PWGL LIBRARY

Jacopo Baboni Schilingi

JBS-PROFILE

jbs-profile
(v 1.0 based on old PW-Profile)

January 18, 2010

Contents

1 Start-Here **2**

2 0-Perturbations **2**
2.1 0-Perturbations . . . 2
2.2 1-Alea-Perturbation . . . 3
2.3 2-Control-Perturbation . . . 3
2.4 3-Compression-Expansion . . . 5
2.5 4-Hybridating-Profile . . . 6

3 0-Reflexions **7**
3.1 0-Reflexions . . . 7
3.2 1-Reflexion . . . 7
3.3 2-Double-Reflexion . . . 9
3.4 3-Multi-Reflexion . . . 10

4 0-Derivations **11**
4.1 0-Derivations . . . 11
4.2 1-Arithmetic-Derivation . . . 12
4.3 2-Mean-Derivation . . . 13
4.4 3-Average-Derivation . . . 14
4.5 4-Barycentre-Derivation . . . 15
4.6 5-Min-Max-Points . . . 16

5 0-Integrations **17**
5.1 0-Integrations . . . 17
5.2 1-Arithmetic-Integration . . . 18
5.3 2-Geometric-Integration . . . 19
5.4 3-Barycentre-Integration . . . 20

6 0-Interpolations **21**
6.1 0-Interpolations . . . 21
6.2 1-Dynamic-Interpolation . . . 22
6.3 2-Multi-Interpolation . . . 23
6.4 3-BPF-Interpolation . . . 24
6.5 4-Profile-Interpolation . . . 25

7 0-Utilities **27**
7.1 0-Utilities . . . 27
7.2 1-Note-Change . . . 27
7.3 2-Range-Approx . . . 28
7.4 3-Weight-Average . . . 29
7.5 4-BPF-Collector . . . 30
7.6 5-Four-Forms . . . 31

Box Index **33**

1 Start-Here

This documentation has been written by Jacopo Baboni Schilingi and Julien Vincenot. This is a library dedicated to control any kind of parameters you can represent into a bpf curve inside PWGL software.

(1) Perturbations

(2) Reflexions

(3) Derivations

(4) Integrations

(5) Interpolations

(6) Utilities

This library is the result of many years of work with Mikael Laurson, Nicola Evangelisti and Mikael Malt. It is based on Hyper-systemic theory I developed since 1987 till today. For those who might be interested, please refer to 'La musique Hyper-Systemique', Edition Mix. Paris 2007, by Jacopo Baboni Schilingi. For those who are familiar with the old PROFILE version, you must know that functions concerning pitches are still available. Nevertheless it is important to mention that this research started in 1994. In that period, I did not know Mikael Laurson's constraint system, that is why I created a sort of harmonic control for pitches. Nowadays it is totally ridiculous comparing to all the rules I've developed for Multi-PMC and Multi-Score-PMC. So for those who might be interested in pitch control for melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL. Particularly look at the patch numbers: 1.03, 1.04 and also 2.01 and 2.02.
To avoid any misunderstanding I called this library JBS-Profile to make a difference with the older one. Those who are familiar with PROFILE, will find in the JBS-Profile for PWGL all the ancient functions (sometimes with much more comprehensible names...) but also many new functions coming from my own personal libraries.
In future, you can freely download the always last upgraded version on my website www.baboni-schilingi.com Please, if you find some bugs or anomalies, write directly to me at jbs@baboni-schilingi.com.

2 0-Perturbations

2.1 0-Perturbations

All the functions grouped here can be applied to any kind of parameters that can be represented with numbers.
By perturbations I mean several functions able to change the initial state of a parameter, described by a curve. In this sense you need to imagine that you have a given profile. The perturbation menu gives you some functions that can keep the global original shape, but with some deep changes.

With the term of perturbation I also mean the manner in which a musical entity or any of its parts' form is changed as a result of functions applied by external elements.
ATTENTION : a lot of these examples are based on pitches. That does not mean at all that you cannot apply the same functions to other parameters.

2.2 1-Alea-Perturbation

1.1 - ALEA-PERTURBATION
This function creates a random perturbation by adding or substracting values between the limits. These limits are set in [a] with a single positive value.
If you put 10, for instance, the limits are automatically +10 and -10. The ALEA-PERTURBATION [1], for each value coming in the list input, will add a value chosen between -10 and +10.
This function is useful if you want to create perturbations of a given list.

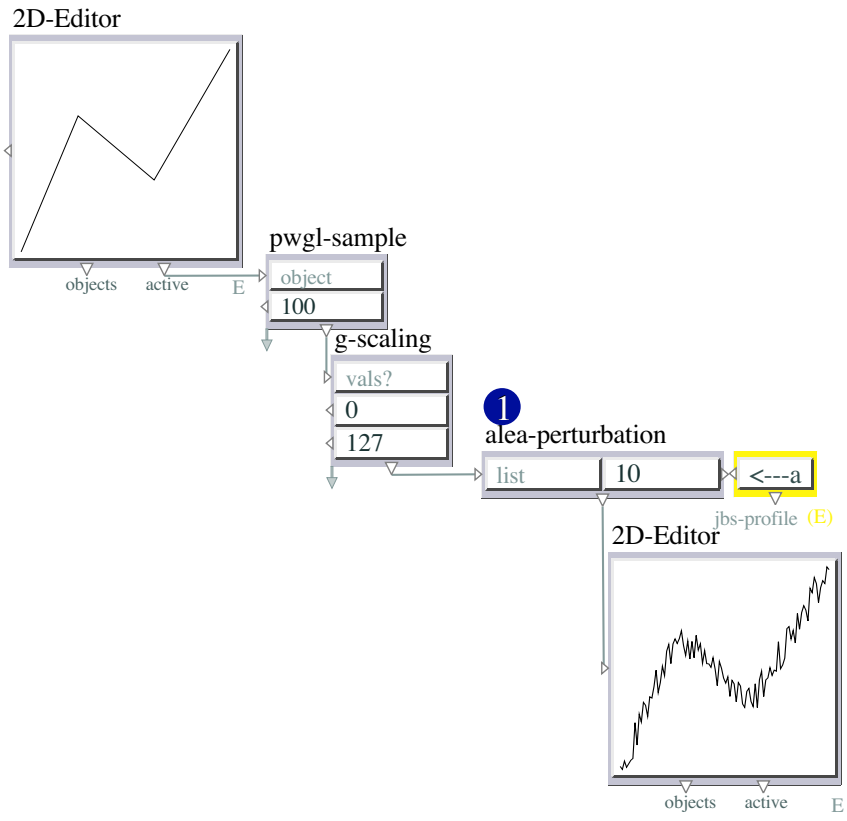

Figure 1: 1-1-alea-perturbation

2.3 2-Control-Perturbation

1.2 - CONTROL-PERTURBATION

This function is useful if you want to create perturbations of a given list, using a second list (or curve) in order to control them.
With [a] I prepared a list using a 2D-Editor. The values of this bpf are scaled between 0 and 1, and the number of points is set to 100. CONTROL-PERTURBATION [1] function accepts a list in the left input [c] and, in the right input, either a bpf object or a list. Use the PWGL-SWITCH [d] in order to choose directly the bpf or a list.
In the 2D-Editor [b] I prepared a second curve. CONTROL-PERTURBATION [1] will take for each element of the incoming list in [c] a correspondent element coming from [d].
If the index [e] is equal to zero, the bpf [a] is not perturbated at all. On the contrary the bigger the index is defined, the bigger the perturbation will be.
Technically the CONTROL-PERTURBATION [1] function adds the correspondent element of the second input [d] to the one set in the first list [a].
If in [a] you have (10 10 10 10 10), and in [b] (0 0 1 -1 0) and that the index is 1 the result will be : (10 10 11 9 10). If the index is equal to 100 the result will be : (10 10 110 -90 10).
Please open the two abstractions (the red ones) to see more examples.

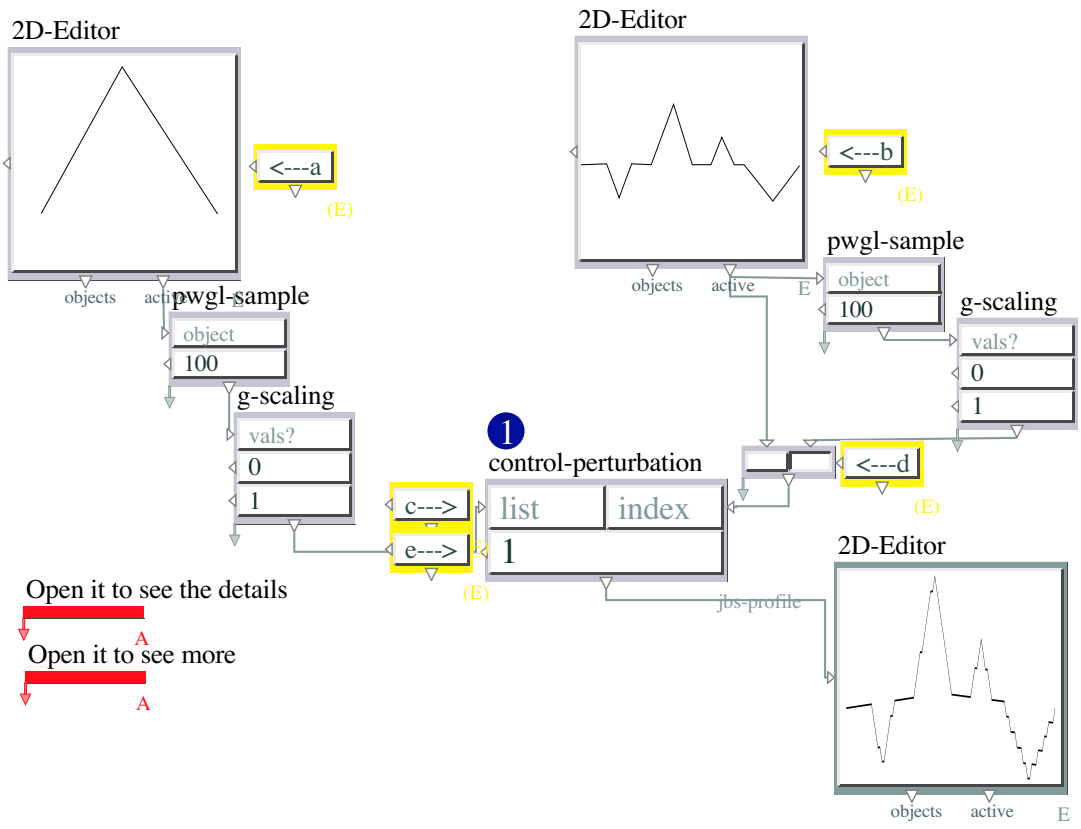

Figure 2: 1-2-control-perturbation

2.4 3-Compression-Expansion

1.3 - COMPRESSION-EXPANSION
This function creates compressions or expansions with the intervals of a sequence. In this example, intervals are defined by the pitches set in [a].
If COMPRESSION-EXPANSION [1] has a value of 1 in it [b] input, the list set in [a] is not changed at all. On the contrary if you put a number bigger than 1 in [b], the intervals will be more or less expanded. This value is the multiplier of the intervals.
If you put a number between 0 and 1 the intervals will be compressed. If you put a value between 0 and -1 the intervals will be compressed and also reversed (multiplication by -1).
If you define a value smaller than -1 (for instance -2, -3, -4...) the intervals will be expanded and reversed.
As compression and expansion are operated on the intervals, the first value (in this case the first note) is always kept.
Open the abstractions (the red one) in order to see more examples.
ATTENTION : Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL. Look particularly at the patch numbers 1.03, 1.04 and also 2.01 and 2.02.

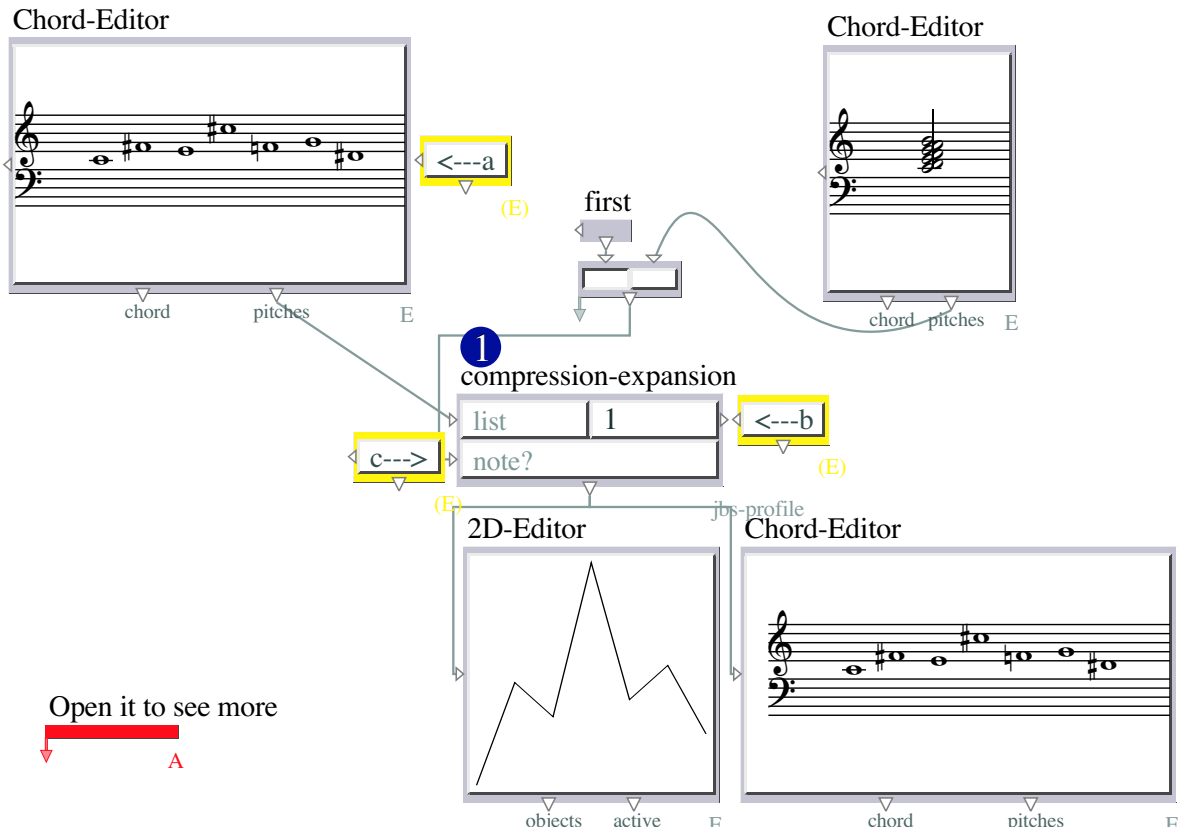

Figure 3: 1-3-compression-expansion

2.5 4-Hybridating-Profile

1.4 - HYBRIDATING-PROFILE
This function changes the directions of a second profile accordingly with the shape of a first one.
In [a] you enter some pitches having a given shape. In [b] you put the profile you want to change. The HYBRIDATING-PROFILE function [1] makes a comparison between the intervals of the two profiles. If the notes of the second one have the same directions of the first profile, these notes are kept as they are. On the contrary, if the notes have different directions from the given model, they are transposed by two criteria set in the menu [c].
If [c] is set on INTER, that means you want to keep the intervals of the second profile and these intervals will change directions but not their nature. For instance a fifth stays a fifth, an octave an octave, but their directions are changed.
If you choose NOTE in the [c] menu, you will keep the same pitches but transposed in order to change the directions accordingly to the first input.

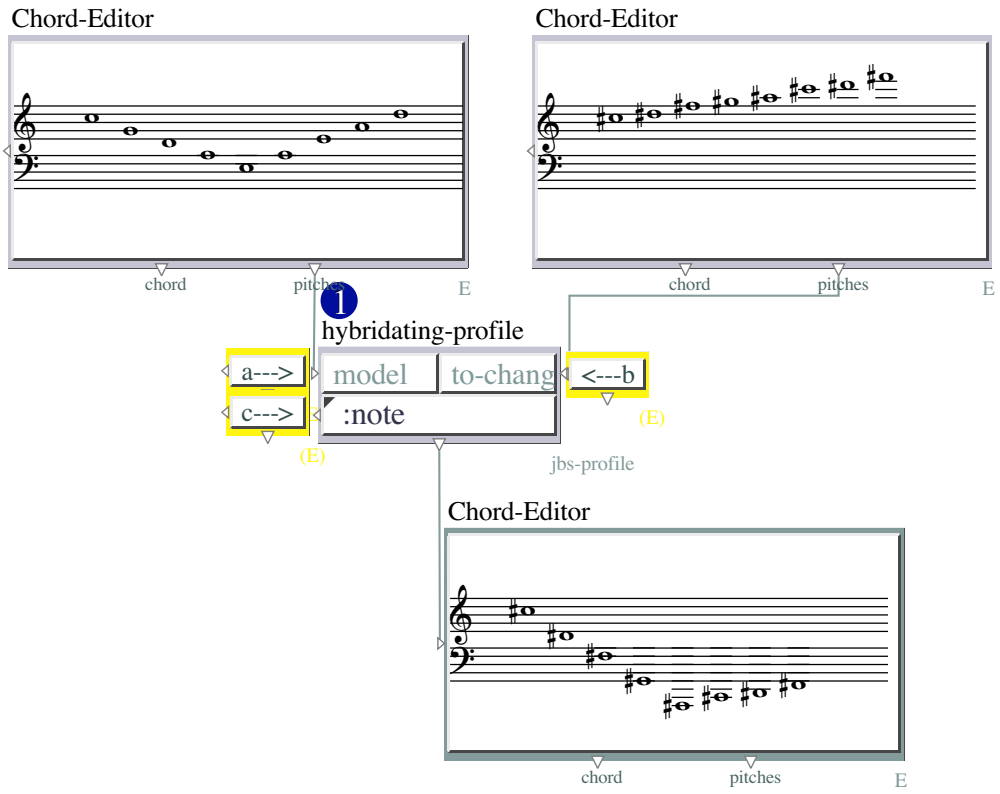

Figure 4: 1-4-hybridating-profile

3 0-Reflexions

3.1 0-Reflexions

All the functions grouped here can be applied to any kind of parameters that can be represented with numbers.
This part of the jbs-profile library is dedicated to the concept of geometric reflexion. The main idea is to create a sort of simulation of the geometrical reflexion around one or two axes.
As we have already seen in the first part of this library, the fact that a lot of functions here are based on pitches, that does not mean at all that these functions are limited to the pitch fields.

3.2 1-Reflexion

2.1 - REFLEXION
The REFLEXION function [1] creates an upper or lower reflexion of a given profile.

In the input [a] you define the profile you want to reflect, and in [b], you give the value of the reflexion. If in the fourth input the menu [d] is set on 'up', the function will create a reflexion changing the first profile in order to obtain only bigger values than the one put in [b]. If the menu [d] is set on 'down', the reflexion will produce values smaller than the one put in [b].
Example, the given profile: ———————— ——*———————— —*—*————*– -*— —*———*——— ————*——*———<<<<<<<<<<<<<< the axis ————-*—*———— —— ———*————- ———————— ————————

The upper reflexion
———————— ——*———————— —*—*—-*———-*— -*———*-*—*—*——— ———— *——*———<<<<<<<<<<<<<< the axis ————-0—0———— ————————0———— —— ———————— ————————

The lower reflexion:
———————— —0———————— —0—0—-0———-0— -0———0-0—0—0——— ———— *——*———<<<<<<<<<<<<<< the axis -*———*-*—*—*——— —*—*—-*———-*— —— *———————— ————————

ATTENTION : The [c] input is useful only if you work with pitches. For that reason, please look at the abstraction (the red one).

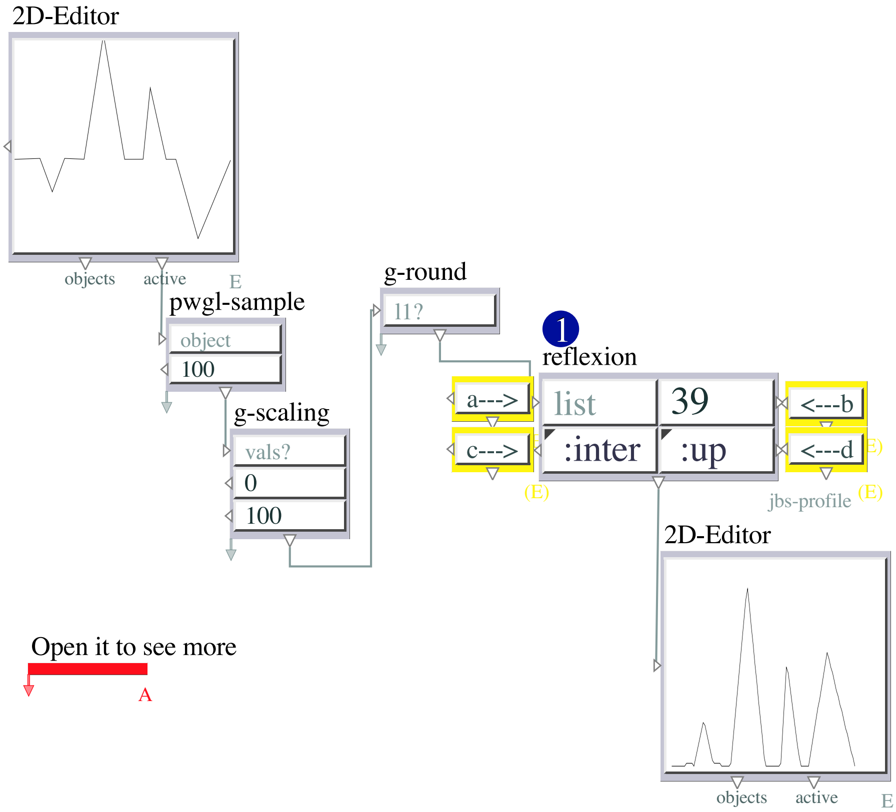

Figure 5: 2-1-reflexion

3.3 2-Double-Reflexion

2.2 - DOUBLE-REFLEXION
This function [1] creates a double reflexion between two given axes.
In the first input [a] you put the profile you want to reflect inside two limits. These limits are defined in the second input [b]. In this example, please, use the PWGL-SWITCH in order to choose the limits.
This function is recursive and it makes the same reflexion until all the points can enter the two limits.
Example: *——————————- -*—————-*————- -*—————*-*———— —*————*— *———— —-*———-*——*———- ——*——*———-*——— ———*—*—————*——— ———-*-*—————— *——- ————*——————-*——
This function makes the reflexion between the two limits.
0————————————- -0—————-0————- -0—————0-0———— —*————*—*————<-
<-<- Upper limit –*-*—*—*-*-*-*———- -*—*-*-*-*—*—*——— *——*—*—————*———<-
<-<- Lower limit ———-0-0—————0—- ————0——————-0—
If you work with pitches, please look at the abstraction (the red one).

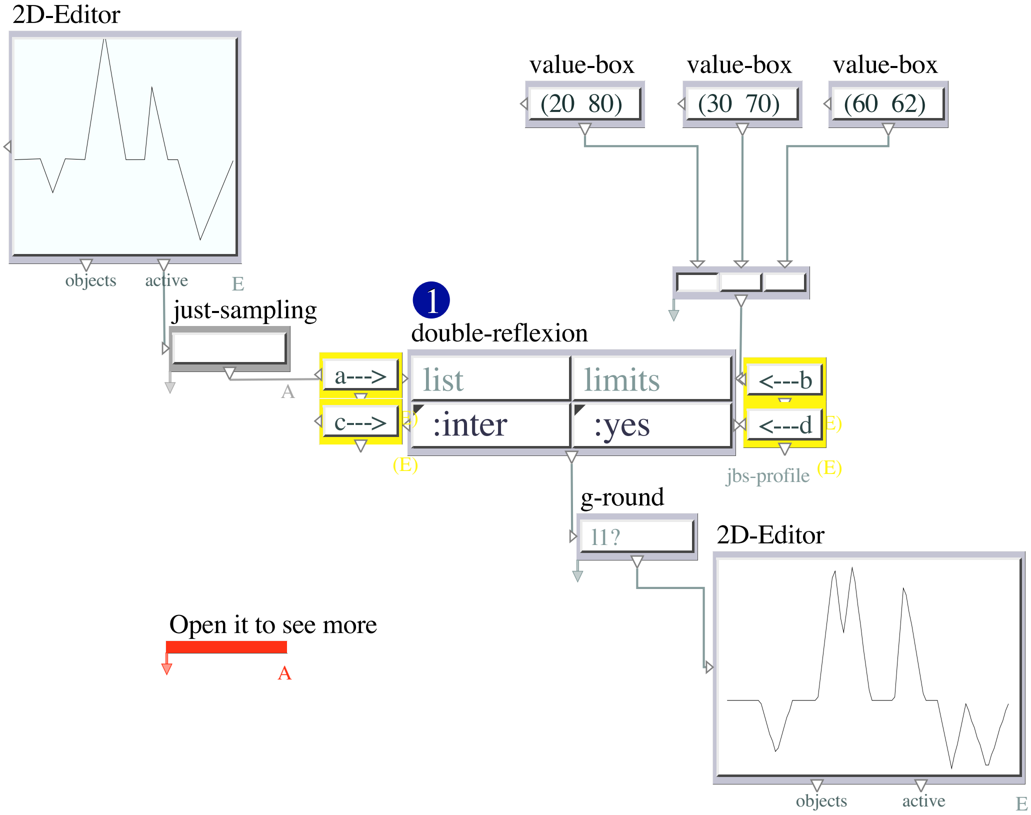

Figure 6: 2-2-double-reflexion

3.4 3-Multi-Reflexion

2.3 - MULTI-REFLEXION
This function [1] allows you to make a particular kind of geometrical reflexion. As in the DOUBLE-REFLEXION here you have a given profile set in the [a] input. Then you have to define two limits. Please use the PWGL-SWITCH in order to chose different limits to put in the second input [b].
This function works like as shown.
The given profile: *———————- -*———-*——- _*——*-*—— __*__ ———*__*—— ___-*—-*___*—-. ___*___*—-*—— ———*__*——*—— ———-*-*——*__ _——*———-*__
The multi reflexion could be:
0———————- -0———-0——- _0——0-0—— __*__*-*__*__*—— <-<-<-< Upper limit *__*__*__*___*—-. -*__*___*__*__*—— _*__*__*__*-*__*___ <-<-<-< Lower limit ——-0-0——0—- ——0———-0—
If you work with pitches, please look at the abstraction (the red one).
ATTENTION: Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL. Look particularly at the patch numbers 1.03, 1.04 and also 2.01 and 2.02.

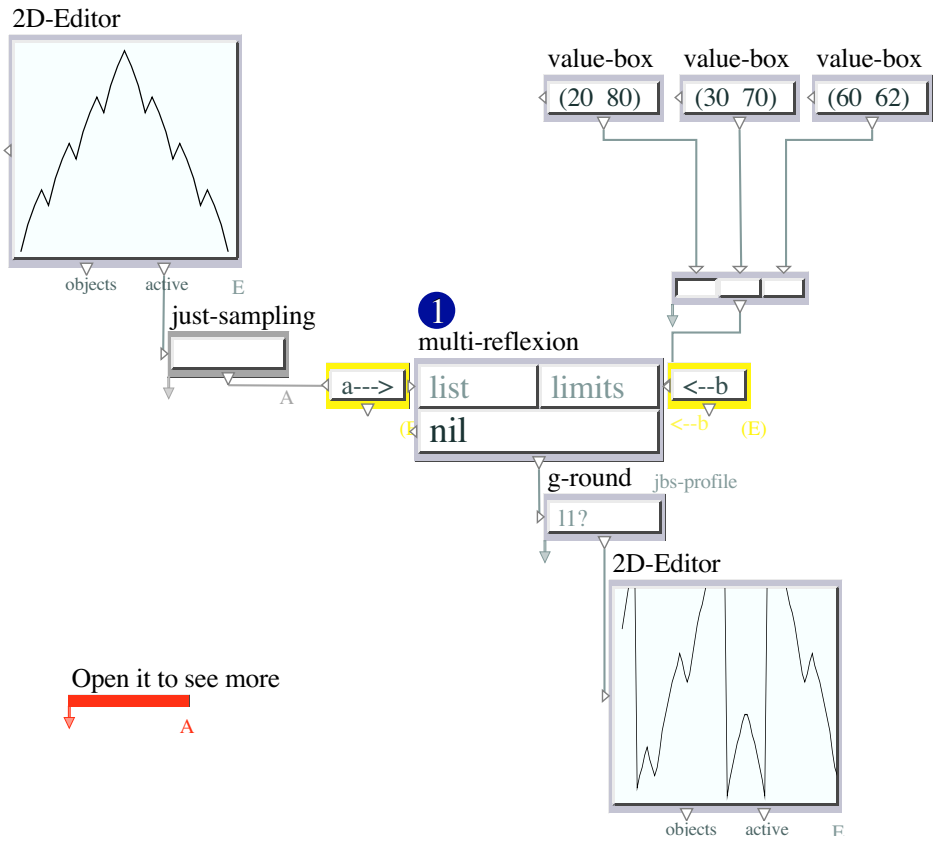

Figure 7: 2-3-multi-reflexion

4 0-Derivations

4.1 0-Derivations

All the functions grouped here can be applied to any kind of parameters that can be represented with numbers.
This chapter is dedicated to the concept of arithmetical derivation.
As an extension of the arithmetical concept, this part of the library considers the derivation as a possible function to simplify a given profile.
The simplification appears here in two ways. The first is given by the reduction of elements constituting the derived result from the original one. Secondly, the derivation is a way of removing specificities inside a curve, keeping as long as possible the global shape.
Almost all these functions are recursive (only the MIN-MAX-DERIVATION is not). In this case recursion means which level of derivation you want to get.
If you put 1 you just obtain the first derivation. If you put 2, you will obtain the derivation of the first one. With 3, the derivation of the derivation of the first one, and so

on.

4.2 1-Arithmetic-Derivation

3.1 - ARITHMETIC-DERIVATION
This function [1] is a metaphor of the arithmetic derivation.
In [a] you put the profile you want to be derivated. In [c] you ask for which derivation you want: the first, the second, the third and so on.
ATTENTION : This function is very special, so for the moment keep the 'start?' Menu on 'first' [b].
Open the red abstraction and read the following documentation.
The derivation function can work in tandem with the integration one. If I ask for the second derivation of the second integration of a curve, I obtain the original one. So, in this case it is exactly what I calculate. First with the derivation function I ask for the second [d] derivation. Then with the integration function (see also the tutorial 4.1 of this same library) I ask to reconstitute the original. To do this I have to set the menu 'start?' on 'orig' [f] and to ask the same number of integration that I used in the derivations (in this case 2 [e]).
These two functions (derivation and integration) allow you to move in the arithmetical space of deriving something and going back to the original.
This function is recursive. In this case recursion means which level of the derivation you want to get.
If you put 1 you just obtain the first derivation. If you put 2, you will obtain the derivation of the first one; with 3, the derivation of the derivation of the first one, and so on.

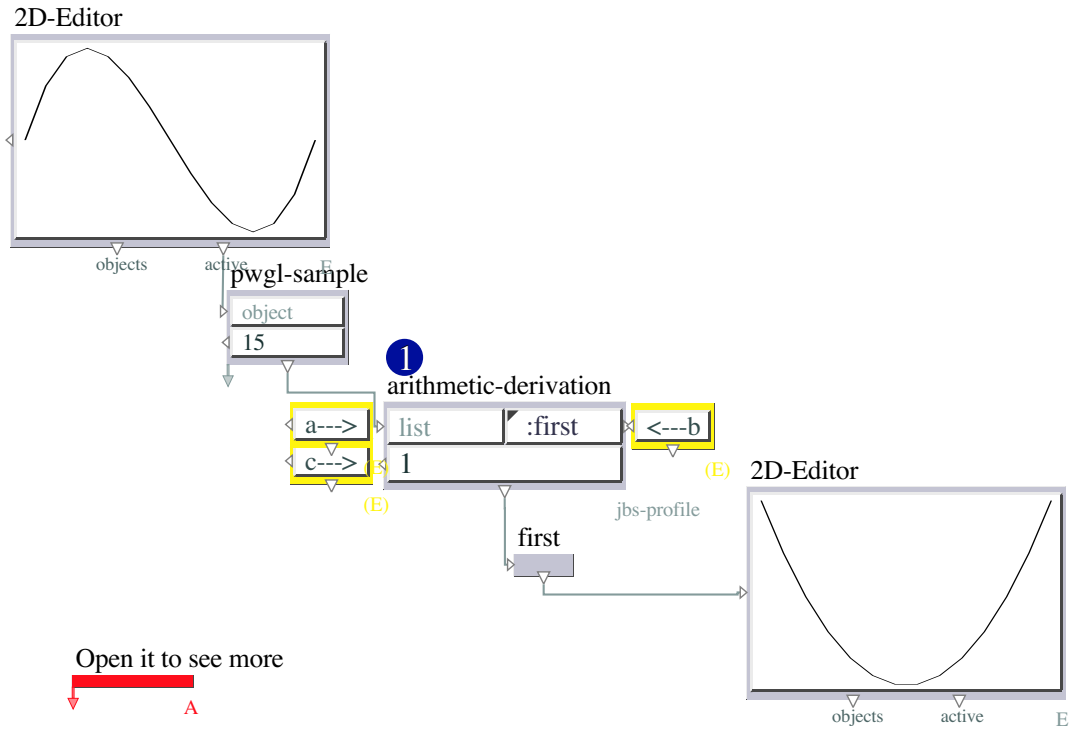

Figure 8: 3-1-arithmetic-derivation

4.3 2-Mean-Derivation

3.2 - MEAN-DERIVATION
This function [1] calculates the average between each couple of points of a given curve. In this way you can obtain a simplification of a given profile.
In [a] you put the curve you want to be derivated.
In [b] you set which derivation you want : the first, the second, the third, and so on.
ATTENTION : You can have only a number of derivations equal to the length of the points constituting the profile minus 1.
If you work with pitches, please open the red abstraction.
This function is recursive. In this case recursion means which level of the derivation you want to get.
If you put 1 you just obtain the first derivation. If you put 2, you will obtain the derivation of the first one; with 3, the derivation of the derivation of the first one, and so on.
ATTENTION : Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL. Particularly look at the patch numbers 1.03, 1.04 and also 2.01 and 2.02.

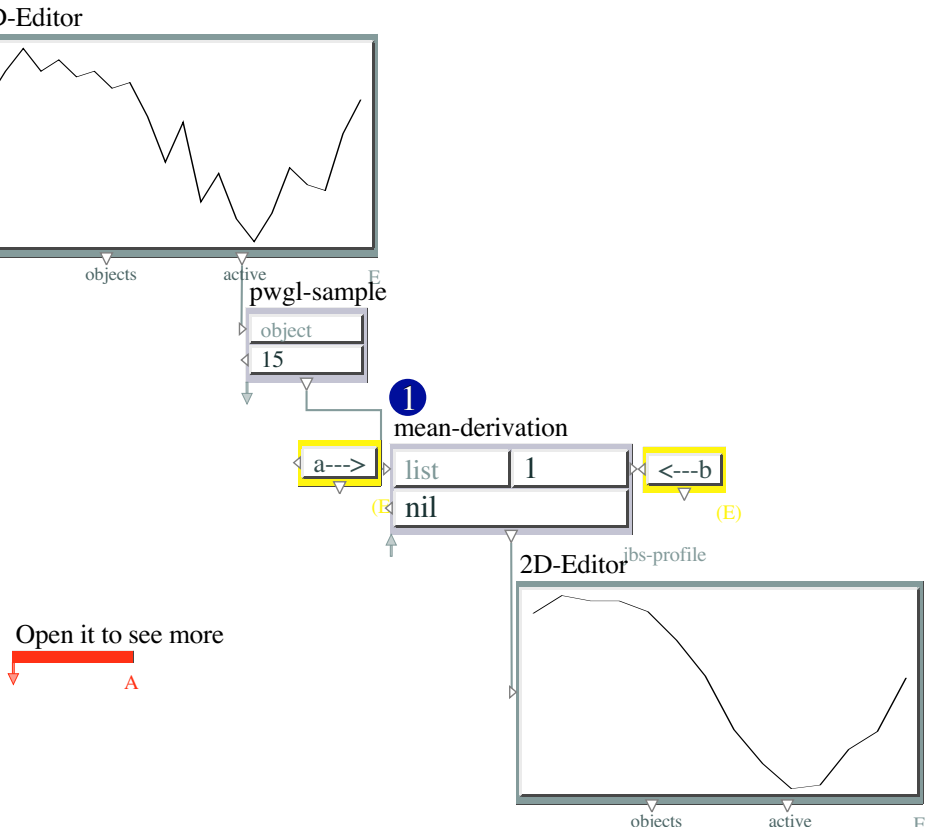

Figure 9: 3-2-mean-derivation

4.4 3-Average-Derivation

3.3 - AVERAGE-DERIVATION
This function calculates the average between all the elements (and not only for each couples of elements) of a given profile.
In [a] you put the profile you want to be derivated. In [b] you set which level of the derivation you want : the first, the second, the third, and so on.
Because this function is recursive, it can happens that after a certain number of derivation you obtain a division by zero. SO for this reason, look at the patch and try out the number I suggest you, for this example.
This function is useful to make another kind of simplification of a curve.
This function is recursive. In this case recursion means which level of the derivation you want to get.
If you put 1 you just obtain the first derivation. If you put 2, you will obtain the derivation of the first one; with 3, the derivation of the derivation of the first one, and so on.

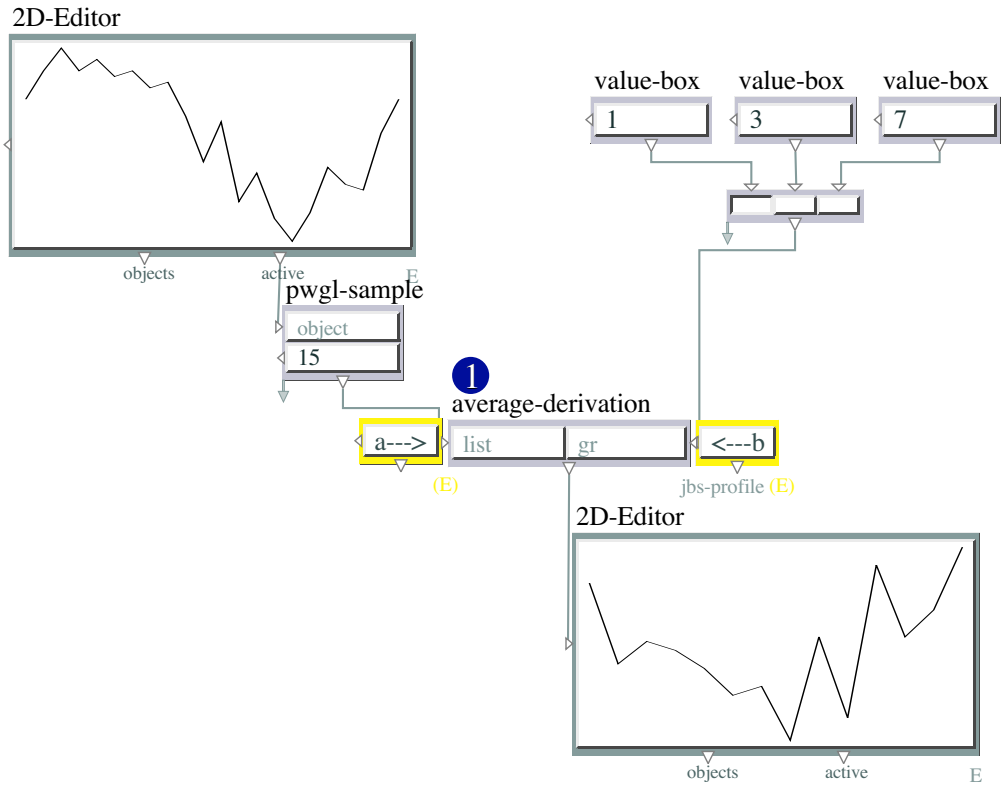

Figure 10: 3-3-average-derivation

4.5 4-Barycentre-Derivation

3.4 - BARYCENTRE-DERIVATION
This function [1] creates another possible simplification of a given profile.
In [a] you put the profile you want to be derivated. In [b] you set which level of the derivation you want : the first, the second, the third, and so on.
This way of deriving calculates each time the barycentre of a list and uses it to constitute the derivation.
ATTENTION : You can have only a number of derivations equal to the length of the points constituting the profile minus 1.
This function is recursive. In this case recursion means which level of the derivation you want to get.
If you put 1 you just obtain the first derivation. If you put 2, you will obtain the derivation of the first one; with 3, the derivation of the derivation of the first one, and so on.

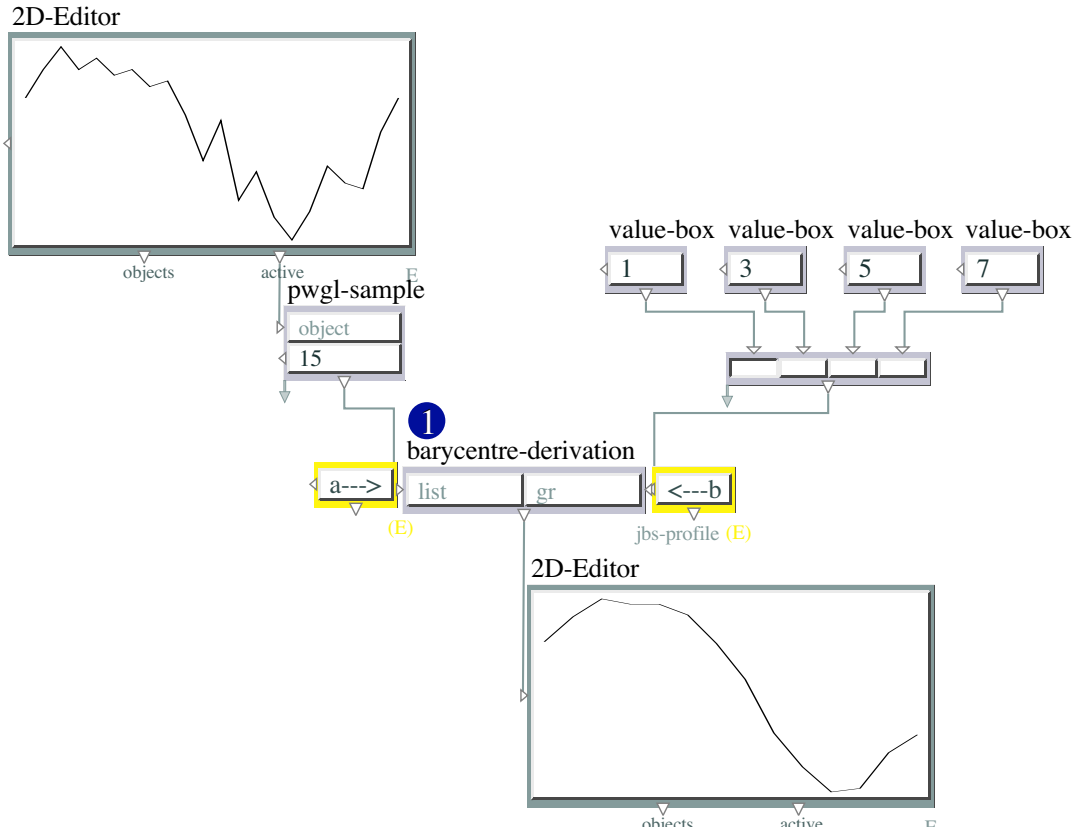

Figure 11: 3-4-barycentre-derivation

4.6 5-Min-Max-Points

3.5 - MIN-MAX-POINTS
This is the last way I found to make a simplification of a profile.
The MIN-MAX-POINTS function [1] gives you only the minimum and maximum points of a given profile within the first and lasts values of the original list.

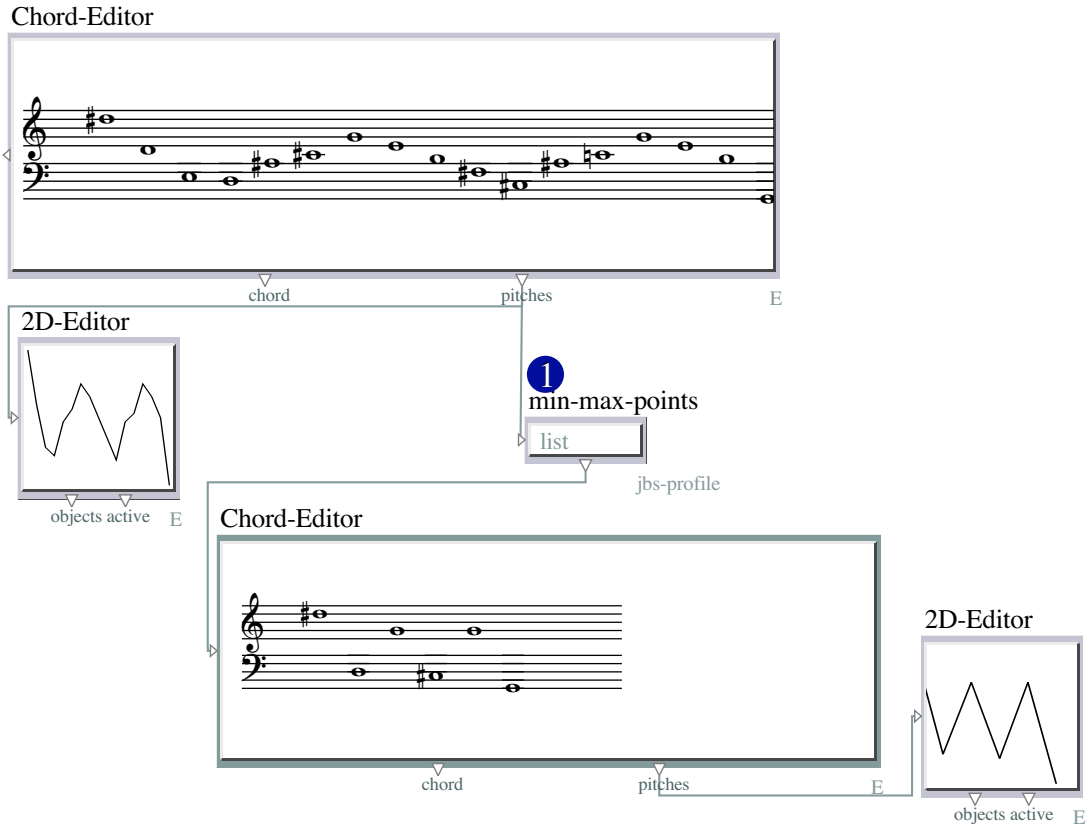

Figure 12: 3-5-min-max-points

5 0-Integrations

5.1 0-Integrations

All the functions grouped here can be applied to any kind of parameters that can be represented with numbers.
This chapter is dedicated to the concept of arithmetical integration.
As an extension of the arithmetical concept, this part of the library considers the integration as a possible function to make more complex a given profile.
The action of making a profile more complex appears here in two ways. The first is given by the augmentation of elements constituting the integrated result from the original one. Secondly, the integration is a way of adding specificities inside a curve, keeping as long as possible the global shape.
All these functions are recursive. In this case recursion means which level of the integration you want to get.
If you put 1 you just obtain the first integration. If you put 2, you will obtain the integration of the first one; with 3, the integration of the integration of the first one,

and so on.

5.2 1-Arithmetic-Integration

4.1 - ARITHMETIC-INTEGRATION
This function [1] is a metaphor of the arithmetic integration.
In [a] you put the profile you want to be integrated. In [c] you ask for which derivation you want : the first, the second, the third and so on.
ATTENTION : This function is very special, so for the moment keep the 'start?' menu on 'first' [b].
Open the red abstraction and read here the documentation.
The integration function can work in tandem with the derivation one. If I ask for the second integration of the second derivation of a curve, I obtain the original one. So, in this case it is exactly what I calculate. First with the integration function I ask for the second [d] one. Then with the derivation function (see also the tutorial 3.1 of this same library) I ask to reconstitute the original. To do this I have to set the menu 'start?' on 'orig' [f] and to ask the same number of derivations that I used in the integrations (in this case 2 [e]).
These two functions (derivation and integration) allow you to move in the arithmetical space of deriving something and going back to the original.
This function is recursive. In this case recursion means which level of the integration you want to get.
If you put 1 you just obtain the first integration. If you put 2, you will obtain the integration of the first one; with 3, the integration of the integration of the first one, and so on.

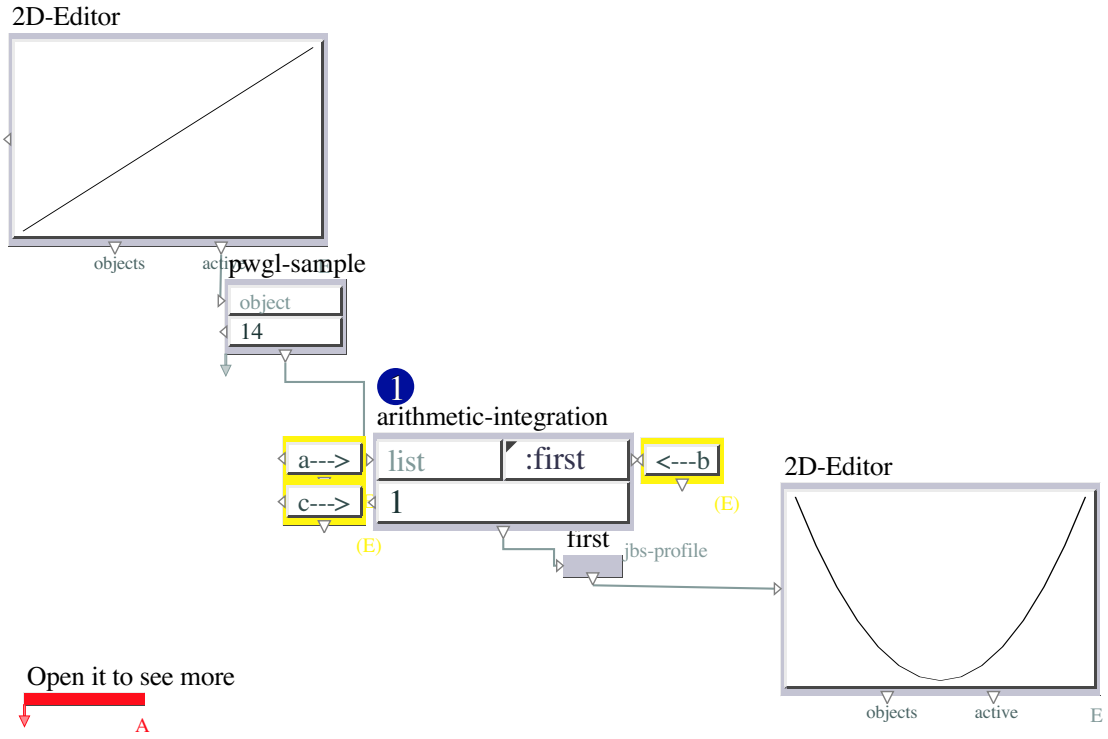

Figure 13: 4-1-arithmetic-integration

5.3 2-Geometric-Integration

4.2 - GEOMETRIC-INTEGRATION
This function [1] creates a sort of integration of a given profile.
In [a] you put the profile, and in [b] you put the numbers you want to be used to make the given profile more complex.
In [c] you can decide how many times you want the original profile to be transformed into a more complex one. So in [c] you ask for which level you want : the first, the second, the third and so on.
Technically, this function inserts the elements you put in [b] between two consecutive elements of the original profile. As this function is recursive, if you put 1 in [c] you get the first insertion. But if you put number bigger than 1, each time you will insert the [b] elements in each new couple of consecutive elements of the previous result.
This function is recursive. In this case recursion means which level of the integration you want to get.
If you put 1 you just obtain the first integration. If you put 2, you will obtain the integration of the first one; with 3, the integration of the integration of the first one, and so on.

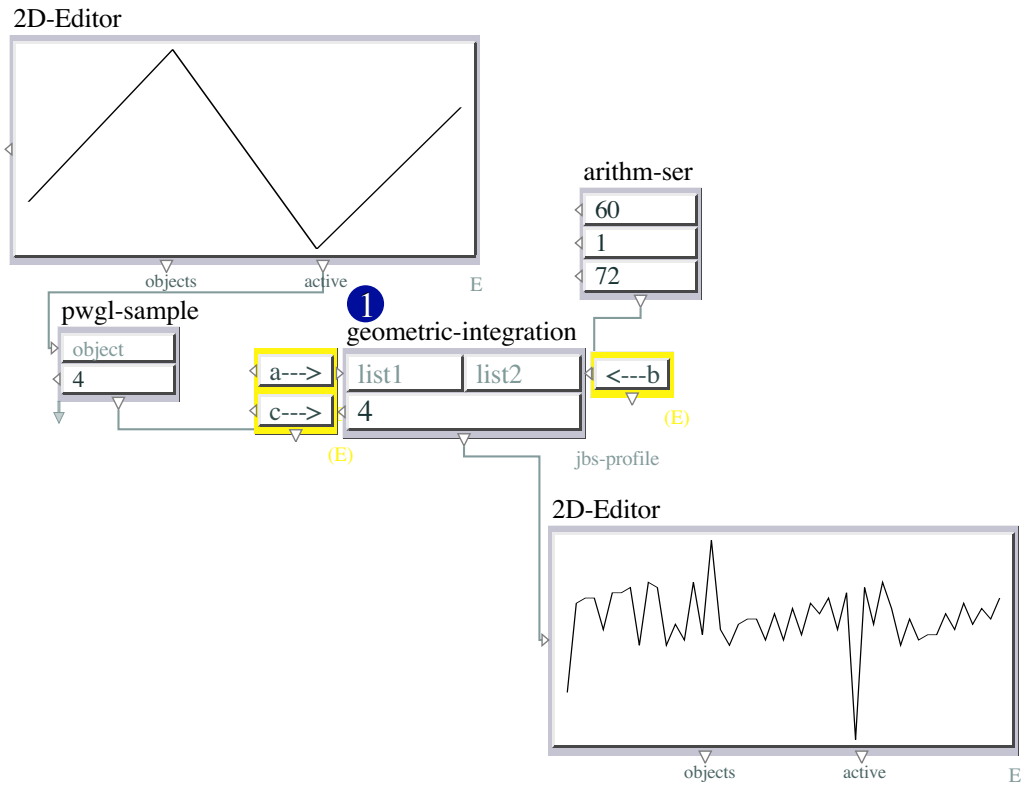

Figure 14: 4-2-geometric-integration

5.4 3-Barycentre-Integration

4.3 - BARYCENTRE-PROFILE
This function [1] gives another way of making a profile more complex.
In [a] you put the profile. With [b] you define which level of integration you want.
This function is recursive. In this case recursion means which level of the integration you want to get.
If you put 1 you just obtain the first integration. If you put 2, you will obtain the integration of the first one; with 3, the integration of the integration of the first one, and so on.

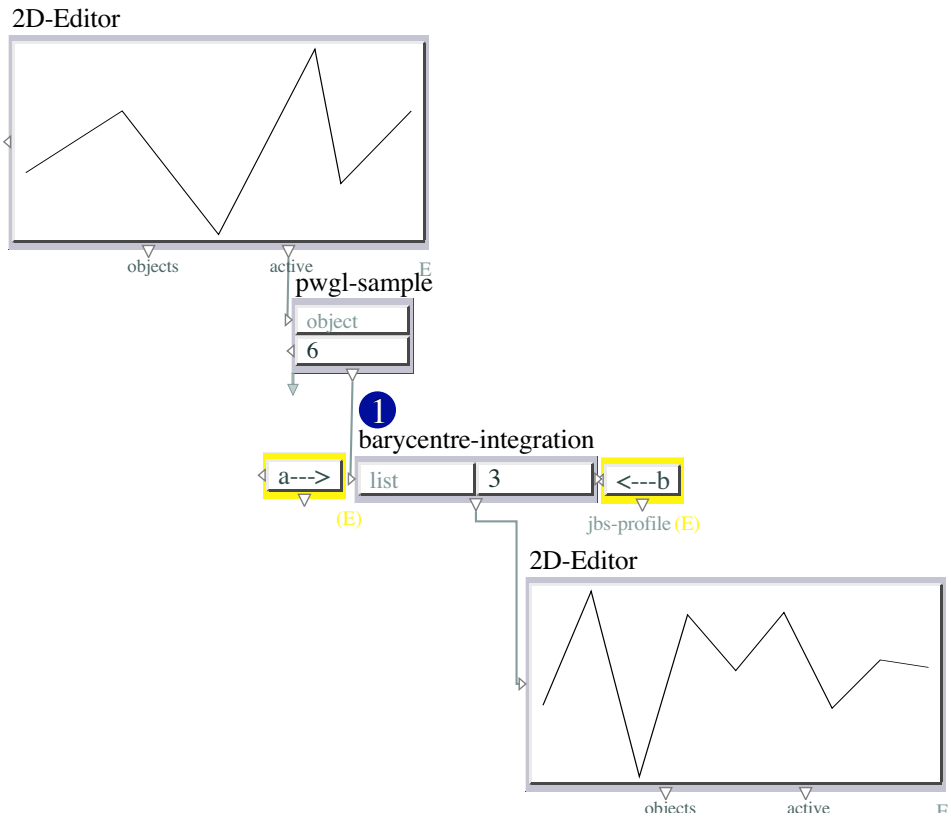

Figure 15: 4-3-barycentre-integration

6 0-Interpolations

6.1 0-Interpolations

The functions of this section are dedicated to the interpolation.
ATTENTION : By interpolation I do NOT mean the mathematical or arithmetical encoding way of making interpolations. Here I use some techniques that are not mathematically exact. I have chosen these interpolators because from a musical point of view they are very convincing.
Obviously that an interpolation is a method of constructing new data points within the range of a discrete set of known data points. In this sense interpolation is a method to calculate values between the ones we already know. But for me the most important aspect is how to make interpolations giving the impression, as much as possible, that from a starting point to an ending point there is no discontinuity. In other words, for me interpolation is a continuous passage from a starting point to an ending one and this continuity depends on many perceptual aspects. That is why I developed some interpolations not using standard linear or polynomial functions, but using intuitive

drawn curves.

6.2 1-Dynamic-Interpolation

5.1 - DYNAMIC-INTERPOLATION
This function [1] creates an interpolation between two lists [begin and end]. In the first you put the starting profile [a]. In the second the ending one [b].
In [c] you define how many steps you want.
ATTENTION : If you put 10 (as it is set in this example) the whole interpolation contains 12 sub-lists. This is because I consider steps independent from the starting and ending profiles. In the menu 'inclu?' [d] you can define if you want or not (with YES and NO) the starting and ending lists.
In 'tab' you can enter a bpf [e] to define the way of going from the first list to the second.
ATTENTION : If you do not specify any bpf, the interpolation is linear.
With 'note?' you can define if you want or not a pitch quantification. Please use the PWGL-SWITCH in order to chose not to have chords (first possibility), to have only one chord for the whole interpolation (second possibility), or to have as many chords as you want for each step of the interpolation.
BE CAREFUL : If you ask for 10 steps, but that you have only a list with less than 10 chords, these chords will be repeated in a circular way.
BE CAREFUL AGAIN : The starting point and the ending one are not included in the pitch quantification.
ATTENTION : Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL. Look particularly at the patch numbers 1.03, 1.04 and also 2.01 and 2.02.

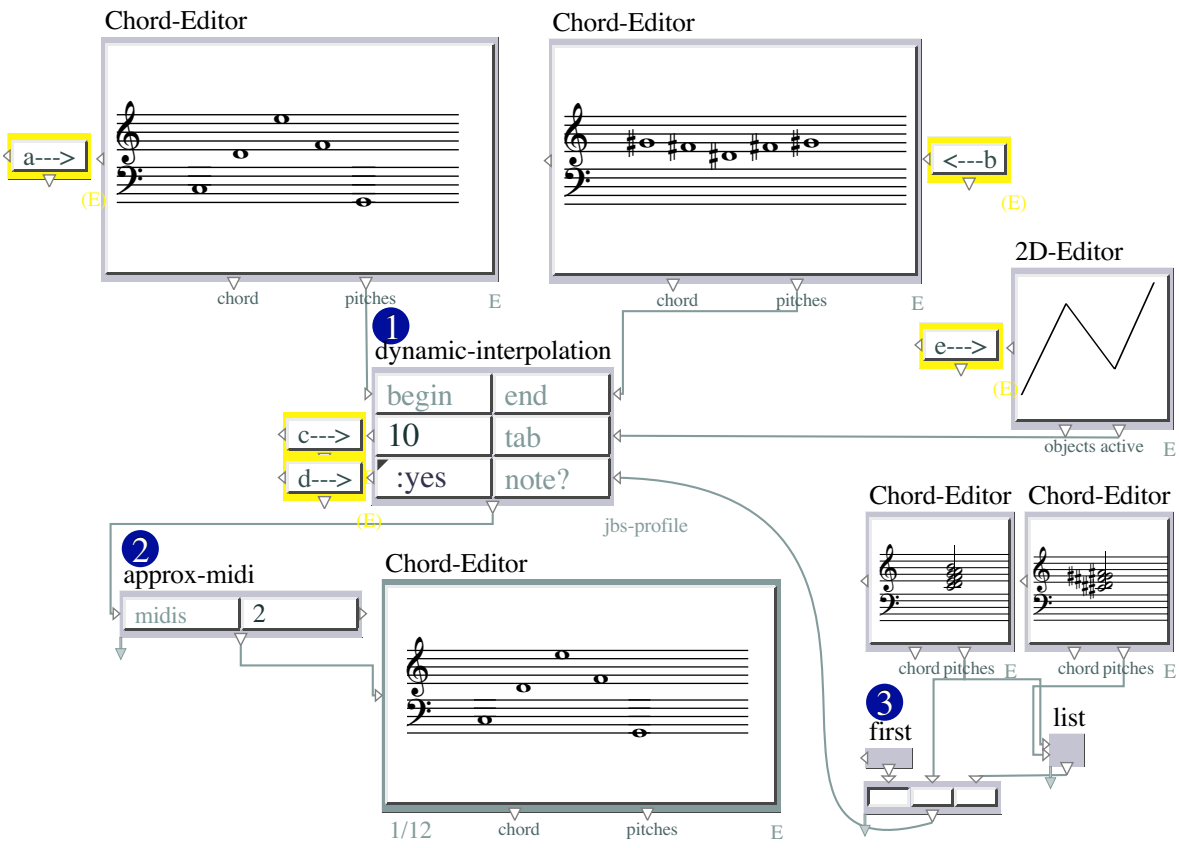

Figure 16: 5-1-dynamic-interpolation

6.3 2-Multi-Interpolation

5.2 - MULTI-INTERPOLATION
This function [1], allows you to create a series of interpolations, with the same criteria developed in the DYNAMIC-INTERPOLATION. (See the tutorial 5.1 of this same Library). The goal is to make an interpolation between each sub-list you enter in [a].
In [a] you put a list of lists. Each sub-list is a point for the interpolation.
In [b] you define how many steps you want for each point of the interpolation.
BE CAREFUL : If the length of this list of steps is inferior to the number of interpolations (defined by the number of sub-lists), the steps will be read in a circular way till the end of the last interpolation.
In [c] you define the curve (bpf) that will be used for the each interpolation. If you do not put any bpf, all the interpolations will be linear. If you put only one bpf, all the interpolations will follow the same shape, but if you put as many curves as there is intermediary interpolations, for each of them the algorithm will use a specific bpf in the order you have defined.
BE CAREFUL : If the number of bpfs is inferior to the number of interpolations, the list of bpfs will be read in a circular way till the end of the last interpolation.

In [d] you put the pitch quantification (if needed). So, if you put only one chord, you will have a global pitch quantification.
BUT BE CAREFUL : The points between interpolations will not be affected by the pitch quantification. If you put more than one chord you can define a specific pitch field for each interpolation between each sub-list. If the number of pitch fields is inferior to the number of chords, these will be read in a circular way till the end of the last interpolation.
ATTENTION : Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL.
Particularly look at the patch numbers: 1.03, 1.04 and also 2.01 and 2.02.

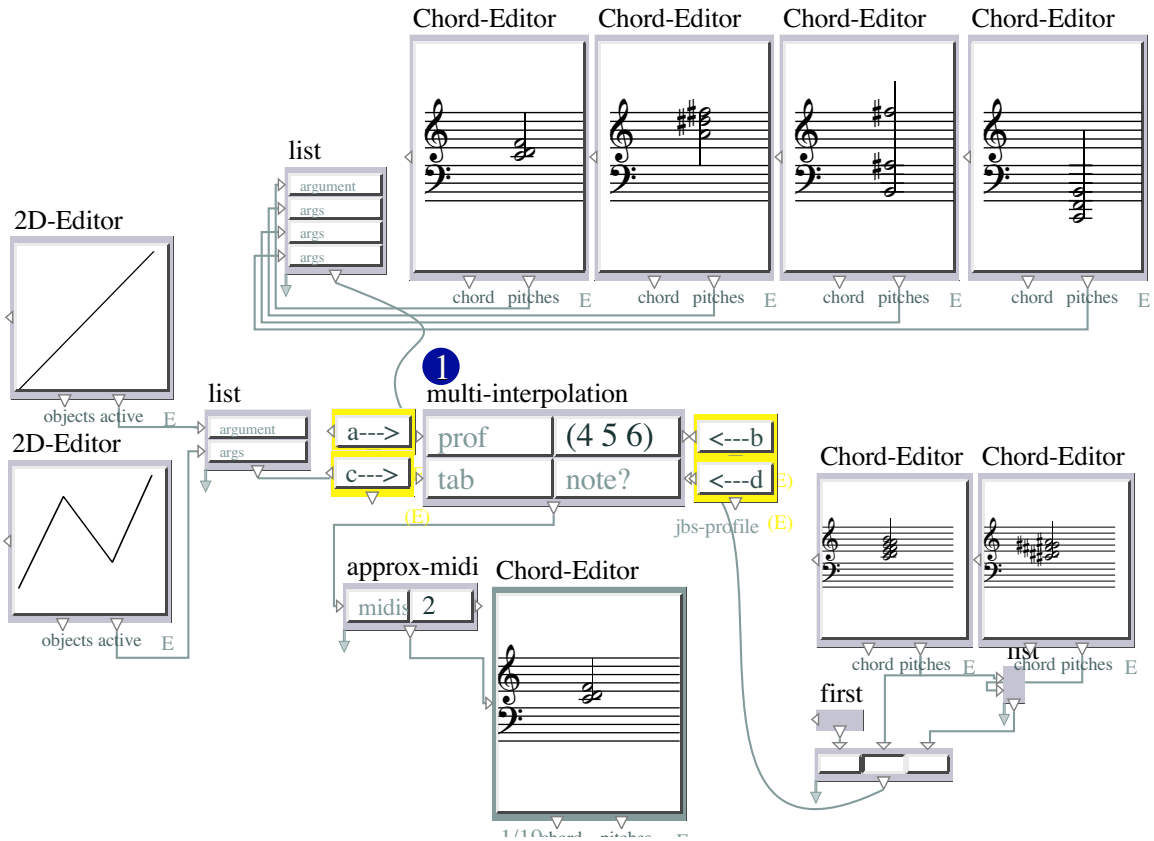

Figure 17: 5-2-multi-interpolation

6.4 3-BPF-Interpolation

5.3 - BPF-INTERPOLATION
This function [1] creates an interpolation of bpfs.
In [a] you put the starting bpf, and in [b] the ending one.
In [c] you define how many samples have to be used to describe each bpf.

In [d] you define the last approximation. 2 is for semitones, 4 for fourth of tone, 100 for hundredths of tones...
In [e] you enter how many steps you want between the starting and the ending bpfs.
In [f] you can define by another bpf the way of going from the starting bpf to the ending one.

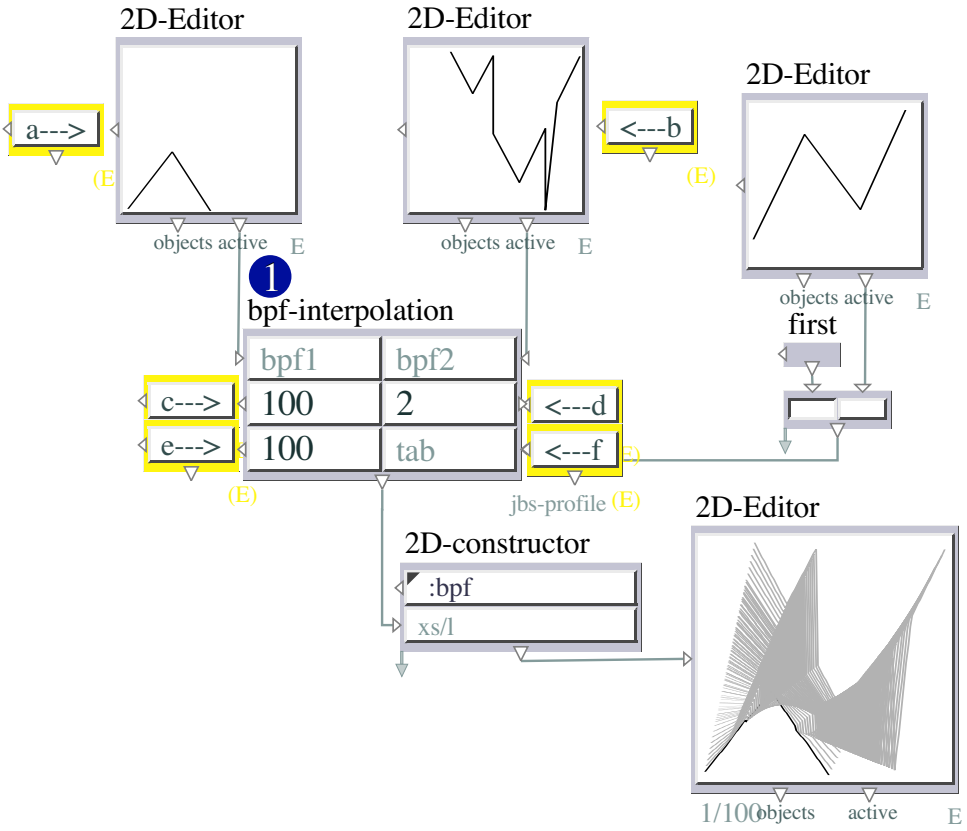

Figure 18: 5-3-bpf-interpolation

6.5 4-Profile-Interpolation

5.4 - PROFILE-INTERPOLATION
This function [1] creates an interpolation between two lists having two different numbers of elements.
In [a] you put the starting profile, and in [b] the ending one.
In [c] you define how many steps you want.
ATTENTION : If you put 12 (as in this example) the whole interpolation has 14 sub-lists. This is because I consider steps independent from the starting and ending profiles.
In [d] you can define how many numbers will constitute each sub-list.
ATTENTION : If you put a number of lengths of sub-groups smaller than the number of steps, the NBR-N will be read in a circular way.

ATTENTION AGAIN : The starting and ending point are not affected by NBR-N.
PLEASE try out the three possibilities using the PWGL-SWITCH [2].
In tab [e] you can enter a bpf to define the way of going from the first list to the second one. If you do not define any bpf the interpolation will be linear.
With 'note?' [f] you can define if you want or not a pitch quantification. Please use the PWGL-SWITCH [4] in order to choose not to have chords (first possibility), to have only one chord for the whole interpolation (second possibility), or to have as many chords as there is steps for the interpolation.
BE CAREFUL : If your ask for 12 steps, and that you give a list with less than 12 pitch fields, these will be repeated in a circular way.
BE CAREFUL AGAIN : The starting point and the ending one are not affected in the pitch quantification.
In [g] you have to decide the sample rate to be as precise as you want. This 'precis index is a multiplication factor to up-sample the given profiles in order to have a good interpolation quality.
ATTENTION : The bigger is this value the slower will be the computation.
In [h] you can define the pitch approximation : 2 is for semitone, 4 is for quarter of tone, 8 for eight of tone, and so on.
Open the red abstraction if you want to look at an old example...
ATTENTION : Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL.
Particularly look at the patch numbers : 1.03, 1.04 and also 2.01, and 2.02.

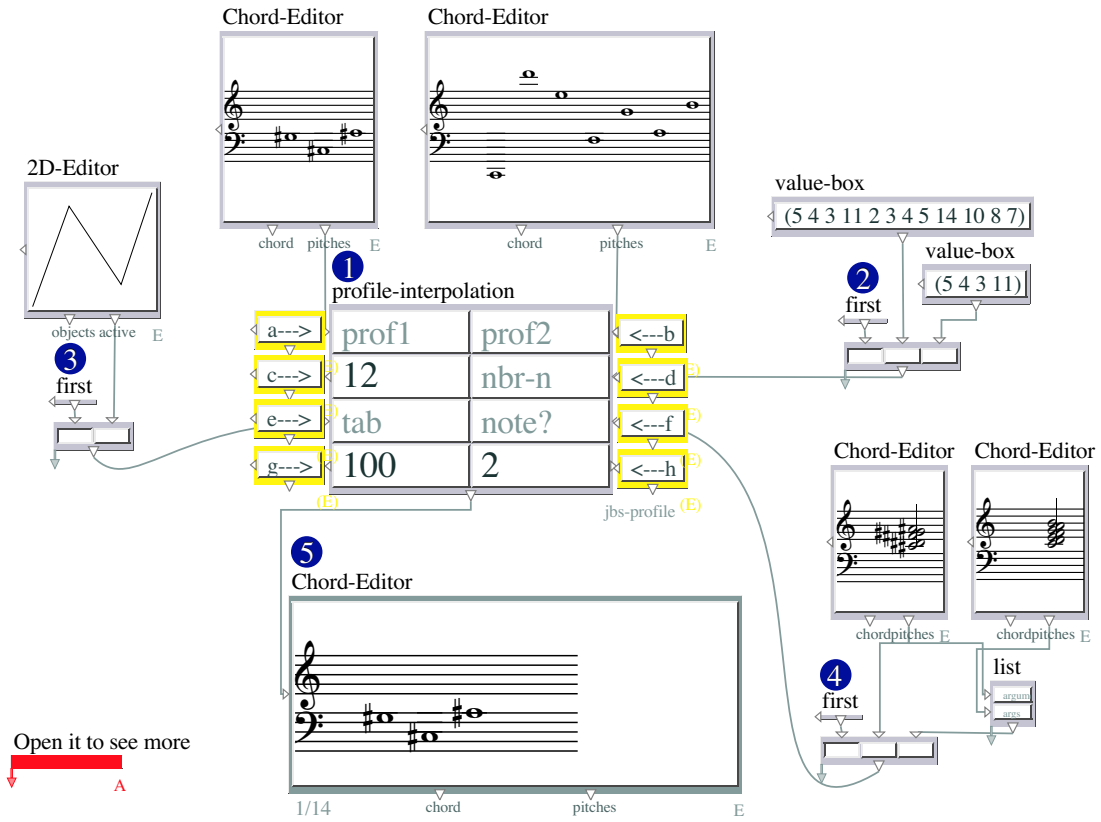

Figure 19: 5-4-profile-interpolation

7 0-Utilities

7.1 0-Utilities

Just some useful functions

7.2 1-Note-Change

6.1 - NOTES-CHANGE
This function [1] makes a pitch quantification.
In [a] you put a list of pitches, and in [b] the scale that will be used for quantification.
ATTENTION : The approximation of the quantification is done by floor.
You can also chose the modulo in 'mod'.
ATTENTION : Those who might be interested in pitch control of melodic profiles, please see the tutorial of the JBS-Constraints library inside PWGL. Particularly look at the patch numbers 1.03, 1.04 and also 2.01 and 2.02.

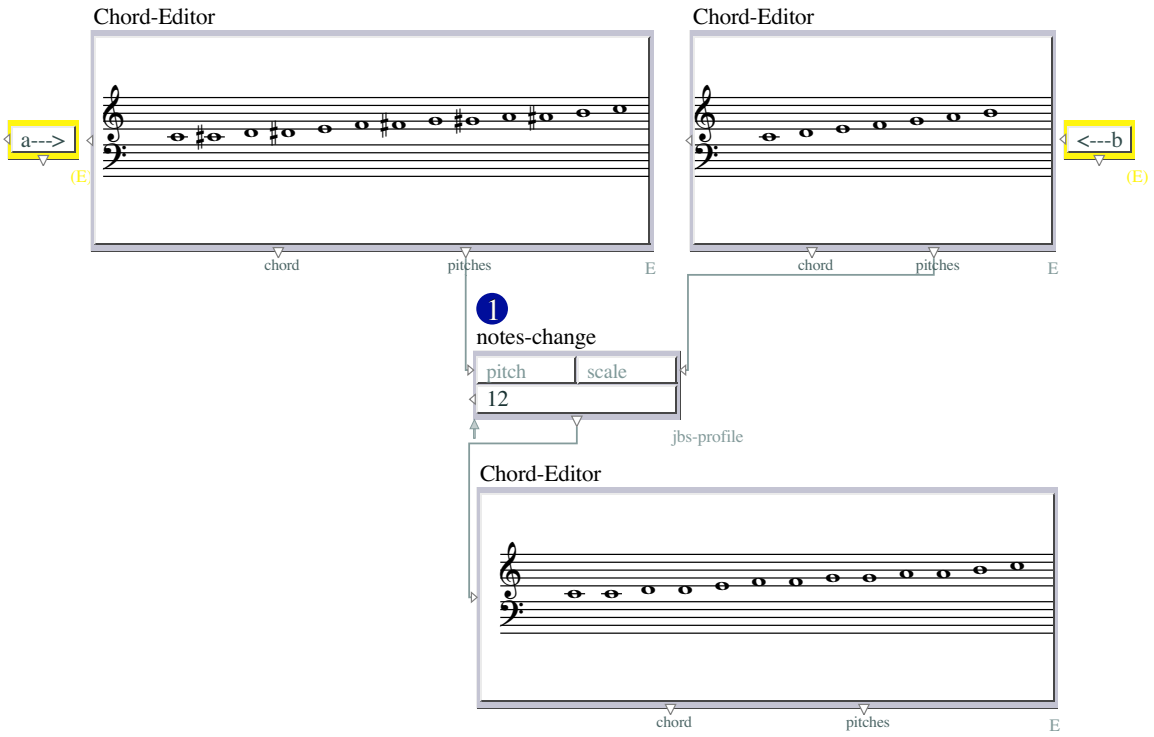

Figure 20: 6-1-note-change

7.3 2-Range-Approx

6.2 - RANGE-APPROX
This function [1] transposes a list of pitches [a] inside two limits [b].
If the transposition does not fit inside the two limits, you can chose to remove the pitches that are not included using the menu 'inclu?' with 'no'. If you put 'yes', then the pitches are transposed as close as possible to the upper and lower limits.

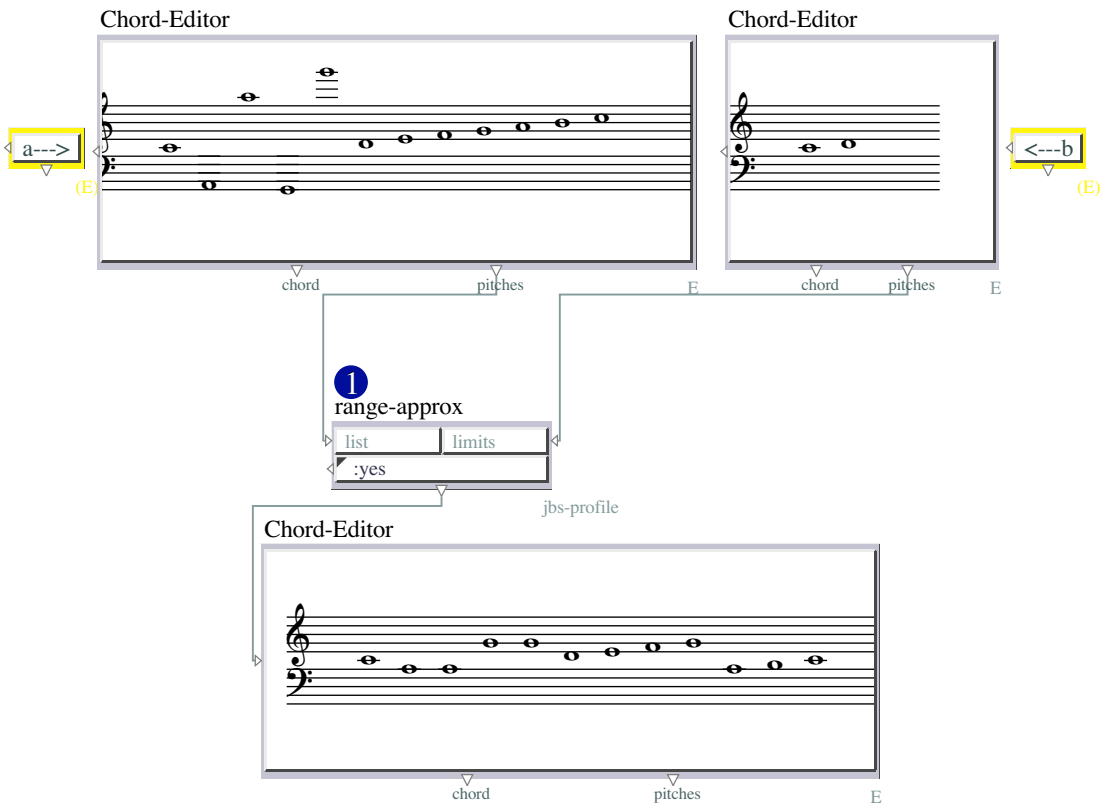

Figure 21: 6-2-range-approx

7.4 3-Weight-Average

6.3 - WEIGHT-AVERAGE
This function [1] calculates the barycentre of a given list [a], by doing the global average.

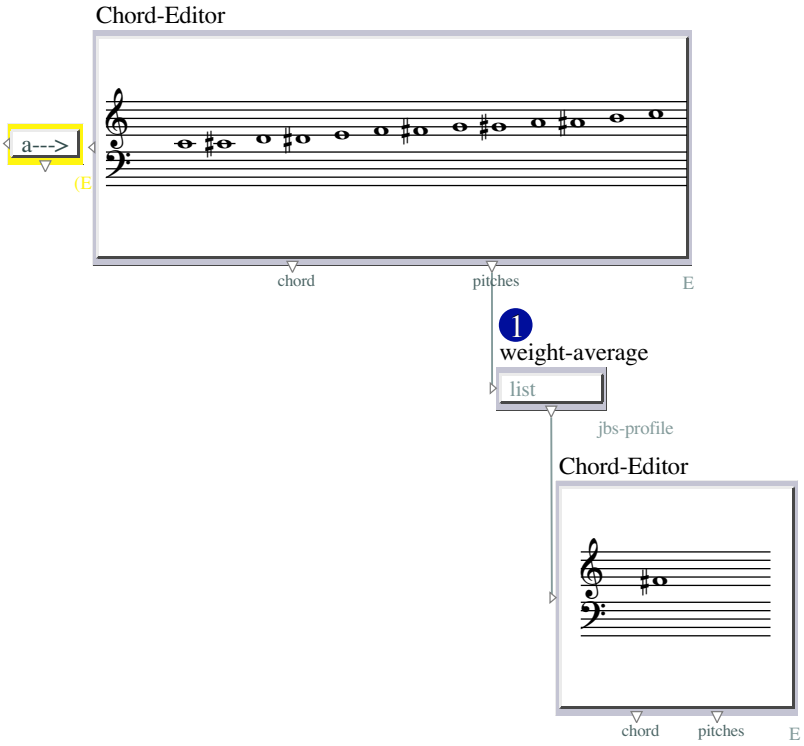

Figure 22: 6-3-weight-average

7.5 4-BPF-Collector

6.4 - BPF-COLLECTOR
This function [1] appends a list of bpfs.
In [a, b, c] you define (in this case) three bpfs. In fact you can put how many bpf you want by extending the BPF-COLLECTOR box.
In [d] you choose the position matching of the entered list of bpfs.
In [e] you define the sample rate for the whole bpfs.
Please evaluate the 2D-Editor [2] and look at the result.

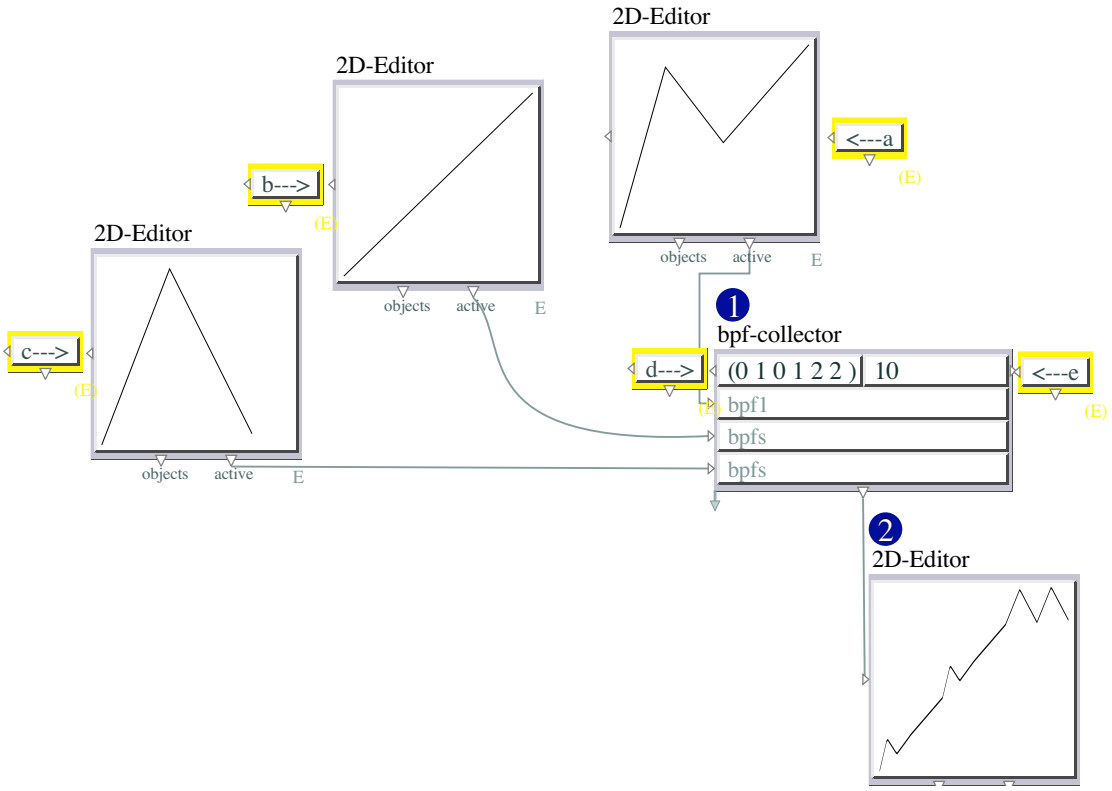

Figure 23: 6-4-bpf-collector

7.6 5-Four-Forms

6.5 - FOUR-FORMS
This function [1] creates the four historical forms of a given pitch list or bpf [a] : the original, the reversed, the inversed and the reversed-inversed.
With the menu [b] you can choose which which form you want to produce.

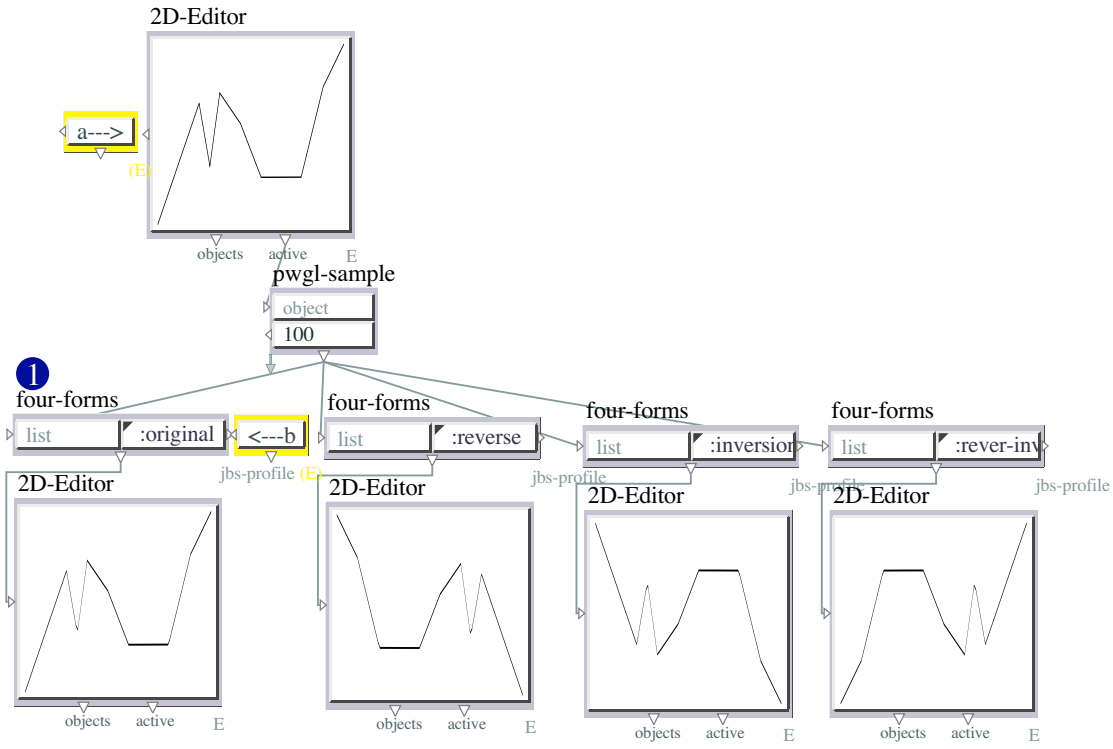

Figure 24: 6-5-four-forms

Box Index

2D-Editor, 3, 4, 6, 8, 9, 11, 13–17, 19–21, 23–25, 27, 31, 32
2D-constructor, 25

Abstraction, 4, 6, 8, 9, 11, 13, 14, 19, 27
alea-perturbation, 3
approx-midi, 23, 24
arithm-ser, 20
arithmetic-derivation, 13
arithmetic-integration, 19
average-derivation, 15

barycentre-derivation, 16
barycentre-integration, 21
bpf-collector, 31
bpf-interpolation, 25

Chord-Editor, 6, 7, 17, 23, 24, 27–30
comment-box, 3, 4, 6–9, 11, 13–17, 19–21, 23–25, 27–32
compression-expansion, 6
control-perturbation, 4

double-reflexion, 9
dynamic-interpolation, 23

first, 6, 13, 19, 23–25, 27
four-forms, 32

g-round, 8, 9, 11
g-scaling, 3, 4, 8
geometric-integration, 20

hybridating-profile, 7

list, 23, 24, 27

mean-derivation, 14
min-max-points, 17
multi-interpolation, 24
multi-reflexion, 11

notes-change, 28

profile-interpolation, 27
pwgl-sample, 3, 4, 8, 13–16, 19–21, 32
pwgl-switch, 4, 6, 9, 11, 15, 16, 23–25, 27

range-approx, 29
reflexion, 8

text-box, 3, 4, 6–9, 11, 13–16, 19–21, 23–25, 27–32

value-box, 9, 11, 15, 16, 27

weight-average, 30