediate change of output. Then hit o+enter to see this menu:

```
[3/3]=> o

-------------------------
Outputmode
-------------------------
1.Mono mode (G1+G2)
2.Stereo mode 80%(G1/G2)
3.Only left channel (G1)
4.Only right channel (G2)
5.Stereo mode 90%(G1/G2)
6.Stereo mode 100%(G1/G2)

Enter(0=nochange) |
```

The output mode determines how the sound is distributed to the two audio channels. Standard is mode 2: stereo (80%). You can now change the output mode with option 1+enter to hear the sound mono.

Some settings menus are hidden behind 4 times repetition. For example: you can switch between fixed formant or moving formant. This can be different for the two formants (generator 1 and 2).
We can demonstrate this by hitting tttt+enter:

```
[3/3]=> tttt

-------------------------
Formant gen. 1
-------------------------
1.Moving
2.Fixed

Enter(0=nochange) 1

[3/3]=> T
Q-factor 1 has changed to: 0.6

[3/3]=>
```

As you can see, there appears a menu to choose moving formant, by option 1. Using the T-key again will change the standard Q-value from 0.5 to 0.6. (The t key doesn't anymore control the formant-frequency (in the case of fixed formant), but has changed to manipulating the Q (because you are using now a moving formant)).

6.5. All keys in short

An overview of all keys and their functions can be seen in the software itself. The scheme refers to the letters and symbols on your keyboard

```
**********************************************************************************************
          PLAY & EDIT SEGMENTED SOUND Segment nr.: 1 Keyboard controls
**********************************************************************************************
!=Stop noteplaying   1-8=Choose octave   -=delete segment  +=copy seg  ?=show spectra
--------------
MODULATION     | e=exit          | T/Y=Formant1        U/I=Amplitude1 | o=Outputmode p=act.seg.info
Q/W=Frequ1/2   | r=repeat        | tttt=moving/fixed1  uuuu=envelop1  | oo=Syncmode  P=all seg info
A/S=Depth1/2   ------------------------------------------------------------------------------------
d/dd=Width1/2  | F=Pitch         H/J=N1/N2           k=Linkmode     L=length
z/x=Mode1/2    | G=Pitch End     hh/jj=decay1/2      kk=ratio/beat  llll=Duration mode
               ------------------------------------------------------------------------------------
zz=layer       c=calculate+play  V/B=Formant2        N/M=Amplitude2 ,=Loop on  </>=Prev/Next seg.
               C/CC=Playmore     vvvv=moving/fixed2  nnnn=envelop2  .=Loop off <</>>=Jump 10 seg.
**********************************************************************************************

[1/3]=>
```

For more detailed information, we refer to the appendix. Mention that both generators can be controlled by these keys.

There is a certain system in the ordering of the functionalities. On the left side you find the modulation controls in two columns QAZ for generator 1, WSX for generator 2.

(blue). Then you find from top to bottom some essential control keys on the left side er and c: e=exit the editor, r=repeat sound and c=calculate and play sound. Other important controls you find left op,.<> (pink).

Then on top from left to right TYUI are main controls of generator 1 (like formant and amplitude), on the bottom from left to right VBNM are the same main controls for generator 2. (green)

Finally controls which hold for both generators, you can find in the middle row, from left to right FGHJKL. (yellow) F and G control pitch, H and J control the spectrum, k the link between the pitches of the generators and L the duration of the segment.

By 1/2 we mean that one key is for generator 1, the other for generator 2. For example: Q and W are for controlling the modulation frequencies of generators 1 resp. 2. With repeated keys certain special actions are indicated. For example: with UUUU+enter one enters the amplitude envelop menu for generator 1. p shows the actual segment values, and upper case P gives an overview of all segments in the buffer at that moment. With a question mark you can plot a spectrum.

6.6. The Note playing mode

Pitch values can be entered in frequency or in notes. Standard is frequency. In the editor, the number keys provide a jump to the corresponding octave. So 3 means change the pitch to the third octave (starting with the standard A4=440 Hz). This also turns on the “Note playing mode”. With upper and lower case f you can now jump up and down the basic scale of semitones, instead of changing frequencies. The “Note playing mode” can be switched off with !.

6.7. The Editor modes

There are several modes in the editor. The mode can be changed with xx.

```
 -------------------------
 Editormodes
 -------------------------
*0.Mode 0: Segmentmode
 1.Mode 1: Instrumentmode
 2.Mode 2: Soundmode

 Enter:  1

 Change to option: Mode 1: Instrumentmode
```

The standard mode is the segmentmode. In this mode there is no special connection between segments. They all function independently.

In the instrument mode, three segments are always connected, beginning with 1-3, 4-7, 8-10 etc. The three segments are seen as parts of one sound and we call them prefix, body and suffix. In this mode, pitch changes are performed on all the pitch dependent segments of this instrument at the same time (standard body and suffix). Using the c for playing, plays in this mode all three segments of the instrument. In the instrument mode the prompt of the editor changes, indicating the number of the instrument and the segment which is edited.

```
Instrument:1 prefix [1/345]=> |
```

In this case the editor is on the prefix of the first instrument.
For more information of instruments see...

The third mode, which is called sound mode, combines 1-4 layers to one sound. This means that you can create more complex sounds, by adding a maximum of 8 generators. For more information see...

7. Some workflows

In this chapter we will guide you through some possible workflows. Of course this is not the only way to use the software but it can give you insight into the intentions we had, when the different functionalities were designed.

As mentioned, Vosim66 has a flexible set-up. The central module is the editor, which works together with the Vosim generators. The editor can be used not only to listen to and edit a segment, but also to create longer series of segments. Creating and adjusting a new sound takes time and experience. We will discuss this later.

Broadly speaking, there are four main workflows. We discuss them here roughly to give you an overview of the possibilities. For more details you can jump to the description of the different used functionalities.

7.1. Pasting segments

You can load a sound from an existing file or create it yourself in the sound creator. In both cases you have a segment in the input buffer to start with. Your editor functions in segment mode. Now you can edit your sound as you wish with the editor. Probably your sound is 1 or 2 seconds long, but you would like to create a longer series of several segments, with changing sound qualities or different tones and durations. To achieve this goal you can copy a segment to the end of the existing series (+ key). Every time you change the sound a little, so it develops. You can make the series as long as you want by just copying and editing. You can also remove segments (- key). You can easily navigate through your input buffer(< > << >>), play each segment,

adjacent segments or the whole series (c C CC CCC). When you are done, you leave the editor and you can play and save the entire sequence.

7.2. Scores

If you have one segment in the input buffer, you can think of it as an “instrument,” a sound that can serve as the basis for a series of sounds, by changing certain parameters. We call these parameters the "prosodic parameters". Think of playing an instrument. You can change pitch, duration, articulation and loudness etc. Such series of notes can be made with the “series” functions. There are two options: the scapes for creating soundscapes and the scores for creating scores. Both functions use segments from the input buffer as the starting material. With the scores option you can write a score. The notes will be applied onto the starting segment in your input buffer and copied to your output buffer. For example, you can create a melody. You can use several layers to make the melody polyphonic and you can use in every layer another sound from the input buffer. For more information, see...

7.3. Scapes

The scape creator works in essence the same as the score creator. It copies segments from the input buffer to the output buffer by changing pitch, duration and amplitude. Instead of defining notes, you determine the ranges of the tone, duration and loudness and the way these parameters develop in time. You can choose random values, wave patterns or just scales. The scape creator gives you the possibility to create very quickly moving sound worlds based on one single sound. You can make random compositions in a certain part of the tone scale. You can easily fill several parallel layers with resembling sounds to investigate how your sound behaves with different tones and durations. You can create scapes to listen to your sound through the whole tone range. You can also enlarge your sound to a continuous scape, moving around in random or waving patterns. Think of sea waves, rain, bubbling, etc. For more about scapes see...

7.4. Instruments

Following the old MIDIM system, you can define a sound based on not only one, but three segments, which we label with: prefix, body and suffix. To do this, you first create three segments in the input buffer, which form one sound concept. In the series functions you can choose whether you want the single-line option or an “instrument” (three segments) to create a series. Of course, you can save "instruments" in the same way you can save series of sounds. You can also save scores and apply them to other sound material. Working with instruments needs a very detailed way of designing, but it finally delivers very dynamic and lively sounds, which are especially nice in combination with scores. For more information see...

7.5. Complex sounds

There are many sounds which can be generated or simulated with two Vosim generators, as Kaegi demonstrated already years ago. But there are certainly sounds which have more complex spectral properties. For this, Vosim66 offers you the possibility to combine a maximum of four layers, which means 8 generators. Use of these complex sounds in scapes or scores is not yet possible, but will be implemented in later versions. For more details see...

8. The Main Parameters

The Vosim66 program uses the new 2020 vector standard. Where the old Vosim 7 standard was strongly determined by the hardware technique of the seventies, the new 2020 vector uses more directly understandable parameters. In this chapter we will discuss them and their sound interpretation.

8.1. The Vosim vector

To give you an idea how the data in the buffers are stored, watch this example of five Vosim segments:

```
****************************************** Input buffer (p.1/4) ******************************************

          Layer  Formant End     Pitch   End    Ampl. End  Decay  N  ---Modulation---   LP  Sync Link
          EdMode (Hz)    (Hz)    (Hz)    (Hz)   (%)   (%)  (%)       Mode Depth Freq.   (s)  QP  Instr.

Segment 1 ------------------------------------------------------------------------------------------------
            1     0.0      0.0   311.0   311.0   100   100   100   1     1    5  0.1   1.009   1.0   2
            1  6010.0   6010.0   311.0   311.0     3     3   100   2     1   40 10.0   1.009   7.0   0
Segment 2 ------------------------------------------------------------------------------------------------
            1     0.0      0.0   440.3   440.3   100   100   100   1     1    5  0.1   1.287   1.0   2
            1  6010.0   6010.0   440.3   440.3     3     3   100   2     1   40 10.0   1.287   7.0   0
Segment 3 ------------------------------------------------------------------------------------------------
            1     0.0      0.0   414.9   414.9   100   100   100   1     1    5  0.1   0.961   1.0   2
            1  6010.0   6010.0   414.9   414.9     3     3   100   2     1   40 10.0   0.961   7.0   0
Segment 4 ------------------------------------------------------------------------------------------------
            1     0.0      0.0   329.3   329.3   100   100   100   1     1    5  0.1   1.090   1.0   2
            1  6010.0   6010.0   329.3   329.3     3     3   100   2     1   40 10.0   1.090   7.0   0
Segment 5 ------------------------------------------------------------------------------------------------
            1     0.0      0.0   465.1   465.1   100   100   100   1     1    5  0.1   1.271   1.0   2
            1  6010.0   6010.0   465.1   465.1     3     3   100   2     1   40 10.0   1.271   7.0   0
```

Each segments consists, as we mentioned, of two lines, for every generator one line. The segments are calculated and played from top to bottom. In the above list not all 25 parameters of the vectors are shown, but just the most important selection.

The list you can print easily in the editor with the upper case P. The lower case p gives you more detailed information for just the actual segment on which you are at the moment.

8.2. Pitch (f-key) and link mode (k and kk)

Pitch (f-key) is the note of musical sounds. Pitch can be denominated by musical note names or by frequency. In the spectrum pitch corresponds with the so-called fundamental or lowest frequency component of the harmonic series of overtones. In the standard settings pitch is given in Hz in Vosim66, but in the note playing mode it can also be indicated by note names and octaves. A_4 is the reference tone of 440 Hz.

Basically the pitch holds for both generators. The way the pitch of the second generator is linked to the pitch of the first one, is determined by the *link mode*. (k-key) There are four link modes:

```
-------------------------
Link between generators
-------------------------
*0.Mode 0: Independant pitch
 1.Mode 1: Same pitch; P2=P1
 2.Mode 2: Fixed ratio QP
 3.Mode 3: Fixed beat QP

 Enter:  2

 Change to option: Mode 2: Fixed ratio QP
```

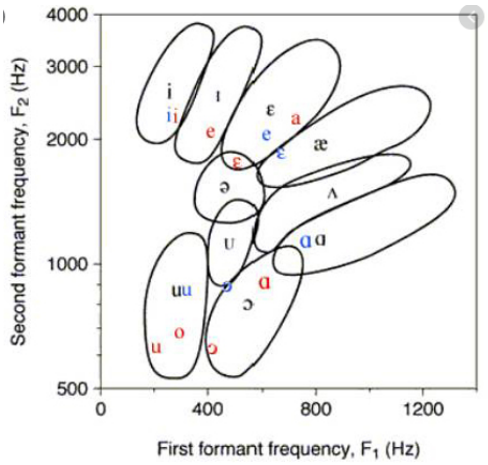
Mode 1 is default. Mode 0 is mainly used in old Vosim 7 files, but not anymore in the new standard. So, in most cases the second generator follows the first one with the same pitch. (If you want separate pitches, use layers).
But some sounds have a strong higher component in the spectrum, following the pitch in a fixed way, for example always as a perfect fith. For this you can use option 2 with a factor 1.5 (M3: 5:4=1.25, P4: 4:3=1.33, P5: 3:2=1.5, P8: 2:1=2 etc.). It's important to understand that this ratio is not an interval between tones of different sounds, but it's an inherent part of the sound.

You can also choose to define a fixed "beat" frequency between the two generators. The "beat" frequency can be heard when two sounds with nearly the same frequency interfere. (Mathematically it's the difference.) This is especially useful when sounds vibrate *in space* in slow movements. Think of gong sounds. In mode 2 the ratio is given by QP, in mode 3 this is the beat frequency, determined by kk.

8.3. Formant and Formant Shift (keys: T Y V and B)

The formant F is given in Hz and determines the strongest maximum in the spectrum. In spoken language the formants make vowel sounds. Choosing the formants of the two generators is very important for *how* it sounds. To make this more concrete we give the vowel-chart with F_1 and F_2 in figure 3.

Figure 4 Vowel chart with the freqencies F1 and F2 of the main formants



Many sounds of all kinds have vowel-like qualities, which can be heard and used to determine the two formants. Just knock on your furniture and listen. What vowel can you hear? In the Sound Creator you can directly choose a vowel and Vosim66 will fill in the frequencies for you.
Think also of formant-singers, who don't change the tone/pitch, but the vowel-sound in their songs. Formants can be controlled with the T and V keys in the editor.

In some sounds the formant shifts up or down in time. Think of a sound like the word 'wow'. The sound changes in the course of the word from /a/ to /u/. This formantshift can be created by making F_{1end} or F_{2end} higher or lower than F_1 or F_2. This end-values are controlled by the Y and B keys.

8.4. Amplitude envelop (Y U N M keys)

Another very important part of creating a sound is determine how its amplitude envelop is shaped in time. Vosim66 has several pre-defined envelops at its disposal. The original, basic contour (Vosim 7) is *linear interpolation*. We have added exponential interpolation, peaks and smooth curves. Using special curves give you the possibility to make a sound of only one single segment instead of three segments. Many sounds can be created in this simple manner.

To control the amplitude-values we use A_1, A_{1end}, A_2 and A_{2end}. These four parameters must be between -100 and +100. Zero is silence. Negative values give you the

possibility to turn the phase 180 degrees. So for example when we use A=20 and A_{end}=80, with a linear interpolation, the amplitude will change from 20 to 80 over the length of a segment (LP). The amplitudes of the two generators can be controlled separately. Generator 1 with the keys U and I in the editor, for generator 2 with the keys N and M.

The following amplitude envelops are built-in and can be changed with the key-sequences: uuuu and nnnn, which open the following menu:

```
 -------------------------
 Amplitude envelop mode gen. 1
 -------------------------
*0.Mode 0: Linear change of amplitude from begin to endpoint
 1.Mode 1: Exponential decay from beginpoint towards silence (time-dep. slope!)
 2.Mode 2: Smooth change of amplitude from begin to endpoint (begin/end slope=0)
 3.Mode 3: Short peak between min and max amplitude with exp. decay (time-dep. slope!)
 4.Mode 4: Smooth peak between 0 and max amplitude and back

 Enter:
```

Mention that some of these envelops do not start in A=50 and/or end in A_{end}=15. (See figure 4). Mode 0 and 2 go from begin to end. Mode 2 starts at the beginning and goes down exponentially. Mode 4 starts at 0, climbs to the maximum and drops down exponentially. Mode 5 starts and ends at 0, with its peak in the given maximum.

Mode 0 is the original linear interpolation. Mode 1 and 3 are suitable for all kind of instruments with a natural damping, like bells, gongs, percussion instruments etc. Mode 2 can very well be used for creating scapes with slowly and smoothly moving, but continuous amplitude. Mode 4 is thought for smoothly starting and ending sounds with a constant body, like for example an organ, a flute etc.

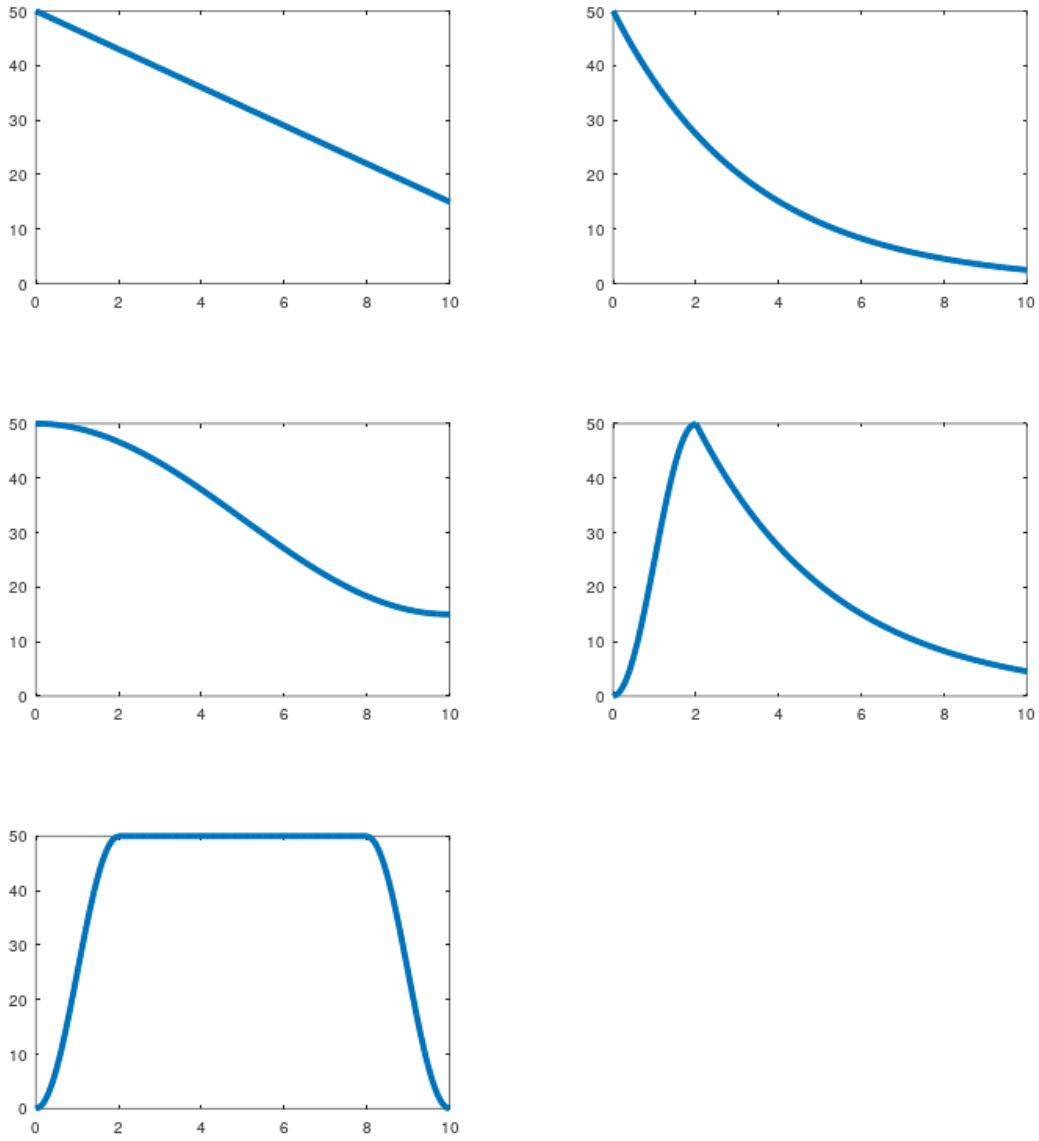


Figure 5 The five available amplitude envelops (mode 0 to mode 4)

8.5. Number of pulses (H and J) and Decay (HH and JJ)

As we explained earlier (chapter 3), the Vosim signal consists of a series of equal sin-square pulses, followed by silence. The numbers of pulses N1 and N2 are of strong influence on the spectrum. Very roughly we can say that higher values for N create stronger overtones. The same holds for the decay-value, which is expressed in %.

100% means no decay. The four parameters: N_1, N_2, D_1 and D_2 can be controlled with the H and J-keys in the editor.

8.6. Prosodic parameters and duration

In 8.1 we talked of pitch. We then suggested, for the sake of simplicity, that pitch is part of the sound concept we are working on. For some sound that's certainly correct, but in music it makes more sense to see pitch as variable. Something you can choose, apart from your instrument. These kinds of parameters, like pitch, duration and articulation are called *prosodic parameters*. Designing an instrumental sound means that you can vary these parameters without destroying the sound quality. In the next chapter we will go deeper into this matter. For now our second prosodic parameter is duration of the segment. You can control it with L on your keyboard. In Vosim66 we generally suppose that both generators use the same length in one segment. If you want to overlay sounds, use layers. To deviate from this rule, you have an extra menu at your disposal with LLLL.

```
 --------------------------
 Mode for Duration LP
 --------------------------
*0.Lmode 0: LP2=LP1
 1.Lmode 1: Edit only LP1
 2.Lmode 2: Edit only LP2

 Enter:
```

8.7. Modulation, vibrato and noise

Modulation is from the very beginning of Vosim an important part of sound design. In the Vosim 7 standard however only frequency modulation (FM) was available. In Vosim66 we added amplitude (AM) and pulse modulation (PM). All of them have their own specific sound quality and therefore application. For all of the three you can also choose random/noise modulation.
How diverse the use of modulation is in sound design, can we demonstrate in the following examples:

- Vibrato of singers is between 4 and 10 Hz and of course periodic.
- Bells and gongs produce many strange extra modulations, which are often non-harmonic and develop strongly in the acoustic space.
- Gurgling of water consists of randomly changing formant frequencies.
- A flute makes a lot of noise, created by the airflow in the mouthpiece.

The following modulation modes are implemented in Vosim66 and can be controlled by the z and x-keys (generator 1 and 2). The first six modes are FM, AM and PM with periodic(sin) resp. random modulation. Nr. 7 to 10 are mixed modes.

```
 -------------------------
 Modulation Mode Gen. 1
 -------------------------
 0.No modulation
 1.Noise/Pitch
 2.Sin/Pitch
*3.Noise/Amplitude
 4.Sin/Amplitude
 5.Noise/Puls
 6.Sin/Puls
 7.Noise/Pitch+Puls
 8.Sin/Pitch+Noise/Puls
 9.Noise/Pitch+Sin/Puls
 10.Sin/Pitch+Puls
 11.Subharmonics
 12.Subharmonics+Noise
 13.Noise old Vos
 14.Sin old Vos

 Enter:
```

Periodic modulations have two more parameters to control. Modulation depth D_{mod} (a and s-keys) determines how strong the modulation is and is expressed in % (of the actual period or amplitude in the signal). The second is the modulation frequency F_{mod} (q and w-keys) in Hz.

The subharmonic modes (11 and 12) use for F_{mod} the number of the subharmonic. F_{mod}=2 means an extra component at $f_{sub}=f_0/2$ (Vosim7).

Modes 13 and 14 are used for imported Vosim7 material. Mode 13 is the noise modulation, 14 the periodic modulation.

8.8. Synchronization of generators

Obviously it's very important that the two generators in Vosim work together properly in time. Thus from the very beginning of Vosim, synchronization was introduced. Originally of course in the hardware as a trigger to wait. Unless the need for additional synchronization has changed since then, Vosim66 still uses some forms of synchronization.

Synchronization is particularly necessary in cases where high accuracy is required, for example with rapidly changing signals. In the Vosim7 vector there was also an inherent need for synchronization, because time was determined by number of periods. When pitch is different for the two generators, the number of periods is also different and they very often do not fit together. In the Vosim2020 standard this problem has been overturned by the time base and internal corrections to make the calculations as precise as possible.

Synchronisation can be controlled by oo in the editor and has three modes:

```
 --------------------------
 Syncmode
 --------------------------
 0.Mode 0: No synch.
*1.Mode 1: LP2=LP1
 2.Mode 2: Syncpoint

 Enter:
```

In the first mode (0-mode), the generators function independently. It means that different durations for the two generators can be used, although we do not recommend it in most applications. Also small calculation deviations will not be corrected.

In the second mode (1-mode), the length of the segments for generator 1 and 2 is always the same. What's more deviations will be corrected in the signal, by adding silence to the shortest track. Special attention is needed when you use this mode in combination with moving formants with high Q-values (around 1). The signal then approaches a sine-wave. In the case consecutive segments with continuous amplitude values are used, there can be clicks between the segments. These clicks are created

by the synchronisation algorithm, which fills differences between the tracks with silence. In this case the 0-mode is advised.

The third mode (2-mode) gives you the possibility to sychronize the generators at a certain point in the buffer. The synch point will always be at *the end* of the segment.

In the development of Vosim66 there were many different attempts to develop a synchronization that satisfies in all situations. After several failures, it was time to rethink the basic concepts. There was a growing conviction that synchronization should not actually be necessary, when the timehandling is really precise. This resulted in a quite pragmatic, but als very effective solution. Instead of controlling the durations for *each* segment before composing them, we choose to control the *complete* timebase from the start of a soundsequence till the actual segment. In Vosim66 deviations of the length of one segment are immediately corrected in the next segment. This means that these deviations never can accumulate over longer sequences and that not only generators work together, but also different layers are synchronous.

8.9. Layers

In the old Vosim and Midim system there were only two synchronized Vosim generators available. In the MIDIM system there were basically two different modes in which the two generators were used:
1. Linked, to create sounds with two formant structures.
2. Unlinked, to create two different musical lines, each with one generator.
Of course, in the second case pitches were independent and synchronization was not always needed. The possible sounds were more limited, because only one signal was used per line.

When designing the new Vosim66 system, we decided to change this old duality. We define a segment as two coupled Vosim generators (never one), creating one combined spectrum with two formants. We think it's more in the sense of the

theoretical basis, namely that all sounds (in the minimal description) need two formants, like language sounds. Of course, you still can use amplitude zero on one generator, if you like. In this definition of segments, unlinked pitch is rare. (Technically it's possible, because it was necessary to play old files). That's the reason why in the editor, you only can control one pitch for both generators. Of course, you can link the pitches with link mode 2 or 3.

To replace the second application of the old system, namely to make separate musical lines, we have introduced layers. The Vosim66 contains four layers of two generators each. Of course, layers are played synchronously. The layer number 1-4 is stored in the Vosim vector (first generator, vector position 1) and can be adjusted in the editor with zz. Mention that segments of different layers are not stored in different buffers, but just can be mixed in one buffer. Here we show you an example:

```
****************************************** Input buffer (p.1/4) ******************************************

          Layer  Formant End     Pitch   End    Ampl. End  Decay  N  ---Modulation---   LP  Sync Link
          EdMode (Hz)    (Hz)    (Hz)    (Hz)   (%)   (%)  (%)       Mode Depth Freq.   (s)  QP  Instr.

Segment 1 ------------------------------------------------------------------------------------------------
            1     0.0     0.0   311.0   311.0   100   100   100   1     1    5  0.1   1.009   1.0   2
            1  6010.0  6010.0   311.0   311.0     3     3   100   2     1   40 10.0   1.009   7.0   0
Segment 2 ------------------------------------------------------------------------------------------------
            1     0.0     0.0   440.3   440.3   100   100   100   1     1    5  0.1   1.287   1.0   2
            1  6010.0  6010.0   440.3   440.3     3     3   100   2     1   40 10.0   1.287   7.0   0
Segment 3 ------------------------------------------------------------------------------------------------
            1     0.0     0.0   414.9   414.9   100   100   100   1     1    5  0.1   0.961   1.0   2
            1  6010.0  6010.0   414.9   414.9     3     3   100   2     1   40 10.0   0.961   7.0   0
Segment 4 ------------------------------------------------------------------------------------------------
            2     0.0     0.0   329.3   329.3   100   100   100   1     1    5  0.1   1.090   1.0   2
            1  6010.0  6010.0  2305.0  2305.0     3     3   100   2     1   40 10.0   1.090   7.0   0
Segment 5 ------------------------------------------------------------------------------------------------
            2     0.0     0.0   465.1   465.1   100   100   100   1     1    5  0.1   1.271   1.0   2
            1  6010.0  6010.0  3255.4  3255.4     3     3   100   2     1   40 10.0   1.271   7.0   0
Segment 6 ------------------------------------------------------------------------------------------------
            3     0.0     0.0   329.5   329.5   100   100   100   1     1    5  0.1   1.853   1.0   2
            1  6010.0  6010.0  2306.3  2306.3     3     3   100   2     1   40 10.0   1.853   7.0   0
Segment 7 ------------------------------------------------------------------------------------------------
            3     0.0     0.0   439.5   439.5   100   100   100   1     1    5  0.1   1.331   1.0   2
            1  6010.0  6010.0  3076.2  3076.2     3     3   100   2     1   40 10.0   1.331   7.0   0
Segment 8 ------------------------------------------------------------------------------------------------
            1     0.0     0.0   348.8   348.8   100   100   100   1     1    5  0.1   1.304   1.0   2
            1  6010.0  6010.0  2441.9  2441.9     3     3   100   2     1   40 10.0   1.304   7.0   0
```

In the first column on the first generator line, you can see the layers. Segments 1,2,3 and 8 will be put in layer 1. Segments 4 and 5 in layer 2. Segments 6 and 7 in layer 3. Layer 4 is still empty. So, the way this will be played is as follows

Layer 1	Layer 2	Layer 3	Layer 4
seg. 1	seg. 4	seg. 6	--
seg. 2	seg. 5	seg. 7	--
seg. 3	--	--	--
seg. 8	--	--	--

When you create sound sequences segment by segment, this representation of the buffers may be confusing, but in the Scape and Score Creator the layers are easily manipulated as tracks are in a sound editing systems. When saving sound, the layers will be exported as separate wav-files for further editing use.

Layers can be combined to create more complex sounds, to a maximum of eight generators working together. We will show you later how the complex sound mode works. We will show you how to make a quite realistic simulation of a cymbal sound. (chapter 13)

8.10. Damping and overlapping

In the world around us, many sounds overlap, sometimes just a bit, when they succeed each other. This is certainly the case in percussion sounds, which were often produced by different sound sources. For example in a piano every note has it's own independent strings to create the sound. The different notes thus overlap, certainly with the damping pedal used, but even when these are short notes, played in a high tempo and the damping pedal is released, there is an accoustic overlap due to resonances.

The Vosimgenerator in Vosim66 is designed in such a way that overlapping of sounds is fundamentally possible. Succeeding sounds are put together by addition, not by concatenation, so they can overlap.

Vosim66 has two different types of overlapping. A general overlapping of the segments can be applied for *all* the segments which end with amplitude zero. This overlapping can be switched on or off in the settings, where you can also determine the overlapping time. Use of a small overlap makes soundsequences often more natural.

A more *soundspecific overlapping* can be achieved by using certain amplitude envelopes in combination with a fixed damping time, as we hear in bells, celesta and chimes. The damping time can be adjusted with the variable t-demp, which has

different meanings in the different envelopes and can be controled with ui/uii and nm/nmm key combinations.

Amplitude envelop mode	Decay time t-demp is
0. Linear	Time between A and Aend
1. Exponential	Time between 100% and 5% of A
5. Peak+exponential decay	Time between 100% and 5% of A

When t=demp is 0, the damping time is the same as the segment length. (This is separate from the general damping mentioned.).

When t-demp > 0 the damping time is *fixed* and defined by the value of t-demp, which is part of the Vosim vector. For example, when you create a bell sound, t-demp could be 0.8 seconds. Every note then will sound over 0.8 seconds, even when the next tone starts f.e. 0.3 seconds after the first. In the case of exponential damping (in which theoretically the sound never reaches silence), the t-demp is defined as the time it takes for the signal to fall to 5% of its initial value.

9. Creating Sounds and Instruments

This chapter cannot cover the title in any respectful way. Because the field of how to make a sound or a sound concept, like an instrument is very diverse and depends a lot on everybody's style, knowledge and creativity. So, in this chapter we will discuss the tools Vosim66 delivers to shape your sounds.

9.1. The Sound Creator

The Sound Creator lets you create one segment by answering a series of organized questions. Deeper knowledge of the way the spectrum depends on the Vosim vector can help you a lot to understand the questions you have to answer in the Sound Creator.

9.2. Using libraries

It's our goal to create more and more useful material and cover a large field of different sounds and collect them in a library, out of which you can choose the sounds which are nearest to your imagination. We would like to encourage everybody to participate to this library by sending us interesting creations. At the moment you can load existing Vosim66 files to inspire you.

9.3. Instruments

In the Sound Creator you generally make one segment with two generators. One segment can make a perfect "instrumental" sound. For now we call it a "one-line instrument". You then

work with the system in it's basic "segment mode". (See editor modes 6.7) In this mode special series functionalities can be used easily to make scapes and scores with your segment.

But sometimes sounds need more shaping in time. In the MIDIM-language the segmented sound was a very important part of the concepts, which were incorporated in the predicator in the MIDIM system and theory. Kaegi stated that every sound concept must comprise four segments, the so-called *prefix p, body b, suffix s* and *stop st*. The idea is that many sounds have a head, a body, a tail and sometimes an empty space behind them, like little snakes. Kaegi also developed a detailed system of articulations, which consists of rules how different parts can be combined. In staccato for example the sequence p-b-s-st will be repeated in the notes. While in legato p-b-b-b-b-s-st is used.

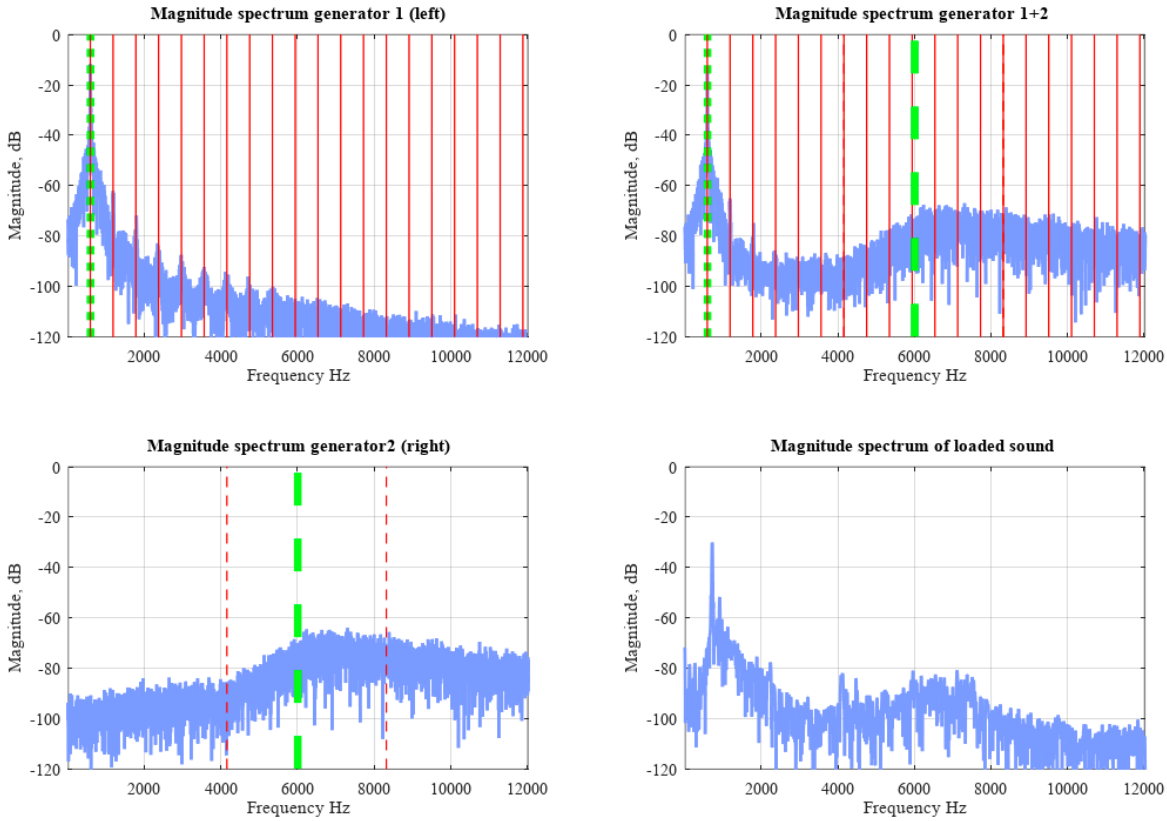
In Vosim66 we have adopted some of the original concepts and will implement some more later. A segmented sound concept we call "an instrument", which has always (in its basic form) three segments: prefix, body, suffix. (To make things easier, we skipped the "stop". The stop can be created later by inserting silences/rests.)

For using instruments, there is a special editor mode (xx for menu). Segments numbers 1-3, 4-6, 7-9 etc. are then always used for instruments and the corresponding parts of the snake. Mixtures of "one-line instruments" and "three-line instruments" in one buffer are not yet possible in Vosim66, but in general this creates no major limitations.

9.4. Spectra and reference sound

To manipulate sounds in an analytical way, spectra are indispensable. With the ?-key a pop-up window will appear with magnitude spectra for both generators and their sum. Here we show

you the spectra of the ocarina sound from ocarina6-online.vos, how it looks in the spectra of Vosim66.



On the left side generator 1 and 2 are shown separately. Top right you see the combined spectrum. Bottom right we have loaded a recording of a real ocarina. You can clearly see that our Vosim simulation is very close to the original.

To load a reference sound, as shown in this example, typ ??. Then you will be asked to enter a wav-file. Mention that this recording shouldn't be longer than some seconds. The sound must be clear and stable. The sampling rate should be the same as you use in the Vosim66 settings.

```
[1/1] => ??

Do you want to enter a sound (wav-file) for spectral comparison?(yes or no) yes

***Audio has been loaded and spectrum can be shown

[1/1] => ?
***Spectrum has been plotted in seperate window
   Green line is formant, Red are harmonics
```

In the spectra there are some help lines plotted, red for harmonics, green for the formants.

To play the calculated Vosim sound you can use the r-key, to play the recorded sound, you can use rr. This makes a quick comparison of the sounds and the spectra very easy.

10. The Scape Creator

Soundscapes in Vosim66 are patterns of sounds. These patterns can be melodic, harmonic, rhythmic, continuous, repetitive, musical, noisy etc. Scapes as we see them in this context are definitely not created with a fixed series of notes in mind. In a scape you can start with some vague ideas about pitch, duration and loudness and fill in all uncertainties with random values. Think of bubbling water or rushing winds. In the first you can certainly hear a rhythm or pattern of rapidly varying tones. In the latter noise will be the main component, but subjected to the slow wave of howl, singing or whistling. All these kind of patterns can be designed in the scapes creator. But also more "musical" patterns can easily be made with the Scape Creator. Think of a very repetitive drumline or a kind of "aleatoric music" in which only specific notes should appear.

10.1. Defining ranges

When designing a Scape, we have to start with a sound of minimum one segment in the input buffer. The new Scape will be created from this input segment(s) and stored in the output buffer as a pattern. This pattern consists of variations of the starting sound.

To transform your input segment into many output segments ("The Scape"), there are several parameters at your disposal. Think of duration, pitch, amplitude etc. To define a scape we have to choose for every parameter a range of values and the way the values can be chosen by the system. Lets illustrate this with the values of the duration of the segments of your scape. Suppose we have chosen values between 0.5 and 1 second with random fill-up. When the system now creates 20 segments from the sound in the input buffer and stores them in

the output buffer, these 20 segments will all have lengths between 0.5 and 1 second, randomly chosen. In this way we can define all necessary parameters of a Scape.

10.2. The Scape Menu

To enter the Scape Creator, you choose 8 in the main menu, then choose 1 in the Series Menu. The Scape Menu is shown when you enter the Scape Creator.

```
SCAPE CREATOR
-------------
Scapes are filtered from the input buffer and stored in the output buffer
Input: 12 seg. Output: 838 seg.

1.Load scapes
2.Edit scapes
3.Recalculate scapes
4.Save scapes
5.Back to main menu

-->
```

With the option Load Scapes you can load a scp file, which contains the information of one or more saved scapes. The second option opens the Scape Editor, which we will dicuss in detail below. Option three gives you the possibility to recalculate the output segments with the Scape information. Mention that the final result can vary, due to the different random values, used for calculation. Option four saves the actual scapes which are stored in the scapes buffer to a file. In the case you save a complete project, your Scape definitions will automatically be saved to the same project folder, together with input and output buffers.

10.3. The Scape Editor

In the Scape Editor you can either create new Scapes or change already created or loaded scapes. When your Scape buffer is empty, the editor will show you default parameters for the first scape. Here an example.

```
SCAPE EDITOR
------------
Actual scape nr.: 18 (new skape)

Layer 1:   462 segments 160.0 seconds   Scapes: 1   6   7  12  13  14  15  17
Layer 2:   204 segments 120.0 seconds   Scapes: 2   4   8   9  16   0   0   0
Layer 3:   172 segments  80.0 seconds   Scapes: 3   5  10  11   0   0   0   0

1.Output mode (Append new Skape to Layer 1)

-----------------------------------------------------------------
2.Duration of segment (Duration Range: 0.05 - 0.23 in seconds Rhythm: 0)
3.Durationmode (Mode: Random Humanizing: 3%)

4.Pitchrange (Subdivision: 12 Octave: 4 - 5 Pitch: 1 - 12 Scale: 0)
5.Pitchmode (Mode: Random Not Connected Humanizing: 2%)

6.Amplituderange (From: 90 To: 100)
7.Amplitudemode (Random Not Connected Silence: 20%)
8.Begin/End (Beginsilence: 3 s Fadein: 9 s Fadeout: 8 s)

9.Source segment (Only nr.: 1 )
10.Total length of scape (20 seconds)
-----------------------------------------------------------------
11.Create scape and play
12.Replay
13.Edit input buffer
14.Edit scales and rhythms
15.Show buffer
16.Recalculate all scapes
17.Exit scapes editor

-->
```

You can see that the editor is on scape number 18 (Actual scape), which is a new (i.e. not yet created) Scape. There are already 17 Scapes in the buffer. Top the right of the layer information you can see how these 17 Scapes are ordered and assigned to the layers. Layer 2 for example consists of the scapes numbers 2, 4, 8, 9 and 16. Together they consist of 204 segments of 120 seconds.
The options 2 to 10 give you the possibility to change the ranges of the parameters, used when the new Scape 18 will be created. We shall dive into these options later.

To really create the new Scape, you need option 11. The new segments will be added to layer number 1, as you can see in the information behind option 1.
(`1.Output mode (Append new Skape to Layer 1)`). The output buffer will be recalculated to play the sound. You can see the new scape no. 18 right to scape 17.

```
Layer 1:    542 segments 240.0 seconds   Scapes: 1   6   7  12  13  14  15  17  18
Layer 2:    204 segments 120.0 seconds   Scapes: 2   4   8   9  16   0   0   0   0
Layer 3:    172 segments  80.0 seconds   Scapes: 3   5  10  11   0   0   0   0   0
```

To replay the already calculated sound, you can use option 11. In option 12 you can edit the input segments, to change the original soundmaterial you use for the scape. Of course this will only affect new Scapes. (To change the sound input of existing Scapes you have to recalculate all Scapes with the recalculate option in the main menu of the Scape Creator.)
Let's now discuss the meaning of the list of parameters, which you can see in options 1 to 10 in the editor.

10.4. Output Mode

In the output mode option, you determine first if you want to change an existing scape or if you want to create a new scape. When you enter the option, a list of all existing scapes will be presented, with their layer, length in seconds and the number of segments.

```
Existing Scapes
---------------
Scape: 1 Layer: 1 Length(s): 20 #Segments: 50
Scape: 2 Layer: 2 Length(s): 4 #Segments: 3
Scape: 3 Layer: 3 Length(s): 4 #Segments: 5
Scape: 4 Layer: 2 Length(s): 16 #Segments: 20
Scape: 5 Layer: 3 Length(s): 16 #Segments: 40
Scape: 6 Layer: 1 Length(s): 10 #Segments: 25
Scape: 7 Layer: 1 Length(s): 10 #Segments: 25
Scape: 8 Layer: 2 Length(s): 10 #Segments: 51
Scape: 9 Layer: 2 Length(s): 10 #Segments: 50
Scape: 10 Layer: 3 Length(s): 20 #Segments: 26
Scape: 11 Layer: 3 Length(s): 40 #Segments: 101
Scape: 12 Layer: 1 Length(s): 10.0014 #Segments: 41
Scape: 13 Layer: 1 Length(s): 10 #Segments: 80
Scape: 14 Layer: 1 Length(s): 10 #Segments: 80
Scape: 15 Layer: 1 Length(s): 10 #Segments: 81
Scape: 16 Layer: 2 Length(s): 80 #Segments: 80
Scape: 17 Layer: 1 Length(s): 80 #Segments: 80
Scape: 18 Layer: 1 Length(s): 80 #Segments: 80

Output mode:
 0=Change/overwrite scape
 1=Create/append new scape
```

Then you can choose if you want to change (i.e. overwrite) or create new (and append).
In the first case Vosim66 will ask you which scape you would like to overwrite and if you want to assign this scape to another layer.

```
Overwrite existing scape nr.:  3
Change layer?(yes or no) no
```

As you can see, scape nr. 3 is put into layer 3. After overwriting the scape will still be in layer 3.
In the case you want to create a new scape and add it to a layer, the system will ask you the layer number to which the new scape will be appended. If you want to make a copy of an

already existing layer, or want to make some changes, based on an existing layer, you can use another layer as "template".

```
Output mode:
 0=Change/overwrite scape
 1=Create/append new scape 1

Append new scape in layer number (0=next layer):  4

Template from layer nr.? (0=default):  3
```

10.5. Time duration of the segments

In option 2 ("2. Duration of segment") you can control the durations of the segments. You can enter a minimum and a maximum time in seconds. In the case these two are the same, the time duration af all segements will be equal ("Fixed duration"). When you choose a Rhythm number other than zero, the parameters minimum and maximum length have no effect. The Rhythm number refers to Rhythms defined under option 13, which we will discuss in paragraph...

In option 3 ("3.Durationmode") you can define the so called duration mode, which determines how the durations are chosen.

duration mode	Rhythm = 0	Rhythm > 0
1. random	Durations are chosen randomly from the duration range (min to max)	Durations are chosen randomly from the in the rhythm defined durations
2. wave	not operational	not operational
3. pattern	??	Durations defined in the rhythm are used in the defined order

In random-mode all values between minimum and maximum length are chosen by the program. The wave mode is not operational at the moment. In patternmode, the durations defined in the Rhythm buffer are used in the defined order.

Humanizing is the random deviation which is applied to the durations in % of a quarter note.

10.6. Pitch of the segments

In options 4 and 5 you can control the way the Pitch is created in your new segments.

```
4.Pitchrange (Subdivision: 12 Octave: 3 - 3 Pitch: 9 - 12 Scale: 0)
5.Pitchmode (Mode: Wave Connected wave1: 0.4 wave2: -0.9 Humanizing: 0%)
```

Option 4 gives you the possibility to choose Subdivision of the octave (standard is 12), the actual octaves used, the pitches and the scale. The Pitch mode defined in option 5 has three values (1=random, 2=wave, 3=pattern).

There are three pitch modes available. The first mode is called random, which means that in this case the pitch values of the tone of every generated segment will be taken out randomly of Octaves 3,4 or 5 and the pitches 1 to 12 (See note range in the menu).

pitch mode	scale = 0	scale > 0
1. random	Pitches are chosen randomly from the pitch and octave range.	Pitches are chosen randomly from the in the scale defined pitches.
2. wave	Pitches are chosen in a waving flow from the pitch and octave range.	Scale has no effect.
3. pattern	Climbing scale is generated from pitch 1 to 12 in the octave range.	Pitches are chosen in order from the in the scale defined pitches.

In the mentioned menu you can see an option "Connection" yes/no. When you choose yes, the tone of a segment will change during the segment and adapt to the next segment. The result gives you a completely different sensation. With no connection you hear different notes, with connection you hear one continuous flow of a changing tone.

The second pitch mode is called wave. Vosim66 can create a waving pattern of pitches between the given range. This pattern has two factors, wavefactor1 and 2, which determine the speed and waveform. Values between 0 and 3 are good for many applications.

The third pitch mode without scale gives you a climbing tone scale or creates a pattern as defined in the scale. The first is especially useful to investigate the behavior of your source

segment in different tone registers. When you add a defined scale, you can generate a melodic pattern, which will be repeated.

10.7. Defining Scales and Rhythms

Creating a score means organizing melodic and rhythmic patterns, usually defined by notes, which combine pitch and duration for every instance. In the Scapes Creator the use of patterns is much more free. In option 13 of the Scapes Editor you can define Scales and Rhythms. Here an example of four different Scales

```
Scale 1.  1  2  3  4  5  6  7  8  9  10  11  12
Scale 2.  1  4  8  11
Scale 3.  1  4  8  4  8  8  11  25  4  1  1  4  1  1  11  8  8  11  8  8  25  1  1  4
Scale 4.  1  8  10  4  2  1  4  12  8  4  1  10  8  4  1  12  20  8  4
```

The first is just all the pitches in one octave. The second consists of c-e-g-b. The third is already more ordered into a pattern: c-e-g-e-g-g-b-c#-e-c-c-e-c-c-b etc. You can apply these scales randomly or ordered.

To add random notes in a pattern, you can use -1, which inserts a random note of the range 1-12. If you want more control over the notes the system can choose, -2, -3 etc. can be used, which refer to the number of the scale. For example:

```
Scale 1.  1  2  3  4  5  6  7  8  9  10  11  12
Scale 2.  1  8  13  1  8  13  1  -3  13  1  8  13
Scale 3.  4  8  10
```

The -3 in Scale to refers to Scale 3, where the system chooses at random a note from and inserts it at the place of the -3 in Scale 2.

The Rhythms work in the same way. Number 1 means one beat (quarter note), in a metronome 60 number 1 is 1 second. 0.5=minim, 0.25=crotchet etc. When you enter the notes you can use ratios: 1/2=minim, 1/4=crotchet, 1/3=triplet etc.

```
Rhythm 1.  1  0.5  0.333333  0.25  0.2  0.125  0.0625  0.03125
Rhythm 2.  0.25  0.25  0.25  0.125  0.125
Rhythm 3.  0.166667  0.166667  0.166667  0.25  0.25
Rhythm 4.  0.0833333  0.0833333  0.0833333  0.125  0.125  0.25  0.125  0.125
```

10.8. Amplitude range and mode

In the same way as we can shape pitch, also amplitude can be handled. When we choose an amplitude range, for example between 20 and 100 %, the editor offers you an option 5 amplitude mode.

```
4.Amplituderange (From: 20 To: 100)
5.Amplitudemode (Random Not Connected Chance of Silence: 0)
```

You can choose between random and wave. You have the possibility to connect the values smoothly and you can add rests (silence) , by entering the chance of occurrence (Values between 0 and 1. The value 0.5 for example givens you 50% chance that a new segment will be silent.).

```
Amplitude of segments is Random and Not Connected
Amplitude mode (1=random,2=wave)=  1
Connection (0=no,1=yes)=  0
Chance of having a rest=  0.4
```

In this example we have chosen the chance of silence 0.4, which means that 4 out of 10 notes will be a rest.

10.9. Silence at the beginning, fade-in and fade-out

```
8.Begin/End (Beginsilence: 3 s Fadein: 9 s Fadeout: 8 s)
```

In option 8 of the menu, you find three functionalities: beginsilence, fade-in and fade-out. Sometimes you don't want a new skape to begin at the point where you are in the layer. The option Beginsilence makes it possible to start a new skape with silence.
For long atmospheric layers, the fade-in and fade-out functions are very useful. The fade-in starts after the Beginsilence. The fade-in and fade-out are added to existing amplitude settings, as described in 10.8.

10.10. Number of source segment

As we mentioned earlier, you first need to prepare your raw material in the input buffer. It can be one single segment, but you can also start with several segments. Let's say you have three segments in your input buffer. Now the Scapes Creator needs to know which segment(s) you want to use. This can be defined in option 9.

```
Range of input segments is now from: 1 to: 1

First segment nr.=  1

Last segment nr.=  3
```

The system will now choose randomly between segments 1 to 3 from your input buffer. Of course in many cases you just like to use one single segment.

10.11. Length of the complete Scape

In option 10 you can enter the length of the scape as number of notes/segments or you can enter the length in seconds. When you want to use several layers, it's practical to define a certain length for all of them.

```
Fillmode for the scape:
 1=Number of notes
 2=Total length in seconds. 2

Length of the complete series (sec)=  20
```

In this case your scape will be approximately 20 seconds. Mention that the precision of the delivered time depends on the length of your segments, defined at option 1, and the total length. Segments will not be cut into pieces, so they have to fit into the total length. But for scapes in general the precision of the length of the layers is not critical.

10.12. Scape to Output

In option 8, finally, you can define how the newly created segments will be added to the layers and the already created material in our output buffer. The option overwrite always cleans the entire output buffer first before putting the new segments into the mentioned layer.

```
-->  8

Fillmode outputbuffer:
 0=overwrite
 1=append  1

Layer number (0=next):  1
```

The option append, means that the segments will be added anyway at the end of existing segments in the indicated layer.

10.13. Example 1 Aleatoric ocarina music

Now let's use your ocarina segment in the input buffer. Try yourself to fill in the following data:

```
SCAPES EDITOR
-------------

1.Duration of segment (From: 0.1 To: 0.5 seconds)
2.Noterange (Subdivision: 12 Octave: 4-5 Pitch: 1-12)
3.Pitchmode (Random Not Connected)
4.Amplituderange (From: 90 To: 100)
5.Amplitudemode (Random Not Connected Chance of Silence: 0.3)
6.Source segment in input buffer (Onlynr.: 1 )
7.Total length of scape (30 seconds)
8.Scape to output buffer (Overwrite Layer 1)
9.Apply and play
10 Quit editor
```

Then choose option 9 to create the scape, calculate and play it. You will get an interesting track of 30 seconds aleatoric ocarina music.

To make it more interesting, change some values and repeat the procedure with option 8: append and Layer 2. A second voice will be added to your first track.

To inspire you, here the next data used for the second layer could be as follows:

```
1.Duration of segment (From: 0.4 To: 1 seconds)
2.Noterange (Subdivision: 12 Octave: 4-5 Pitch: 1-12)
3.Pitchmode (Random Not Connected)
4.Amplituderange (From: 90 To: 100)
5.Amplitudemode (Random Not Connected Chance of Silence: 0.5)
6.Source segment in input buffer (Onlynr.: 1 )
7.Total length of scape (30 seconds)
8.Scape to output buffer (Append Layer 2)
9.Apply and play
10 Quit editor
```

You can also see, that there appears some information under the title, to show you how the layers have been used so far.

```
SCAPES EDITOR
-------------
Layer 1: 100 segment 30.0509 seconds
Layer 2: 43 segment 29.2154 seconds

1 Duration of segment (From: 0.4 To: 1 seconds)
```

10.14. Creating Scapes in a flow

In the main Vosim Editor, as we have discussed earlier, you can create a series of connected sounds, just by repeated copying and changing segments. The same method you can now apply on your scapes. Choose in option 8 append en then layer 0.

```
-->  8

Fillmode outputbuffer:
 0=overwrite
 1=append  1

Layer number (0=next):  0
```

Every time you apply the scape you have changed in your editor, the system chooses the next available layer. The first layers will be added in parallel till four layers are used, then new segments will be added after layer 1,2 etc. until the four layers are filled up again.

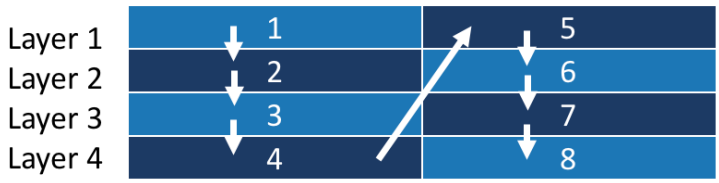


Figure 6 Automatic filling of the layers

10.15. Making scapes more alive

Making scapes is very tempting. It gives you in a short time interesting patterns and very often the results are quite surprising. Random patterns can appear more organized than you could believe. But still your raw material dictates the final result a lot. Try to make the Scape more dynamic by creating several segments in the input buffer first, which resemble each other a lot. Just copy them and change the modulation a little bit, for example. Then choose in your Scape a range of segments with option 6. The slightly varied segments in your input buffer will now be chosen randomly and give the scape more liveliness.

10.16. Using 3-segment instruments as material

Instruments in Vosim 66 are defined as groups of three segments, defining prefix, body and suffix. When you have switch to Instrument mode in the central Editor (see 6.7 Editor Modes), when preparing your material, or you have loaded an instrument made in an earlier stage, using the instrument mode, your Scape Creator will recognize the instrument and apply it as a triple. Of course, it makes less sense to use an instrument for creating a continuous Scape, so some options in the Scapes Editor, do not work for instruments. For example, you cannot use "Connected" pitch or amplitude.

10.17. Example 2 Wind Noise

10.18. The Scale mode

Sometimes you just want to make a scape on notes.

10.19. Saving and loading Scapes

11. The Score Creator

12. Spectra and simulations

13. Complex Sounds

14. Using sound libraries

15. Loading and saving

15.1. Vosim2020 files

The basic idea for Vosim66 was to keep things transparent and simple. The Vosim2020 file format as basic storage reflects this line and combine - as we explained earlier in this manual - several older concepts. The file stores a series of double precision numbers as represented in the segment buffers in an editable format.

15.2. Older Vosim formats

Later some other formats were added to make the software compatible with old MIDIM and intermediate Vosim7 formats. The most important difference between the older formats and the 2020 format is that we have replaced NP (number of periods) by LP (time in seconds). Tests showed that most of the original sounds can be recalculated with Vosim66, but there can be some small differences, especially in timing and synchronization. Commented files as defined in the Vosim convertor of Luuk Trip, can be read in 4 us and 1 us formats (T and DT-values). In these cases the first line will be seen as header. All other number lines are read in alternating pairs of Vosim7 vectors. Precise definitions of the various formats can be found in the appendix.

15.3. Old predicators

Predicators in the original MIDIM system were meant to define sound-concepts and not sound-events. Importing a predicator into Vosim66 is possible, but means that some conversion takes place and some prosodic parameters will be added. Moving formant (F1), link functions/values and modulation depths in cent can be correctly read.

15.4. Scapes and scores

To store or load a certain scape vector, there is a scp-file format. Scores can also be stored and read in the sco-type files.

15.5. Saving audio

Audio data can be saved after calculation in wav-format, corresponding the settings (bit rate and sampling rate, standard 16 bit, 44.1 kHz). When saving, there will always root and combined files available: stereo-file combining all layers and the two channels, mono-files for left and right channels and finally left and right tracks for every layer (maximum 4). The number of stored layers corresponds to how many you have used. The tracks are not normalized, but the levels of the combined tracks are adjusted by 3 dB for every additional layer. Mention that the system always saves the *last calculated* sound. The audio files can easily be dragged into an audio mixing software, like audacity and processed further.

15.6. Filters

Vosim66 uses two filters to make the calculated sounds more balanced. By aliasing created higher frequencies will be filtered with a FIR lowpass around 2 times the formant frequency. And finally, to make the output more standardized, every track goes through a dcblocker to remove the dc-component, which is inherent to every Vosim-signal. The filters can be turned on/off and adjusted in the settings menu.

15.7. Saving projects

To make working on certain project more easy you define a project name. (See) In the save menu you have the option to save you whole project: input buffer, output buffer, last calculated audio and the last score. The system will create a folder and puts all files together.

16. Restoring old Midimsounds

Vosim files cannot be transformed back to Predicators.

Sometimes there is a possibility to extract a predicator from a Vosim file. ?? Automatic extractor, for example for linkmode, Q values etc.

17. Appendices

17.1. Appendix 1 - Vosim 66 keyboard commands

Key *	1 time hit **	2 times hit ***	3 times hit	4 times hit
Q	Modulation Frequency Hz gene 1	“	“	
W	Modulation Frequency Hz gene 2	“	“	
E	Exit Editor			
R	Repeat last calculated sound			
T	Formant starting value Hz Gen.1	“	“	Moving form. on / off
Y	Formant endvalue Hz Gen.1	“	“	
U	Start amplitude% Gen. 1	“	“	Envelope mode gen 1
I	End amplitude% Gen. 1	“	“	
O	Output mode (mono / stereo)	Syncmode		
P	Print data or actual segment	Print all segments		
A	Modulation depth in% gene 1	“	“	
S	Modulation depth in% gene 2	“	“	
D	Modulation width Gen. 1	Mod. width Gen. 2		
F	Pitch at start in Hz or notes ****	“	“	
G	Pitch at end in Hz	“	“	
H	N number of pulses Gen 1	Decay in% Gen 1	“	
J	N number of pulses Gen 2	Decay in% Gen 2	“	
K	Linkmode	Linkvalue	Linkvalue	
L	Segment length in sec.	“	“	Change generator for input of duration
Z	Modulation mode gen 1	Mod mode gen 1	Change layer	
X	Modulation mode gen 2	Mod mode gen 2		
C	lc: Calculate segment and play uc: idem but 3 segments around act.	uc: Play instrument	uc: Play all segments	
V	Formant starting value Hz Gen.2	“	“	Moving form. on / off
B	Formant endvalue Hz Gen.2	“	“	
N	Start amplitude% Gen. 2	“	“	Envelope mode gen 2
M	End amplitude% Gen. 2	“	“	
+	Copy segment to end of list			
-	Delete actual segment			
<	Jump one segment backwards	Jump 10 backward		
>	Jump one segment forward	Jump 10 forward		

?	Show spectrum			
,	Start loop playing mode			
.	Stop loop playing mode			
=	Add instrument info			
1-8	Jump pitch to octave 1,2,3…, 8. Start of noteplaying mode. ****			
!	Stop noteplaying mode			

* All the commands should be in lower case basically. When changing values, lower case means decrease, upper case means increase.

** All input must be followed by CR (Enter / Open key on the keyboard)

*** Example of a repeated input: vv + enter decreases the formant frequency of generator 2 by 100 Hz. VVV + enter = decrease or formant 2 by 1000Hz.

**** The F-key has two different input modes. Normally it works on pitch in Hz. But when noteplaying is activated, the F-key jumps from note to note by semitones.

lc= lower case, up=upper case

17.2. Appendix 2 - Some thoughts on synchronisation

Technische synchronisatie

Musicale synchronisatie

Twee redenen om te synschroniseren in Vosim:

1. Nauwkeurigheid met name bij langere sequenties, waar kleine fouten optellen.

2. Verschillende lengte van de periodes in het geval van niet harmonische pitchverhoudingen tussen de twee generatoren.

Werner ziet synchronisatie zeker ook als een muzikaal belangrijk begrip, dat behoort bij een bepaalde esthetiek die de Componist hanteert. Zo zijn "stemmen" in de polyfonie synchroon, behalve bij de versieringen, waar juist een zeker mate van afwijkingen van de vaste organisatie in de tijd wordt gewaardeerd. Denk aan Chopin, die heel bewust probeert afwijkingen in de bovenstem te creëren. Denk ook aan de blue notes in de Jazz. Denk aan versieringen in de barokmuziek, die niet exact genoteerd worden en afwijken van de systeemnoten om de melodielijn te opspelen.

In het MIDIM-systeem en in het algemeen in de elektronische muziek vond Werner afwijkingen vaak juist welkom om het mechanische van algoritmisch geluid te doorbreken. Met name kleine afwijkingen maken het geluid levendiger.

Bij het gebruik van layers in de elektronische muziek zijn soms toevallige asynchrone mixen van verschillende lagen deel van de compositie.

17.3. Appendix 3 - Vosim vector standards

The old Vosim7 vector was defined by Werner Kaegi in his MIDIM/VOSIM reports (Interface vol. 15 Number 2-4 (1986).

Vosim 7 standard

Nr.	Code	Variable	Dimensions	Exceptions
1	T	Pulswidth (in 4us)		
2	ΔT	Increment of Pulswidth (in 4us)		
3	M	Delay (in us)		
4	ΔM	Increment of Delay (in us)		
5	D	Depth of Modulation		
6	A	Amplitude		
7	ΔA	Increment of Amplitude		
8	N	Number of Pulses		<0 is syncpoint between generator 1 and 2 at the end of the segment
9	C	Decay in %		
10	Sw	Modulation Switch		
11	Mf	Modulation Frequency		<0 Subharmonic (Mf is number of subharmonic)
12	NP	Number of Pulses		

When we started with the new Vosim system, which resulted after many versions in the Vosim66 version, there was an urgent need to define a new Vosim standard. This resulted in the Vosim 2020 vector. Although this vector is used in the software, it turned out not to be the best choice for the new needs. That led unfortunately to a somewhat chaotic use of the different fields. There are plans to make a future vector fully variable, allowing for further changes and/or additions.

Vosim 2020 standard

Nr.	Gen 1	Gen 2	Variable 1	Dimensions
1	Layer	Mode	Layer in output/ Editormode	Layers 1-4 / Editor 0=seg. 1=instr. 2=sound only on first line of buffer!!
2	F1	F2	Formant in Hz	10-20000 Hz, 0=Moving Formant Other=Fixed Formant, <0 Phase 180 deg. turned
3	F1end	F2end	Formant end in Hz	10-20000 Hz
4			Evt. Q-factor, new location	
5	F1switch	F2switch	Mode for example fixed/moving formant	Not in use yet, but to be designed
6	P1	P2	Pitch (in Hz)	10-20000 Hz
7	P1end	P2end	Pitch end (in Hz)	10-20000 Hz
8	Pmin	Pmax		
9				
10	A1	A2	Amplitude in %	0-100%
11	A1end	A2end	Increment of Amplitude %	0-100%
12	A1inp	A2inp	Extra variable for envelop modes	different values
13	A1env	A2env	Envelop mode over segment	0 to ? (in development)
14	C1	C2	Decay in %	From puls to puls in case N>1
15	N1	N2	Number of Pulses	Number of sin^2-Vosim pulses in pulsetrain
16	Nmax1	Nmax2		Not in use yet
17	D1mod	D2mod	Modulation depth	In % of T or T' for pulse and pitchmodulation, In % of amplitude in amplitude modulation
18	C1mod	C2mod	Modulation Mode	0-14, 0=no modulation
19	F1mod	F2mod	Modulation Frequency	in Hz 0-100 Hz
20	W1mod	W2mod	Modulation Width	In number of periods T' (0=standard) Only for Noise modulation
21	LP1	LP2	Duration of segment in s	
22	Q1	Q2	Q-factor moving formant Evt. new location for overlap time or duration modes	Q=(N*T)/T'
23	audioindex1/ kiki1	audioindex2/ kiki2	Audio index Generator3 Sound class/ MIDIMfunctions	Was Kiki (Not yet implemented), is now audioindex
24	sync	QP0	Sync mode/Link ratio or beat	Sync 0=no, 1=LP1eq.LP2 2=syncpoint QP=ratio or beat in link modes 2 and 3.
25	link	Number of scape	Link mode/Instrument mode	Link 0=no, 1=P2eqP1, 2=ratio 3=beat

17.4. Appendix 4 - Vosim66 generator properties

Wavetables

Samplingrates and bitrate

Layers

Corrections:

Duration correction

Pitch correction

Total times corrections

Modulation phase transition from segment to segment

Overlay mode

Micro correction to avoid stuttering caused by sampling rate

Calculation speed

17.5. Bibliography and links