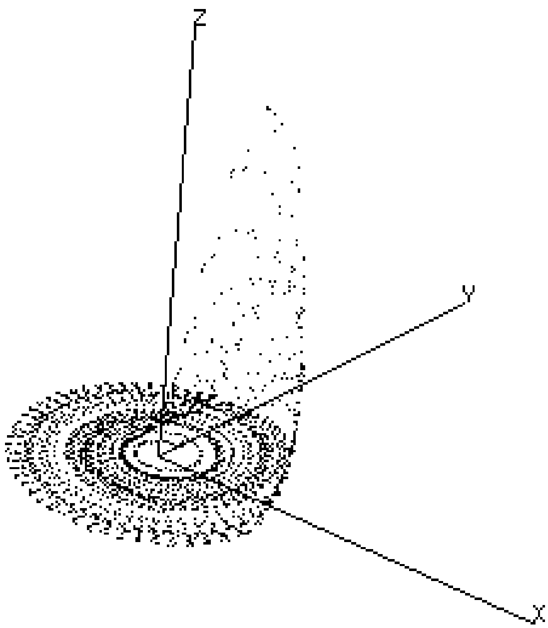
Ircam documentation
- Research reports
- Musical works
- **Software**

PatchWork

Chaos Library



Second English Edition, March 1997

IRCAM Centre Georges Pompidou

This manual was written by Mikail Malt, translated into English by J. Fineberg, and was produced under the editorial responsibility of Marc Battier, Marketing Office, Ircam.

Patchwork was conceived and programmed by
Mikael Laurson, Camilo Rueda, and Jacques Duthen.

The Chaos library was conceived and programmed by Mikail Malt.

First English edition of the documentation, February 1996.
Second edition, revised, March 1997.

This documentation corresponds to version 1.1 of the library, and to version 2.1 or higher of PatchWork.

Ircam
1, place Igor-Stravinsky
F-75004 Paris
Tel. 01 44 78 49 62
Fax 01 44 78 15 40
E-mail ircam-doc@ircam.fr

IRCAM Users' group

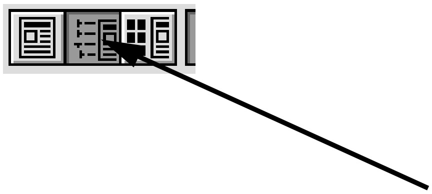
The use of this software and its documentation is restricted to members of the Ircam software users' group. For any supplementary information, contact:

Département de la Valorisation
Ircam
Place Stravinsky, F-75004 Paris

Tel. 01 44 78 49 62
Fax 01 44 78 15 40
E-mail: bousac@ircam.fr

Send comments or suggestions to the editor:
E-mail: bam@ircam.fr
Mail: Marc Battier,
Ircam, Département de la Valorisation
Place Stravinsky, F-75004 Paris

http://www.ircam.fr/forumnet



To see the table of contents of this manual, click on the Bookmark Button located in the Viewing section of the Adobe Acrobat Reader toolbar.

Contents

Résumé

La librairie Chaos est composée de plusieurs séries de modules PatchWork. Ces modules sont destinés à générer et traiter des valeurs numériques selon différents modèles de systèmes dynamiques et non linéaires, ainsi que par des algorithmes fractals. La librairie est diuvisée en quatre parties :

Orbitals

La section Orbitals section contient un groupe de modules basés sur des équations récursives et sur la résolution d’équations différentielles.

IFS

La section IFS contient des modules destinés à créer des systèmes récursifs linéaires, afin de construire, par exemple, des objets fractals.

Fractus

La section Fractus contient trois algorithmes pour générer des courbes fractales.

Outils

La section Tools contient des outils pour effectuer des manipulations géométriques en deux dimensions.

La librairie Chaos ne propose pas de liaison directe avec des applications musicales. Chaque utilisateur est libre d’en concevoir à sa guise. A cet effet, les références bibliographiques placées à la fin du manuel peuvent s’avérer utiles.

1 Introduction

The Chaos Library consists of a series of PatchWork modules which may be used to generate and manipulate numerical values based upon various different models of dynamic and non-linear systems as well as fractals. This library is divided into four parts:

Orbitals

The Orbitals section of the library contains a group of modules which generate values based upon the iterations of recursive equations and the resolution of differential equations.

IFS

This section contains modules for creating and manipulating linear recursive systems. This type of system permits the construction of fractal objects and are a generalization of linear transformation in a plane.

Fractus

Three algorithms for generating fractal curves.

Outils

Tools for manipulating geometry in two dimensions.

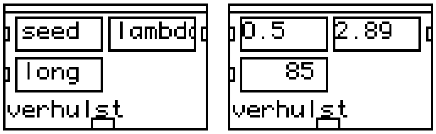
As opposed to other PatchWork libraries, Chaos does not immediately lend itself to a musical application. All of its constituent modules were conceived in such a way as to be as close as possible to the original mathematical models. It is left to each composer to decide how the generated material should be "read." Any musical application of an abstract model must be more than a simple application of the algorithm; rather, it should be a reflection on the relationship between the mathematical model and its musical potential.

It is strongly advised that users of this manual consult the bibliography at the end of this document. It is useful for familiarization with the concepts underlying the presented model, and also for deepening ones understanding of these concepts.

2 Orbitals

This section is made up of a group of equations for non-linear dynamic systems.

1.1 verhulst



Syntax

(alea::verhulst *seed lambda long*)

Inputs

seed	whole or floating-point number between zero and one
lambda	whole, floating-point number or list of values between zero and three
long	whole or floating-point number

Output

list

Generates a sequence of length long based on the logistical equation of Pierre-François Verhulst :

yn = xn-1 + xn-1 * *lambda* * (1 - xn-1)

This equation describes population growth.

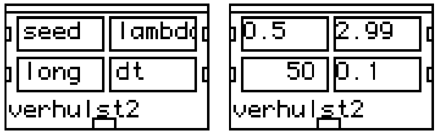
• *lambda* is a number or a list of parameters which define the 'turbulence' of the generated values;

• *seed* is an initial value between zero and one (this value represents the initial population as a ratio to the maximum population);

• *long* is the length of the list generated, which is equivalent to the number of iterations.

The output of this module is a list of values for each iteration.

1.2 verhulst2



Syntax

(alea::verhulst2 *seed lambda long dt*)

Inputs

seed	whole or floating-point number between zero and one
lambda	whole, floating-point number or list of values between zero and three
long	whole number greater than or equal to one
dt	whole or floating-point number

Output

list

Generates a sequence of length long based on the logistical equation of Pierre-François Verhulst (see **verhulst** above).

• *lambda* is a number or a list of parameters which define the 'turbulence' of the generated values;

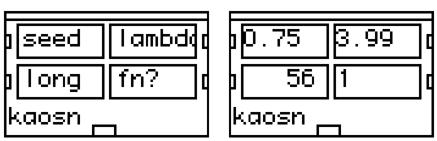
• *seed* is an initial value between zero and one (this value represents the initial population as a ratio to the maximum population);

• *long* is the length of the list generated, which is equivalent to the number of iterations. This version allows the manipulation of the parameter of time *dt*;

• *dt* is a value of time for the numerical integration in the equations.

The output of this module is a list of values for each iteration.

1.3 kaosn



Syntax

(alea::kaosn *seed lambda long fn?*)

Inputs

seed	whole or floating-point number between zero and one
lambda	whole, floating-point number or list of values between zero and four
long	whole number greater than or equal to one
fn?	whole number greater than or equal to one

Output

list

Generates a sequence of length long based on the logistical equation:

$y_n = x_{n-1} * \text{lambda} * (1 - x_{n-1})$ where *lambda* is a number or a list of parameters which define the 'turbulence' of the generated values.

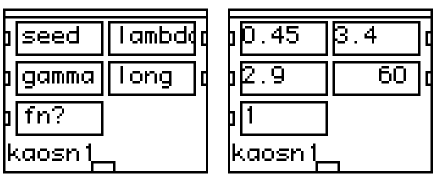
• *seed* is an initial value between zero and one;

• *fn* is the degree of iteration of the logistical equation, if fn = n the sequence calculated will be the function composed of $y_n = y(y_{n-2})$;

• *long* is the length of the list generated, which is equivalent to the number of iterations;

The output of this module is a list of values for each iteration.

1.4 kaosn1



Syntax

(alea::kaosn1 *seed lambda gamma long fn?*)

Inputs

seed	whole or floating-point number between zero and one
lambda	whole, floating-point number or list of values between zero and four
gamma	whole, floating-point number or list of values between zero and four
long	whole number greater than or equal to one
fn?	whole number greater than or equal to one

Output

list

Generates a sequence of length long based on a variation of the logistical equation:

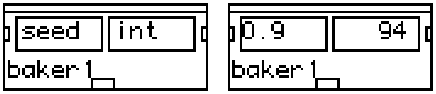
$yn = x_{n-1} * lambda - gamma *x_{n-1}^2$

where

- *lambda* and *gamma* are the parameters which define the 'turbulence' of the generated values;
- *seed* is an initial value between zero and one;
- *long* is the length of the list generated, which is equivalent to the number of iterations.

The output of this module is a list of values for each iteration.

1.5 baker1



Syntax

(alea::baker1 *seed int*)

Inputs

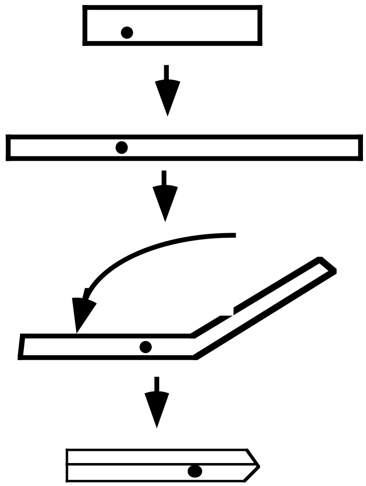
seed whole or floating-point number between zero and one

int whole number greater than or equal to one

Output

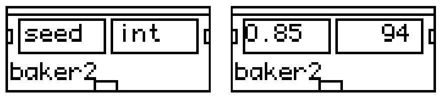
list

Baker's transformation (*Stretch and fold*), for this transformation we consider that the dough has an initial length of one. At moment zero a grain of spice is placed at the coordinate *seed*. This module allows the position of that grain to be determined after *int* number of iterations. The bakers work is, in this case, modeled in such a way as that each iteration corresponds to the complete stretching of the dough to double its length and its refolding in a way that it regains its original length of one.



The output of this module is a list of positions (between zero and one) of the hypothetical grain of spice, after each iteration.

1.6 baker2



Syntax

(alea::baker2 *seed int*)

Inputs

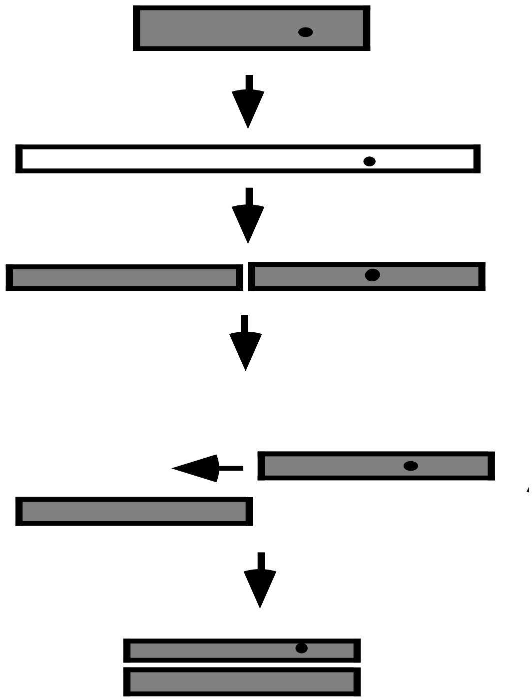
seed whole or floating-point number between zero and one

int whole number greater than or equal to one

Output

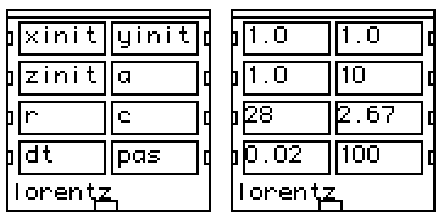
list

Baker's transformation (*Stretch, cut and paste*) , for this transformation we consider that the dough has an initial length of one. At moment zero a grain of spice is placed at the coordinate *seed*. This module allows the position of that grain to be determined after *int* number of iterations. The bakers work is, in this case, modeled in such a way as that each iteration corresponds to the complete stretching of the dough to double its length, the cutting of the dough into two pieces and the superposition of those pieces, as shown in the following illustration:



The output of this module is a list of positions (between zero and one) of the hypothetical grain of spice, after each iteration.

1.7 lorentz



Syntax

(alea:lorentz *xinit yinit zinit a R c dt pas*)

Inputs

xinit	whole or floating-point number
yinit	whole or floating-point number
zinit	whole or floating-point number
a	whole or floating-point number
R	whole or floating-point number
c	whole or floating-point number
dt	whole or floating-point number
pas	whole number greater than or equal to one

Output

List of coordinates in three dimensions.

Lorentz's equation system :

dx= -ax + ay

dy= Rx - y - xz

dz= -cz + xy

These equations give an approximate description of a fluid layer heated from below. The warm fluid which is below is lighter, and thus tends to rise. This creates a convection movement. If the temperature difference between the top and bottom is sufficiently large, the convection will be turbulent and irregular. The parameter *R* is proportional to the temperature difference, this is referred to as the Reynolds number. The parameter a is the Prandtl number.

• *xinit*, *yinit* and *zinit* are the initial coordinates;

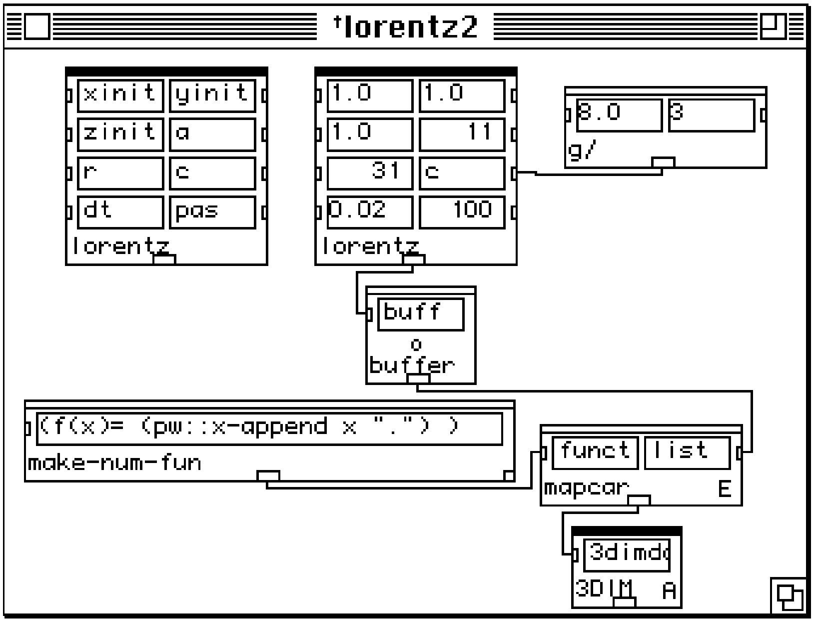
• *pas* is the number of iterations, or generated points;

• *dt* is a value of time for the numerical integration in the equations.

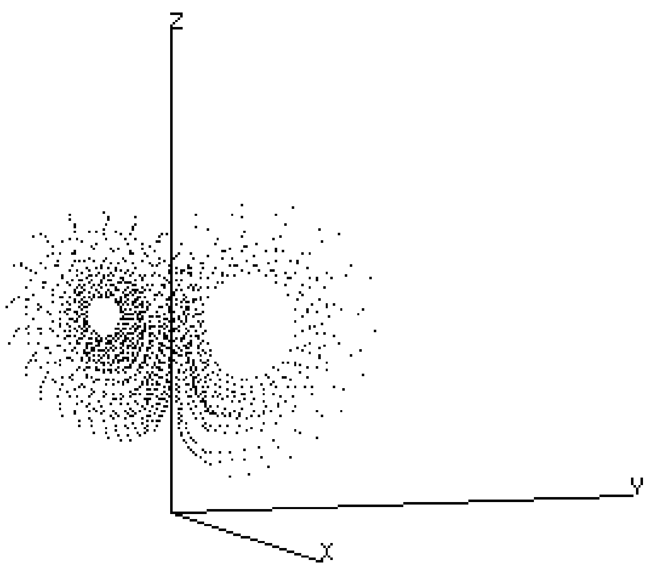
The output of this module is a list of coordinates in three dimensions :

((*xinit yinit zinit*) (x_0 y_0 z_0) (x_1 x_2 x_3) ... (x_n y_n z_n)).

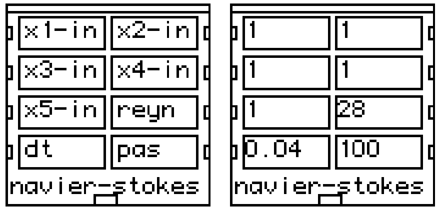
Here is an example patch. Be careful : the library 3Dim-disp must be loaded before opening this patch so as to give access to the module **3dim**, for the three dimensional display.



☛ the modules **make-num-fun** and **mapcar** (Lisp functions) are used here to add the character 'point' to each sub-list of coordinates (see the 3Dim-disp library's documentation) to make the display easier to understand.



1.8 navier-stokes



Syntax

(alea::navier-stokes *x1-in x2-in x3-in x4-in x5-in reyn dt pas*)

Inputs

x1-in	whole or floating-point number
x2-in	whole or floating-point number
x3-in	whole or floating-point number
x4-in	whole or floating-point number
x5-in	whole or floating-point number
reyn	whole or floating-point number
dt	whole or floating-point number
pas	whole number greater than or equal to one

Output

List of coordinates in five dimensions.

A model obtained by an appropriate truncation to five modes of the Navier-Stokes equations for an incompressible fluid in a torus.

dx1= -2*x1 + 4*x2*x3 + 4*x4*x5

dx2= -9*x2 + 3*x1*x3

dx3= -5*x3 - 7*x1*x2 + *reyn*

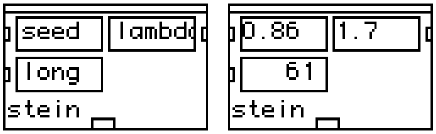
dx4= -5*x4 - x1*x5

dx5= -x5 - 3*x1*x4

• *reyn* is the Reynolds number, which has a certain number of interesting behaviors in function of different values of *reyn*. For the different critical values of *reyn*, the most remarkable point is the stochastic behavior observed when R1 *reyn* R2.
With 28.73 R1 29.0 and R2 ± = 33.43.

• *x1*, *x2*, *x3*, *x4* et *x5* are the initial coordinates
• *pas* is the number of iterations, or generated points;
• *dt* is a value of time for the numerical integration in the equations.
The output of this module is a list of coordinates in five dimensions :
$((x_{1\text{-}in}\ x_{2\text{-}in}\ x_{3\text{-}in}\ x_{4\text{-}in}\ x_{5\text{-}in})\ ...\ (x_{1n}\ x_{2n}\ x_{3n}\ x_{4n}\ x_{5n}))$.

1.9 stein



Syntax

(alea::stein *seed lambda long*)

Inputs

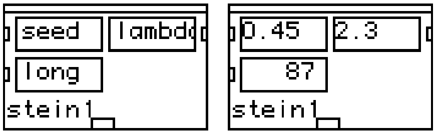
seed	whole or floating-point number
lambda	whole, floating-point number or list of values
long	whole number greater than or equal to one

Output

list

Iterative quadratic equation:
X_{n+1}=lambda*sin(pi*X_n)
• *lambda* is a number or a list of parameters which define the 'turbulence' of the generated values.
• *seed* is an initial value between zero and one;
• *long* is the length of the list generated, which is equivalent to the number of iterations.
The output of this module is a list of values for each iteration.
☛ see the article : M. Feigenbaum. "Universal Behavior in Nonlinear Systems."

1.10 stein1



Syntax

(alea::stein1 *seed lambda long*)

Inputs

seed	whole or floating-point number
lambda	whole, floating-point number or list of values
long	whole number greater than or equal to one

Output

list

Iterative quadratic equation:

X_{n+1}=lambda*x_n^2*sin(pi*X_n)

Variation of the equation Xn+1=lambda*sin(pi*Xn).

• *lambda* is a number or a list of parameters which define the 'turbulence' of the generated values;

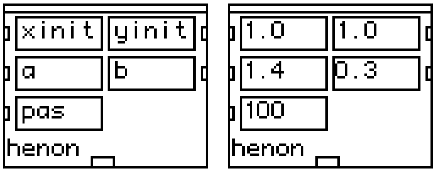
• *seed* is an initial value between zero and one;

• *long* is the length of the list generated, which is equivalent to the number of iterations.

The output of this module is a list of values for each iteration.

☛ see the article : M. Feigenbaum. "Universal Behavior in Nonlinear Systems."

1.11 henon



Syntax

(alea:henon *xinit yinit* a *b pas*)

Inputs

xinit	whole or floating-point number
yinit	whole or floating-point number
a	whole or floating-point number close to 1.4
b	whole or floating-point number close to 0.3
pas	whole number greater than or equal to one

Output

list of coordinates in two dimensions

This model is a simplified version of the Lorentz dynamic system. It was suggested by the French astronomer Michel Hénon in 1976.

$X_{n+1} = y_n - a*x_n^2 + 1$

$Y_{n+1} = b*x_n$

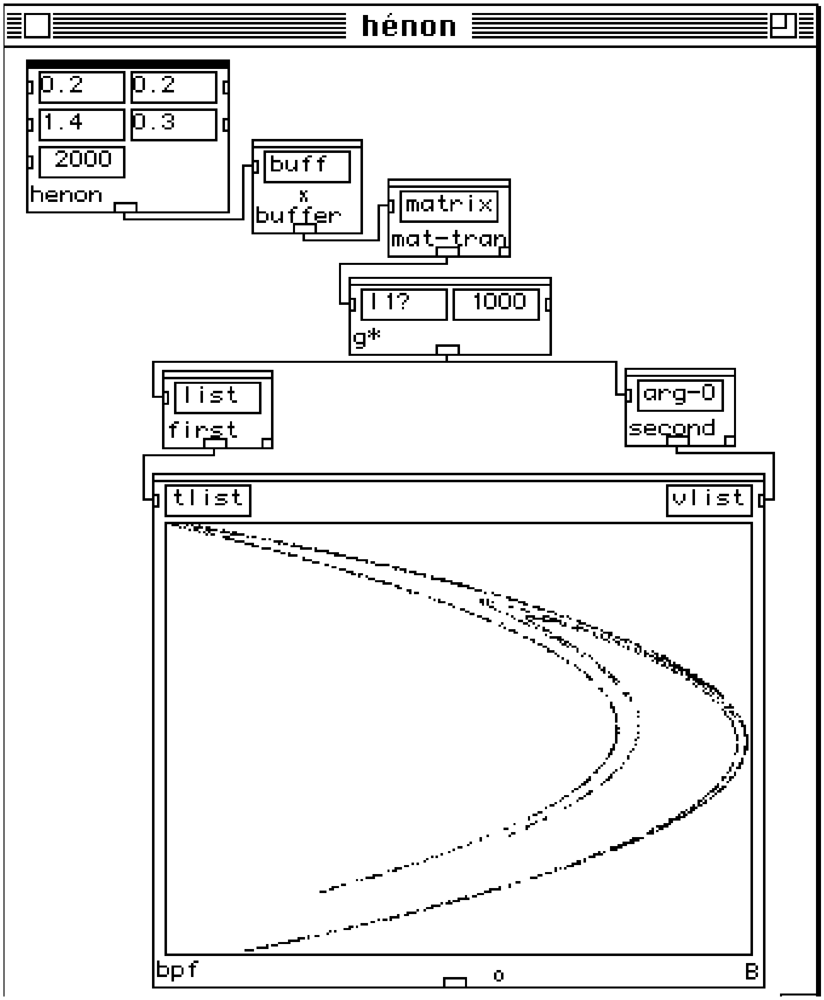
with a = 1.4 and b = 0.3

- *xinit* and *yinit* are the initial values;
- *a* and *b* are the system parameters;
- *pas* is the number of iterations, or generated points.

The output of this module is a list of coordinates in two dimensions :

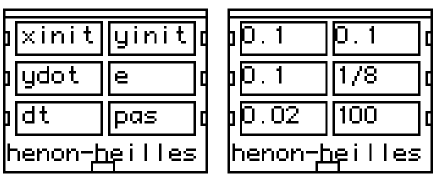
$((x_{init}\ y_{init})\ (x_0\ y_0)\ (x_1\ x_2)\ ...\ (x_n\ y_n))$

✒ see the article : D. Ruelle. "Strange Attractors."



The module **g*** to scale the data, thus clarifying the display.

1.12 henon-heilles



Syntax

(alea:henon-heilles *xinit yinit ydot e dt pas*)

Inputs

xinit	whole or floating-point number
yinit	whole or floating-point number
ydot	whole or floating-point number close to 1.4
e	positive whole or floating-point number less than or equal to 1/6
dt	whole or floating-point number
pas	whole number greater than or equal to one

Output

list of coordinates in four dimensions

This system was originally introduced as a simplified model of the individual movement of a star within a gravitational field:

$$\frac{dx^2}{dt^2} = -x - 2xy$$

$$\frac{dx^2}{dt^2} = -y + y^2 - x^2$$

where

x and y are the star's coordinates,

E is the total energy of the system,

$$E = \frac{1}{2}\left(x^2 + y^2 + 2x^2y - \frac{2}{3}y^3\right) + \frac{1}{2}(dx^2 + dy^2)$$

The maximum permitted value for E is 1/6.

• *xinit*, *yinit* and ydot are the initial values;

• *E* is the value of the total energy;

• *dt* is a value of time for the numerical integration in the equations;

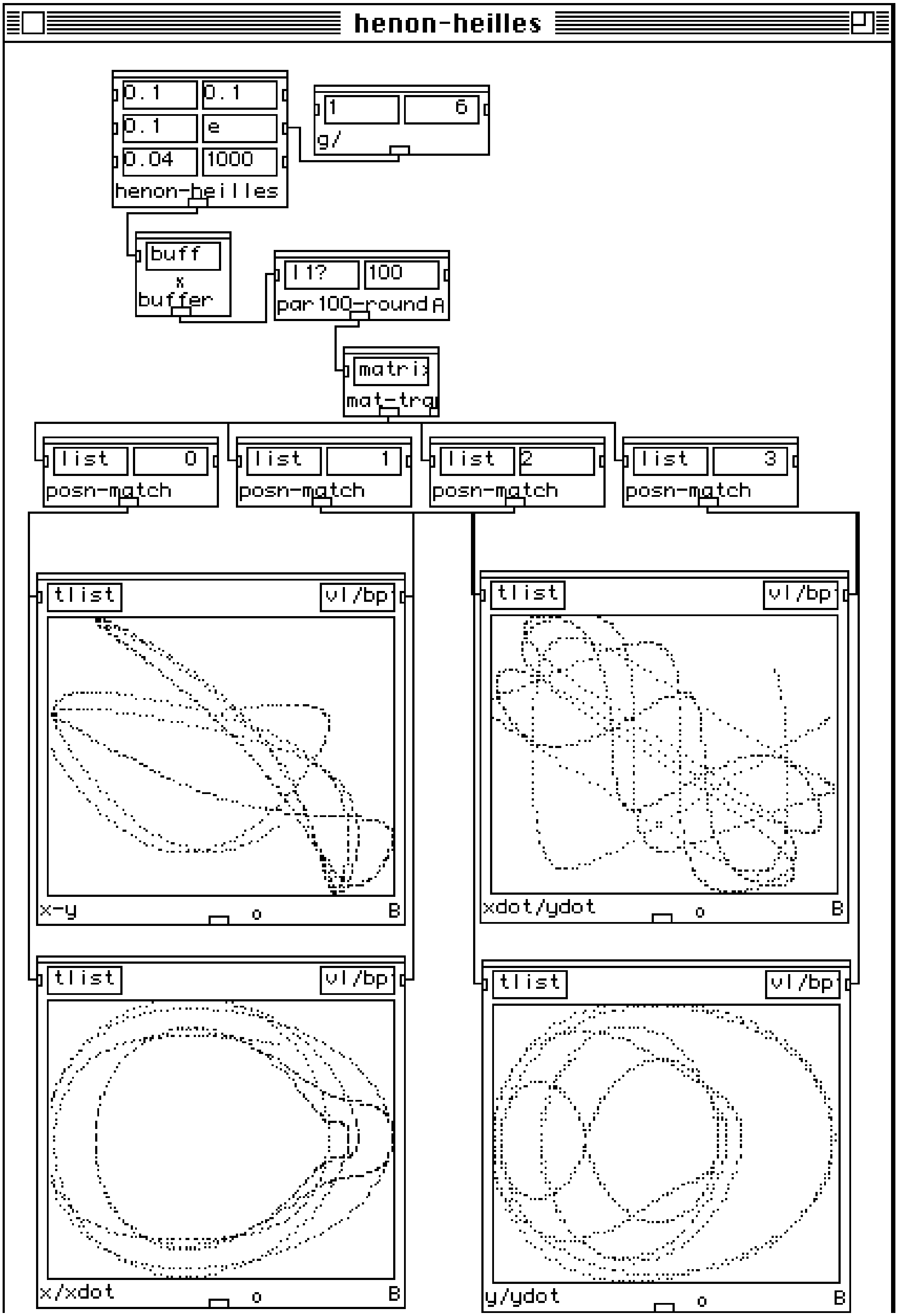
• *pas* is the number of iterations, or generated points.

The output of this module is a list of coordinates in four dimensions :

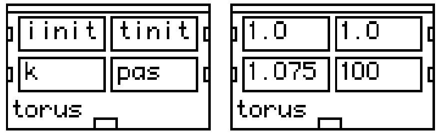
$((x_{init}\ y_{init}\ x_{dot}\ y_{dot})\ (x_0\ y_0\ x_{dot0}\ y_{dot0}\ )\ (x_1\ x_2\ x_{dot1}\ y_{dot2})\ ...\ (x_n\ y_n\ x_{dotn}\ y_{dotn}))$.

☛ See R. Bidlack, "Chaotic Systems as Simple (but Complex) compositional Algorithms." et R. Helleman, "Self-Generated Chaotic Behavior in Nonlinear Mechanics."

Here is an example where the output list was formatted to construct the various phase-planes :


henon-heilles
0.1
0.1
0.1
e
0.04
1000
henon-heilles
1
6
g/
buff
x
buffer
l1?
100
par100-round
A
matrix
mat-trą
list
0
posn-match
list
1
posn-match
list
2
posn-match
list
3
posn-match
tlist
vl/bp
x-y
o
B
tlist
vl/bp
xdot/ydot
o
B
tlist
vl/bp
x/xdot
o
B
tlist
vl/bp
y/ydot
o
B

1.13 torus



Syntax

(alea:torus *iinit tinit k pas*)

Inputs

iinit	whole or floating-point number modulo 2*pi
tinit	whole or floating-point number modulo 2*pi
k	whole or floating-point number
pas	whole number greater than or equal to one

Output

list of coordinates in two dimensions

This equation system is derived from a model of a pendulum submitted to periodic perturbations :

$I_{n+1} = I_n + K * \sin T_n$

$T_{n+1} = T_n + I_{n+1}$

where

• *k* is a parameter of perturbation;

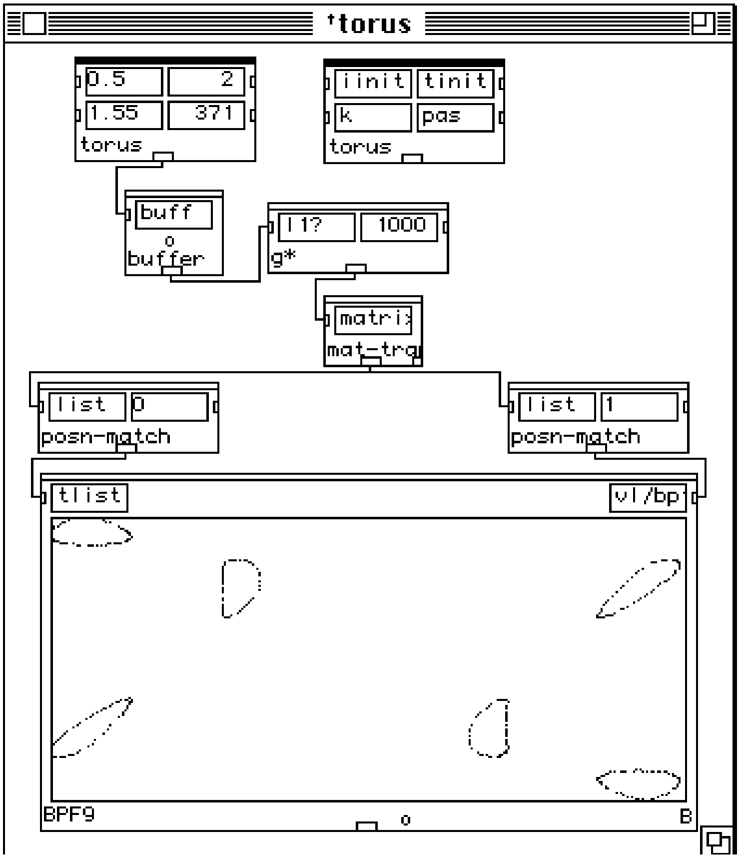
• *I* and *T* are the variables of the phase-space in modulo 2*pi between 0 and 2*pi;

• *init* and *tinit* are the initial values k is the parameter of perturbation *pas* is the number of iterations, or generated points.

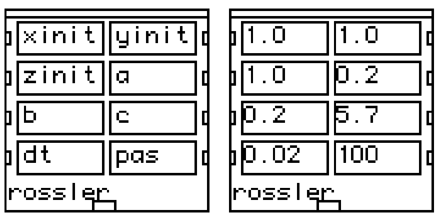
The output of this module is a list of coordinates in two dimensions :

((Iinit tinit) (I_0 T_0) (I_1 T_2) ... (I_n T_n)).

☛ See R. Bidlack, "Chaotic Systems as Simple (but Complex) compositional Algorithms."



1.14 rossler



Syntax

(alea::rosslereq *x y z* a *b c*)

Inputs

xinit whole or floating-point number

yinit whole or floating-point number

zinit	whole or floating-point number
a	whole or floating-point number
b	whole or floating-point number
c	whole or floating-point number
dt	whole or floating-point number
pas	whole number greater than or equal to one

Output

list of coordinates in three dimensions

The Rossler equation system is an artificial system which was created solely to be a simple model for studying a strange attractor. The following are the systems equations :

$$\frac{dx}{dt} = -(y + z)$$

$$\frac{dx}{dt} = x + ay$$

$$\frac{dx}{dt} = b + xz - cz$$

• *xinit*, *yinit* and *zinit* are the initial coordinates;

• *pas* is the number of iterations, or generated points;

• *a*, *b* and *c* are the system parameters;

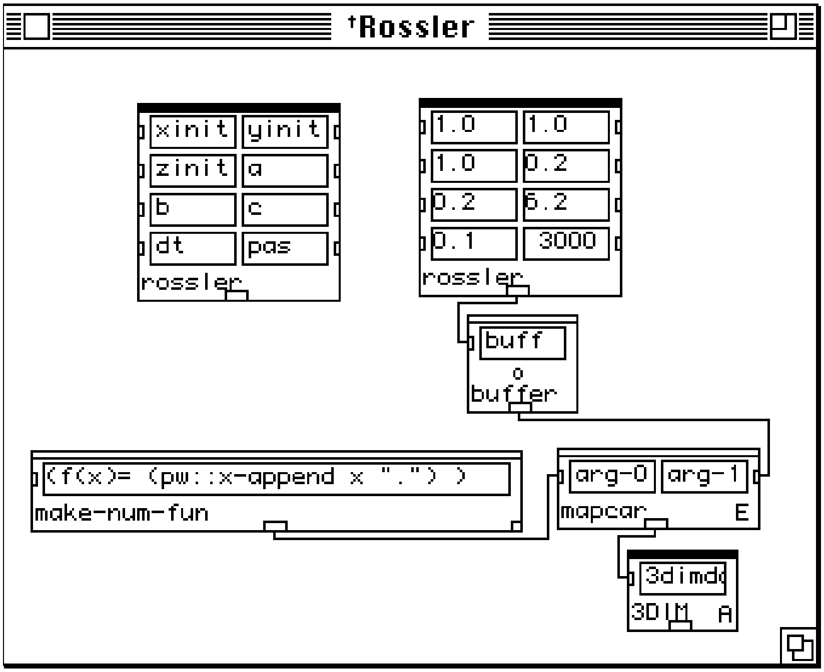
• *dt* is a value of time for the numerical integration in the equations.

The output of this module is a list of coordinates in three dimensions :
((xinit yinit zinit) $(x_0\ y_0\ z_0)$ $(x_1\ x_2\ x_3)$ ... $(x_n\ y_n\ z_n)$)

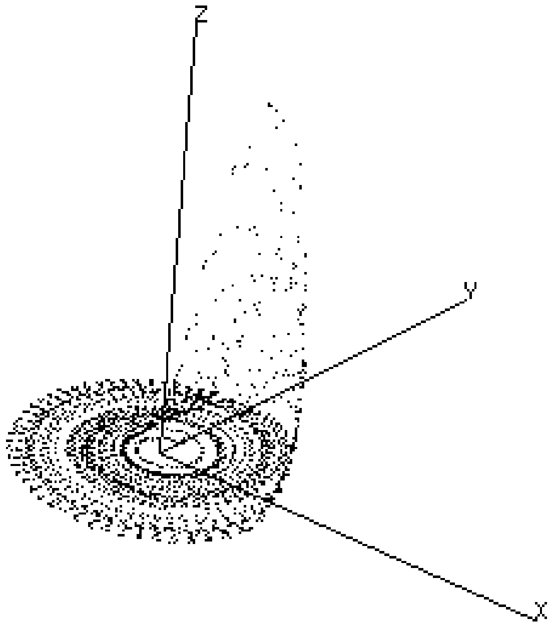
☛ See O. Rossler, "An equation for Continuous Chaos."

Here is an example patch.

☛ The library 3Dim-disp must be loaded before opening this patch so as to give access to the module **3dim**, for the three dimensional display.



☛ The modules **make-num-fun** and **mapcar** (Lisp functions) are used here to add the character ’point’ to each sub-list of coordinates (see the 3Dim-disp library's documentation) to make the display easier to understand.



1.15 ginger

Syntax

(alea:ginger *xinit yinit cr pas*)

Inputs

xinit	whole or floating-point number
yinit	whole or floating-point number
cr	whole or floating-point number between zero and one
pas	whole number greater than or equal to one

Output

list of coordinates in two dimensions

Iterative equation system :

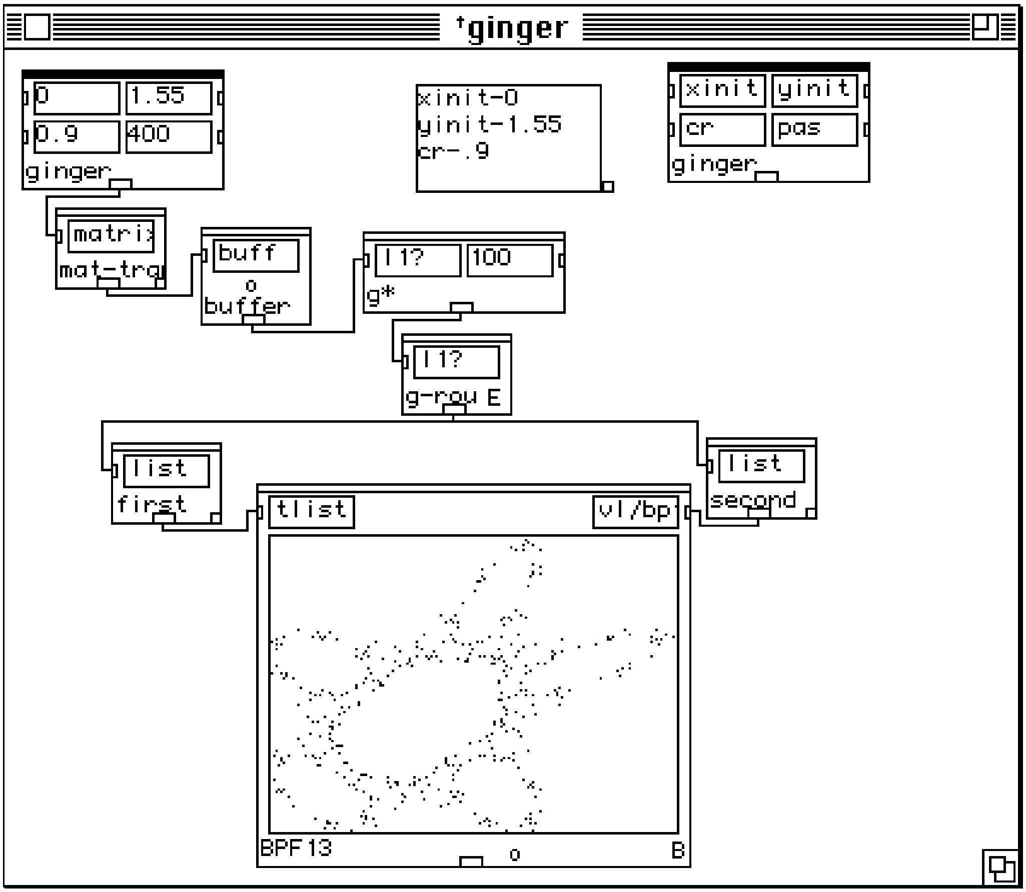
$X_{n+1} = 1 - y_n - cr*(\text{abs } x)$

$Y_{n+1} = x_n$

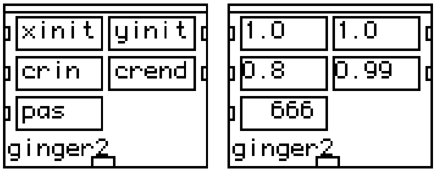
where

• *xinit* and *yinit* are the initial values *cr* is a control parameter between zero and one, and *pas* is the number of iterations, or generated points. The output of this module is a list of coordinates in two dimensions :

((xinit yinit) (x_0 y_0) (x_1 x_2) ... (x_n y_n)) :



1.16 ginger2



Syntax

(alea:ginger2 *xinit yinit crin crend pas*)

Inputs

xinit	whole or floating-point number
yinit	whole or floating-point number
crin	whole or floating-point number between zero and one
crend	whole or floating-point number between zero and one
pas	whole number greater than or equal to one

Output

list of coordinates in two dimensions

Iterative equation system :

$X_{n+1} = 1 - y_n - cr*(abs\ x)$

$Y_{n+1} = x_n$

with an evolving control parameter *cr* where :

• *xinit* and *yinit* are the initial values;

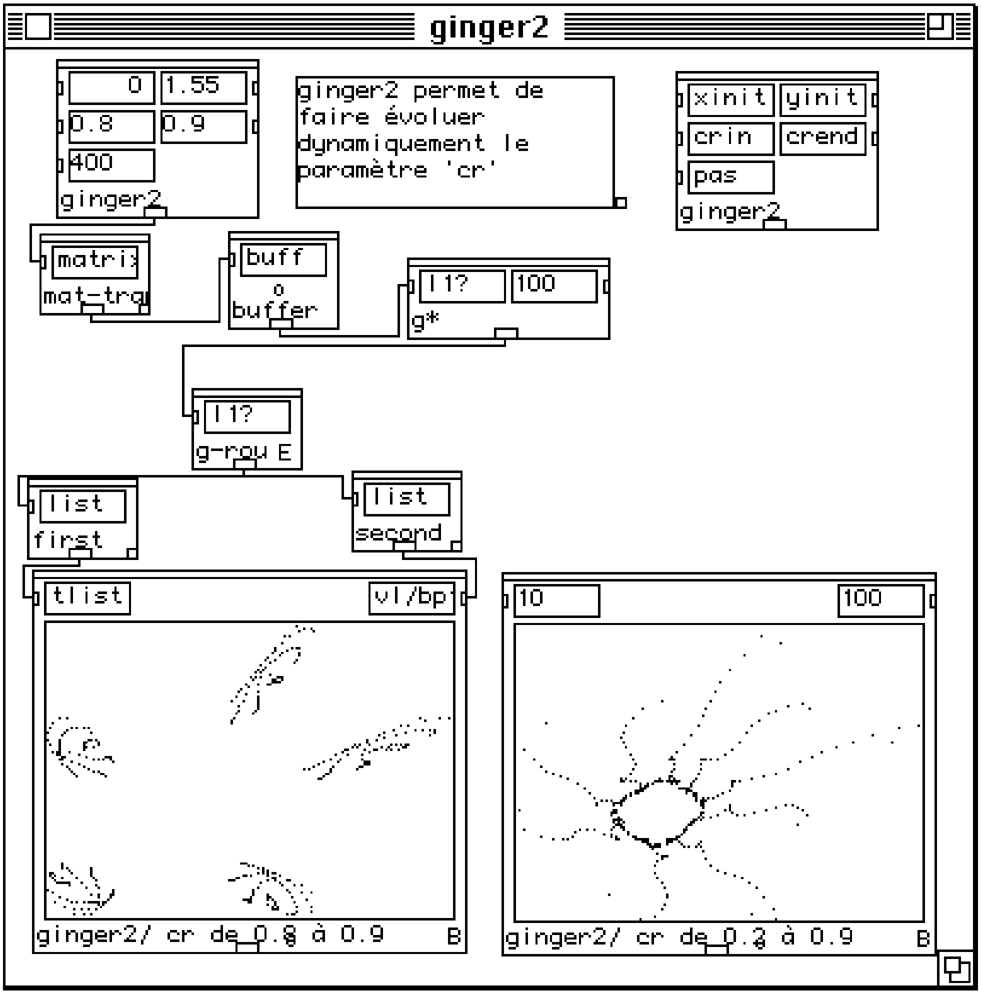
• *crin* is an initial control parameter between zero and one;

• *crend* is a final control parameter between zero and one. As the evolution of the system is calculated, the value for the control parameter *cr* will be interpolated between *crin* and *crend*;

• *pas* is the number of iterations, or generated points.

The output of this module is a list of coordinates in two dimensions :

$((xini_t\ yinit)\ (x_0\ y_0)\ (x_1\ x_2)\ ...\ (x_n\ y_n))$


ginger2
0
1.55
0.8
0.9
400
ginger2
ginger2 permet de faire évoluer dynamiquement le paramètre 'cr'
xinit
yinit
crin
crend
pas
ginger2
matrix
mat-trq
buff
buffer
l 1?
100
g*
l 1?
g-rou E
list
first
list
second
tlist
vl/bp
ginger2/ cr de 0.8 à 0.9
B
10
100
ginger2/ cr de 0.2 à 0.9
B

3 IFS

The functions in IFS are systems of iterative linear equations.

If W is an iterative system where :

$$W = \sum_{i=1}^{n} w_i$$

each equation w is in the following form :

$$w(x,y) = (ax + by + e, cx + dy + f)$$

It is possible to represent these equations in a matrix form :

$$w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix} + t$$

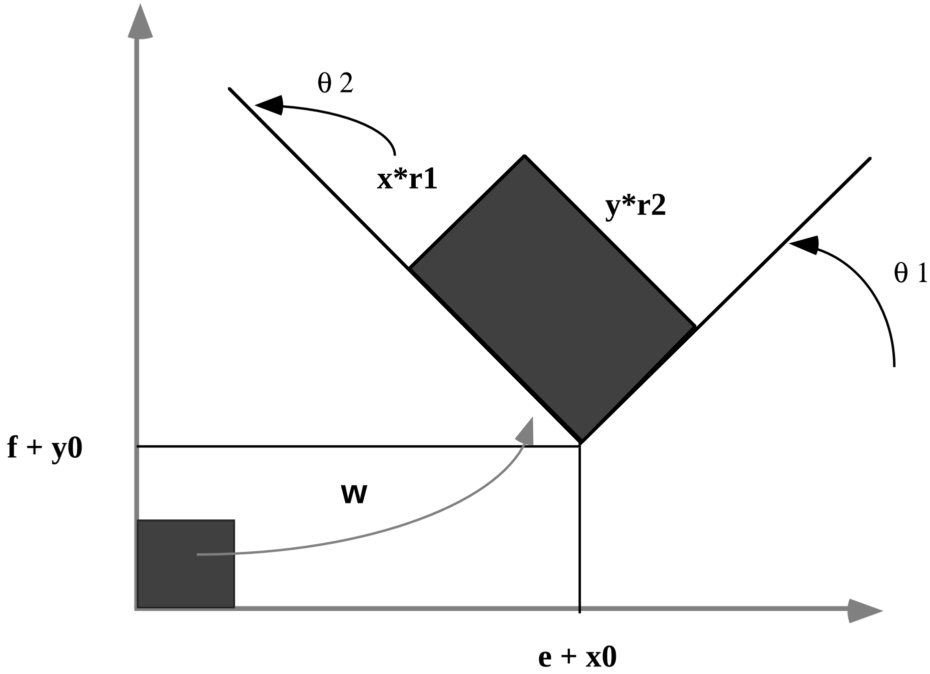
where ‘t’ is the translation matrix of the points ‘x’ and ‘y’, and ‘A’ the rotation and contraction matrix of the space. The matrix ‘A’ may be visualized in a polar form, which would clarify the incidence of each one of its components where :

où :

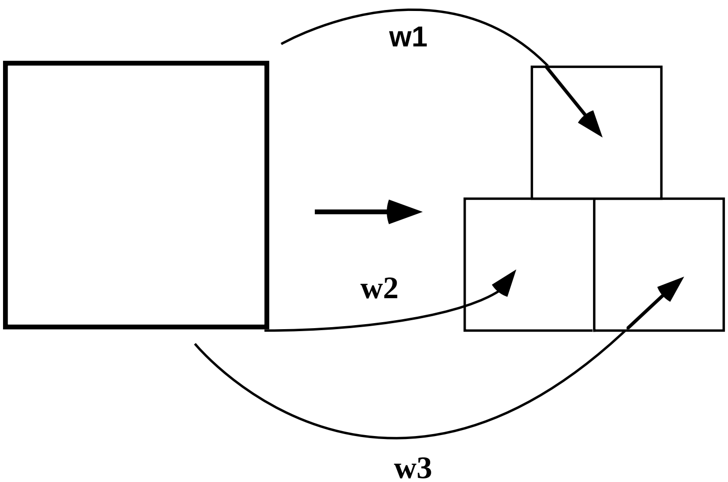
$$A = \begin{bmatrix} r1\cos\theta 1 & -r2\cos\theta 2 \\ r1\sin\theta 1 & r2\sin\theta 2 \end{bmatrix}$$

r1 and r2 are the contraction factors of the x and y axes, respectively. q1 and q2 are the angular offsets for the x and y axes, also respectively.

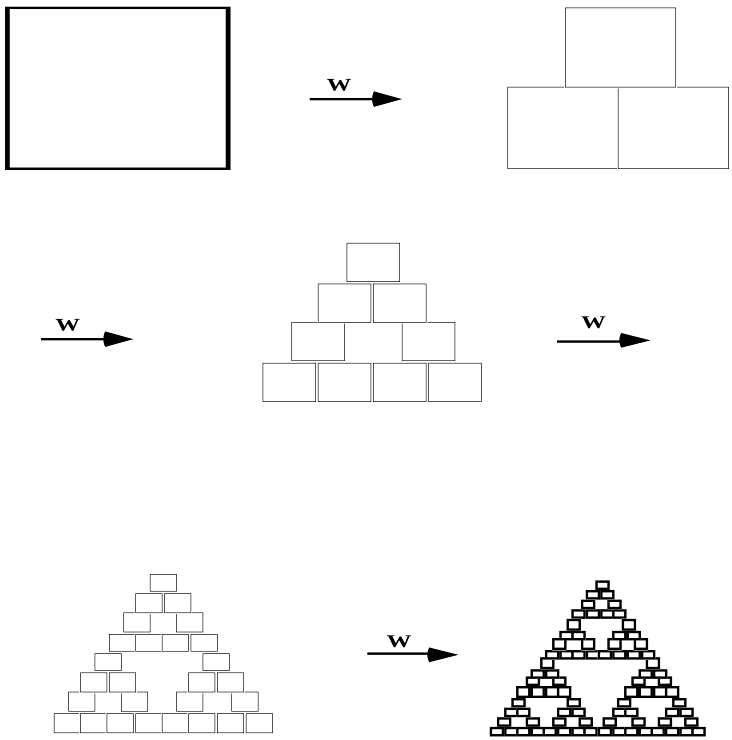
The application of a function w (a single iteration) on an object, for example a square, will produce the following effect:



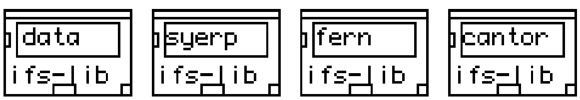
A system will normally be the result of two or more equations applied together to a given object :



The repeated application of this group of equations will often converge toward particular attractors :



2.1 ifs-lib



Syntax

(alea::ifs-lib *data*)

Inputs

data scrolling menu options

Output

list of data to be connected to the input *data* of the **ifsx** module

Library of data for use with the module IFSx The input of this module is a list of menu options which allow the user to select a particular model of linear transformation. The output of this module is a list containing seven sub-lists. It should be noted that each transformation is composed of two matrices

$$A = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \text{ et } t = \begin{bmatrix} e_1 \\ f_1 \end{bmatrix}$$ and one associated probability p1

where A is a space transformation and t is a translation.

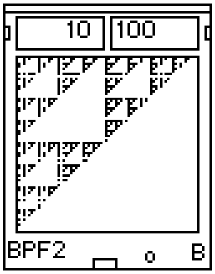
The output list corresponds to seven groups of data :

$((a_1\ a_2\ a_3\ ...\ a_n)\ (b_1\ b_2\ b_3\ ...\ b_n)\ (c_1\ c_2\ c_3\ ...\ c_n)\ (d_1\ d_2\ d_3\ ...\ d_n)$
$(e_1\ e_2\ e_3\ ...\ e_n)\ (f_1\ f_2\ f_3\ ...\ f_n)\ (p_1\ p_2\ p_3\ ...\ p_n))$,

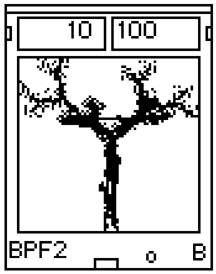
where ‘n’ is the number of transformation which make up the system.

The module **ifs-lib** offers 19 basic models, each with its own attractor:

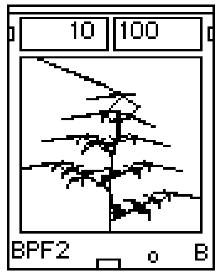
syerpinsky *(called* **syerp***)*



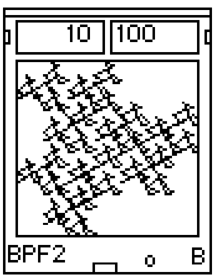
tree0



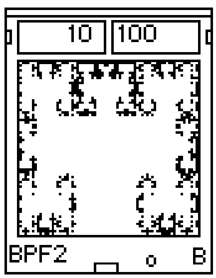
fern



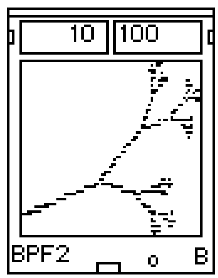
dragon *(called* **drag***)*



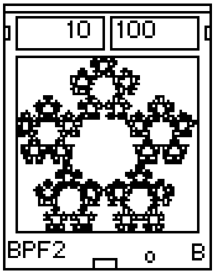
cantor



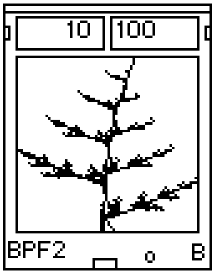
twig



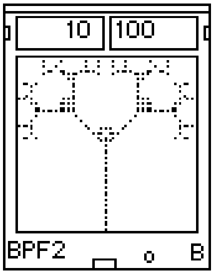
cristal



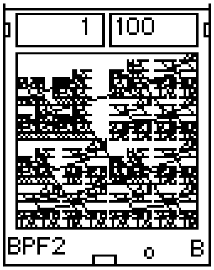
fern1



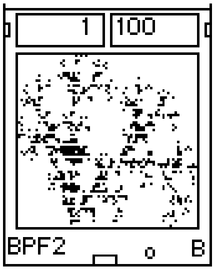
tree1



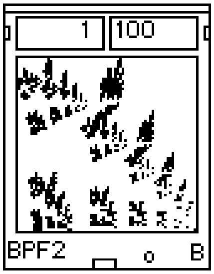
castle



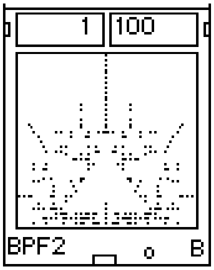
cloud



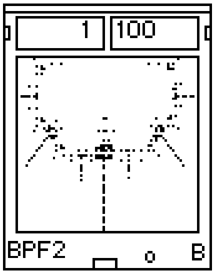
frnsqr



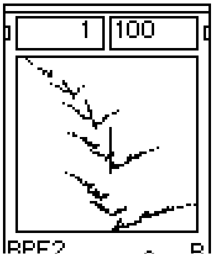
jewel



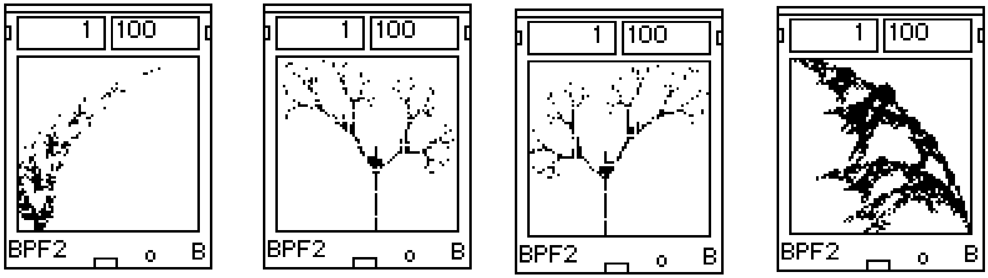
jewel2



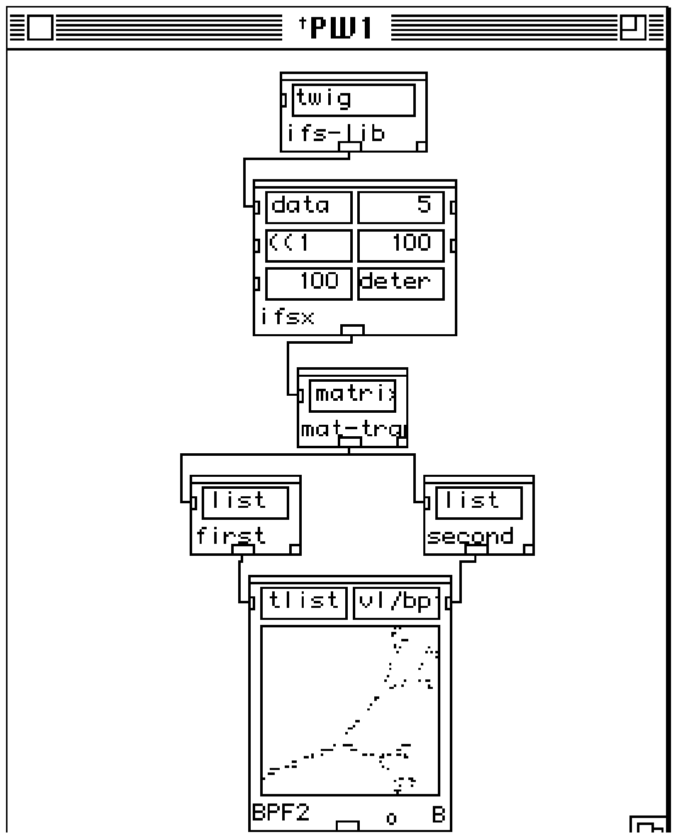
frntre7



fern2 **plant1** **plant2** **mountain**

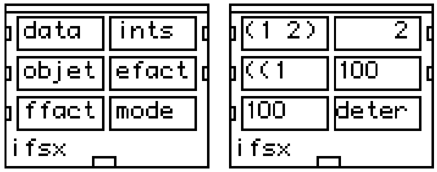


Several models among the ones shown here have been conceived by Mikael Laurson. To construct the linear transformations, it is strongly advised to use the modules **make-w**, **make3-w** and **app-w-trans**. The configuration used to make the figures shown above is the following :



using as initial data the default value ((1 1)).

2.2 ifsx



Syntax

(alea:ifsx *data ints objet efact ffact mode*)

Inputs

data	list with seven sub-lists (see the **ifs-lib** module)
ints	whole number greater than or equal to one
objet	list of lists, or a BPF object
	patch-work::*c*-break-point-function
efact	whole or floating-point number
ffact	whole or floating-point number
mode	menu options

Output

list of coordinates in two dimensions of the transformed object

• *objet* is a list of lists containing the coordinates of an object (a figure) or a BPF with a geometric object;

• *ints* is the number of desired iterations;

• *data* is a list of lists containing the data for the linear transformations. To this input it is possible to connect either a module **ifs-lib**, or a module **make-w** (which allows the user to construct personalized linear transformations), or a module **make3-w** (which is the equivalent of three make-w modules) or a module **app-W-trans** (used to group multiple **make-w** modules;

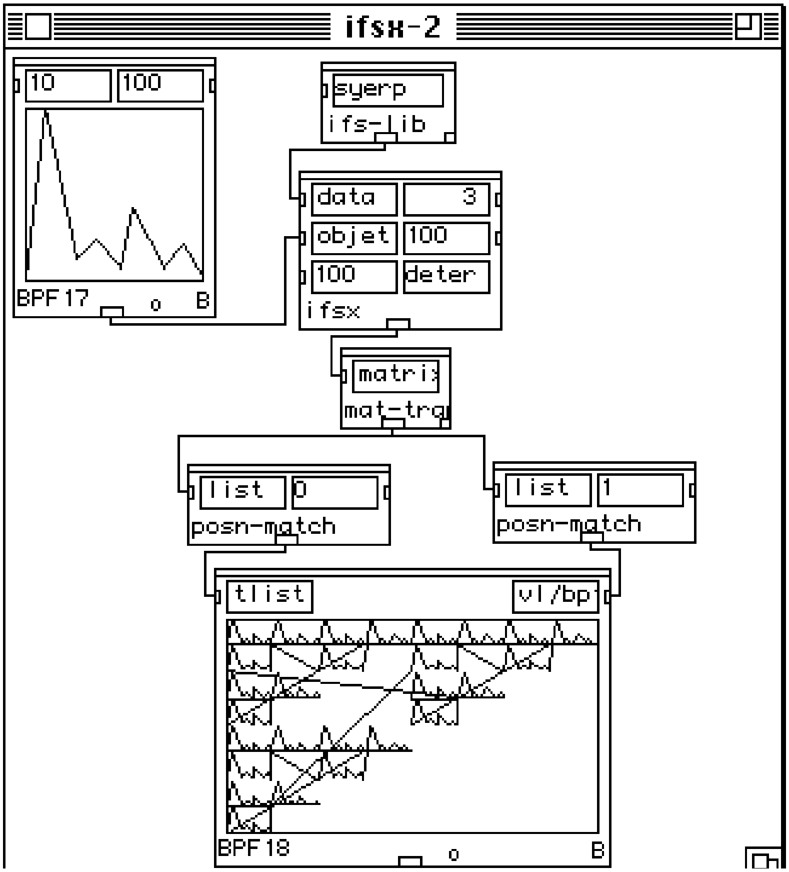
• *efact* is a multiplicative factor for the horizontal translation;

• *ffact* is a multiplicative factor for the vertical translation;

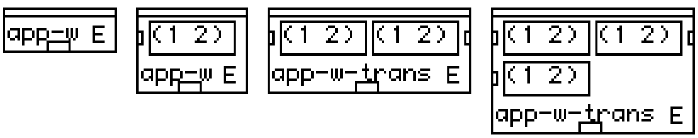
• *mode* is in fact a list of menu options which allow the user to chose the way in which the module will function: either deterministicly or probalisticly .

The output of this module is a list of coordinates in two dimensions of the transformed object:

($(x_0\ y_0)\ (x_1\ x_2)\ ...\ (x_n\ y_n)$).



2.3 app-w-trans



Syntax

(alea::app-w-trans *&rest list*)

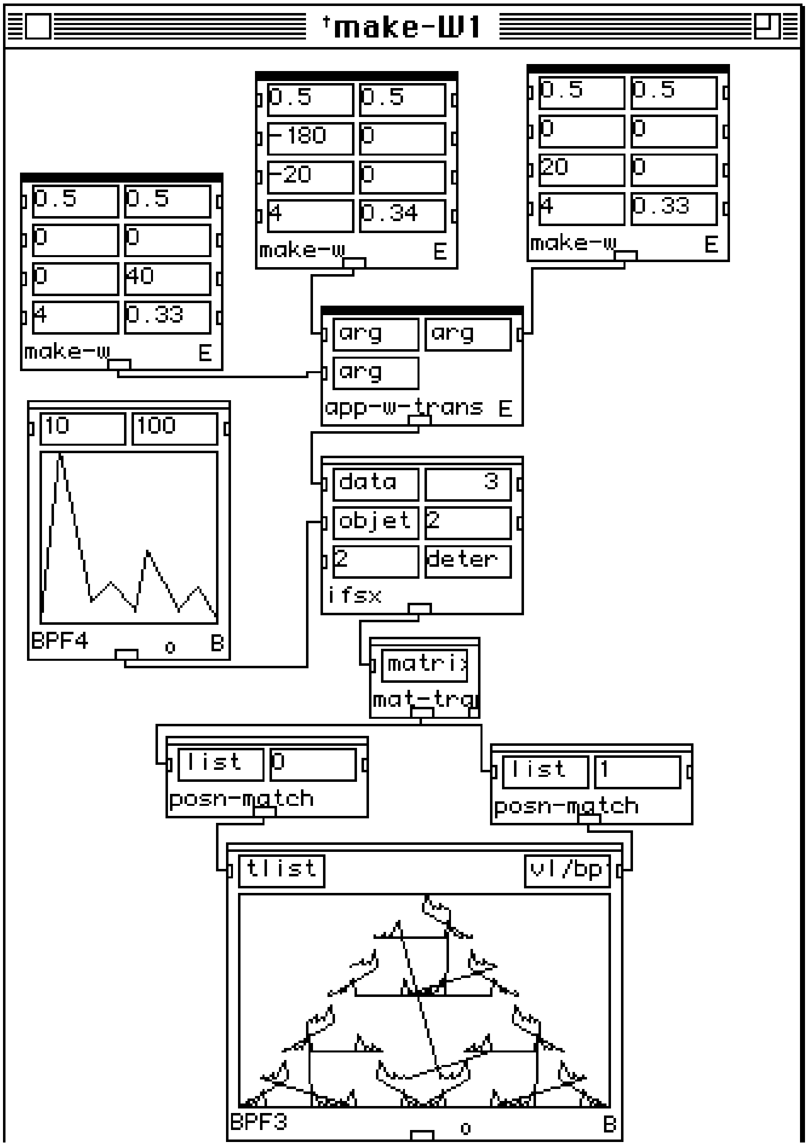
Inputs

list list of lists (the output of a **make-w** module)

Output

list of lists (parameters to be connected to the input *data* of the **ifsx** module)

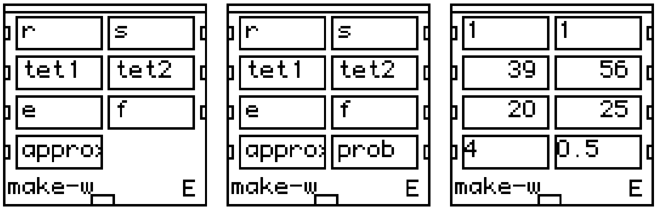
This module is used to group together two or more **make-w** modules to construct a system of linear transformation[1] :



☛ when first placed in a patch window this module is 'closed' the needed number of inputs must then be 'opened'.

1. In this example, a **BPF** module has been used to introduce a figure which will be transformed by the system formed by the gathering of the three **make-w** modules.

2.4 make-w



Syntax

(alea::make-w *r s tet1 tet2 e f approx &optional prob*)

Inputs

r	whole or floating-point number
s	whole or floating-point number
tet1	whole or floating-point number
tet2	whole or floating-point number
e	whole or floating-point number
f	whole or floating-point number
approx	whole number greater than or equal to zero
&optional	
prob	floating-point number between zero and one (a probability)

Output

list of lists (parameters to be connected to the input *data* of the **ifsx** module)

Constructs a matrix for a linear transformation.

• *r* is the coefficient of contraction for the x axis;

• *s* is the coefficient of contraction for the y axis;

• *tet1* is the angular offset for the x axis;

• *tet2* is the angular offset for the y axis;

• *e* is the horizontal translation;

• *f* is the vertical translation;

• *prob* is a probability effecting the linear transformation in the case of a stochastic system transformation;

• *approx* is the number of decimal places to be included in the output data (in the matrix).

Note that each transformation is composed of two matrices:

$$A = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \text{ et } t = \begin{bmatrix} e_1 \\ f_1 \end{bmatrix}$$

and an associated probability p1

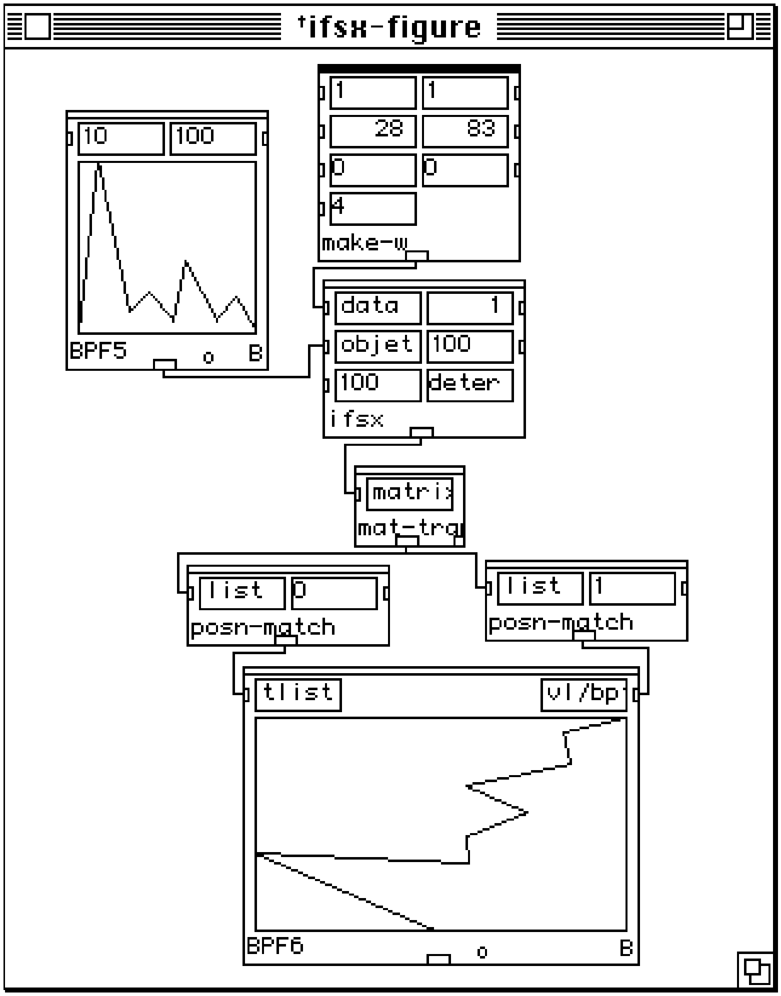
where A is a space transformation and t is a translation.

It is possible to rewrite the matrix A as follows :

$$A = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} = \begin{bmatrix} r \cdot (\cos tet1) & -s \cdot (\sin tet2) \\ r \cdot (\sin tet1) & s \cdot (\cos tet2) \end{bmatrix}$$

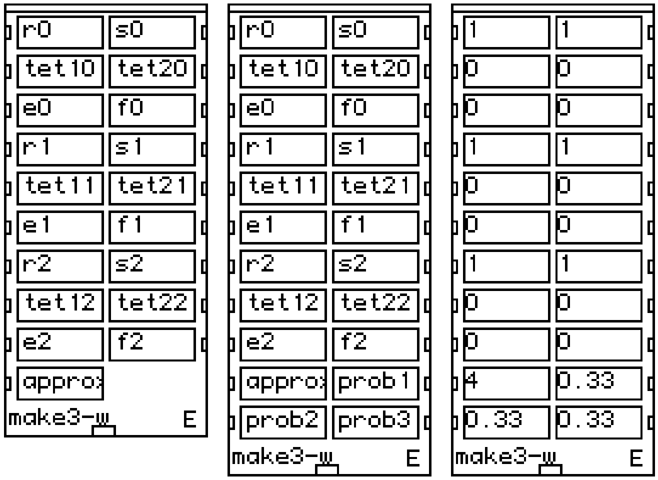
where *r* and *s* are the contraction factors of the x and y axes, respectively. *tet1* and *tet2* are the angular offsets for the x and y axes, also respectively (See the introduction to this chapter).

This module may be used singly or in combination (see **app-w-trans**).



†ifsx-figure
1
1
28
83
0
0
4
make-w
10
100
BPF5
o
B
data
1
objet
100
100
deter
ifsx
matri
mat-tra
list
0
posn-match
list
1
posn-match
tlist
vl/bp
BPF6
o
B

2.5 make3-w



Syntax

(alea::make3-w *r0 s0 tet10 tet20 e0 f0 r1 s1 tet11 tet21 e1 f1 r2 s2 tet12 tet22 e2 f2 approx &optional prob1 prob2 prob3*)

Inputs

r 0 whole or floating-point number (pour la transformation 1)

s 0 whole or floating-point number (pour la transformation 1)

tet10 angle in degrees (for transformation 1)

tet20 angle in degrees (for transformation 1)

e 0 whole or floating-point number (for transformation 1)

f 0 whole or floating-point number (for transformation 1)

r 1 whole or floating-point number (for transformation 2)

s 1 whole or floating-point number (for transformation 2)

tet11 angle in degrees (for transformation 2)

tet21 angle in degrees (for transformation 2)

e 1 whole or floating-point number (for transformation 2)

f 1 whole or floating-point number (for transformation 2)

r 2 whole or floating-point number (for transformation 3)

s 2 whole or floating-point number (for transformation 3)

tet12	angle in degrees (for transformation 3)
tet2 2	angle in degrees (for transformation 3)
e 2	whole or floating-point number (for transformation 3)
f 2	whole or floating-point number (for transformation 3)
approx	whole number greater than or equal to zero
&optional	
prob1	floating-point number between zero and one (a probability for transformation 1)
prob2	floating-point number between zero and one (a probability for transformation 2)
prob3	floating-point number between zero and one (a probability for transformation 3)

Output

list of lists (parameters to be connected to the input *data* of the **ifsx** module)

Constructs a matrix for a system of three linear transformations, where:

- *r* is the coefficient of contraction for the x axis;
- *s* is the coefficient of contraction for the y axis;
- tet1 is the angular offset for the x axis;
- tet2 is the angular offset for the y axis;
- en is the horizontal translation;
- *fn* is the vertical translation;
- *prob* is a probability effecting the linear transformation in the case of a stochastic system transformation;
- *approx* is the number of decimal places to be included in the output data (in the matrix).

Note that each transformation is composed of two matrices

$$A = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \text{ et } t = \begin{bmatrix} e_1 \\ f_1 \end{bmatrix}$$

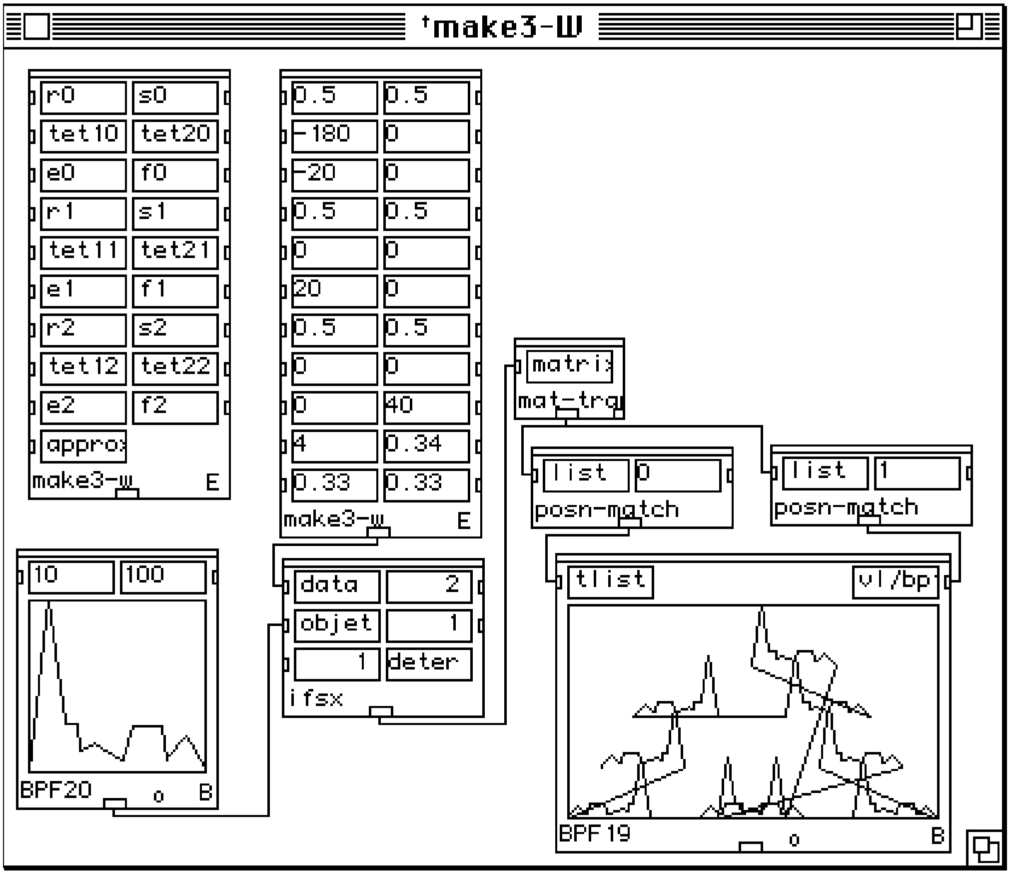
and an associated probability p1

where A is a space transformation and t is a translation.

It is possible to rewrite the matrix A as follows :

$$A = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} = \begin{bmatrix} r \cdot (\cos tet1) & -s \cdot (\sin tet2) \\ r \cdot (\sin tet1) & s \cdot (\cos tet2) \end{bmatrix}$$

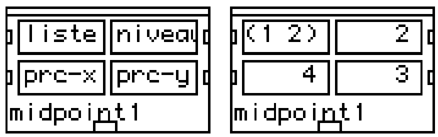
where *r* and *s* are the contraction factors of the x and y axes, respectively. tet1 and tet2 are the angular offsets for the x and y axes, also respectively.



4 Fractus

This section contains three algorithms for the construction of fractal curves.

3.1 midpoint1



Syntax

(alea:midpoint1 *list1* niveaux prc-x prc-y)

Inputs

list1	list of lists, or a BPF *objet*
niveaux	whole number greater than or equal to one
prc-x	whole or floating-point number
prc-y	whole or floating-point number

Output

list of coordinates in two dimensions

Constructs a list of points with their x and y locations based on the algorithm of movement of the mean.

• *list1* is a list of lists, where each sub-list is a pair of values indicating the coordinates of fixed points; *list1* may also be a BPF, in this case the coordinates of the points will be extracted and used as data;

• niveaux is a whole number which indicates the depth of the transformation of *list1*;

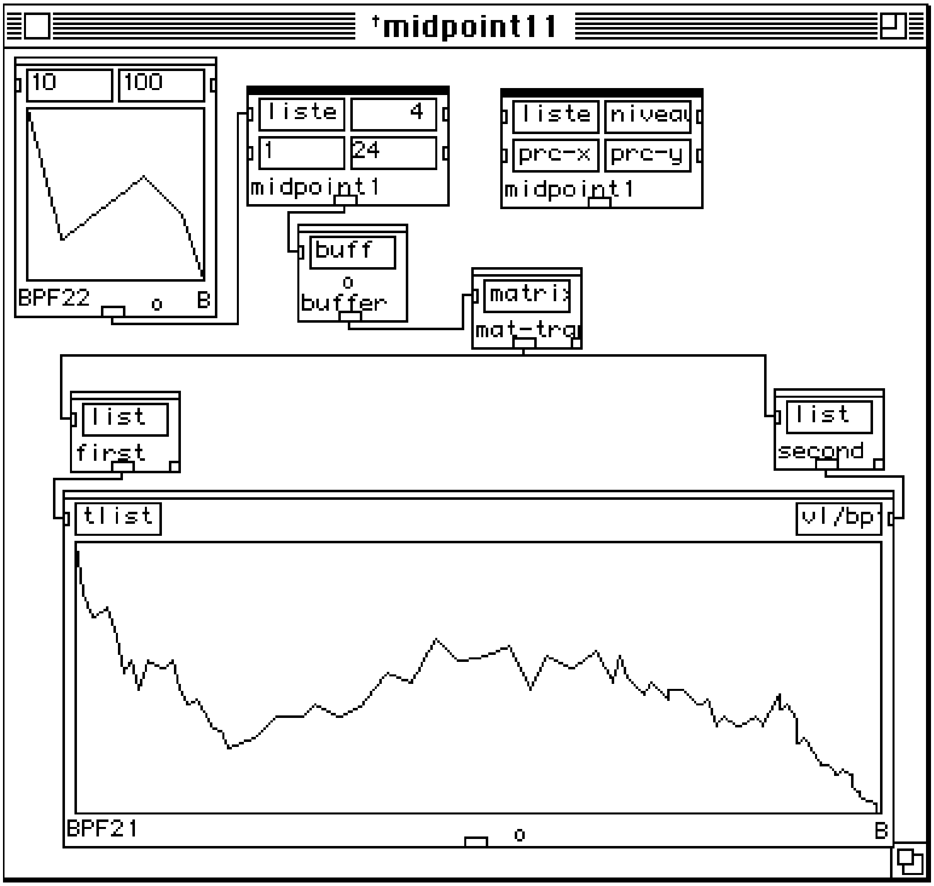
• prc-x is the percentage of random perturbation of the 'x' values;

• prc-y is the percentage of random perturbation of the 'y' values.

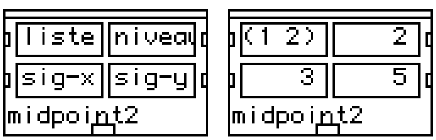
In this version the perturbation is based on a uniform distribution.

The output of this module is a list of coordinates in two dimensions of *list1* transformed : ((x_0 y_0) (x_1 x_2) ... (x_n y_n)).

Below is an example of the application of this algorithm on a curve contained within a BPF module :



3.2 midpoint2



Syntax

(alea:midpoint2 *list1 niveaux sig-x sig-y*)

Inputs

list1	list of lists, or a BPF object
niveaux	whole number greater than or equal to one
sig-x	whole or floating-point number
sig-y	whole or floating-point number

Output

list of coordinates in two dimensions

Constructs a list of points with their x and y locations based on the algorithm of movement of the mean.

• *list1* is a list of lists, where each sub-list is a pair of values indicating the coordinates of fixed points; *list1* may also be a BPF, in this case the coordinates of the points will be extracted and used as data;

• *niveaux* is a whole number which indicates the depth of the transformation of *list1*;

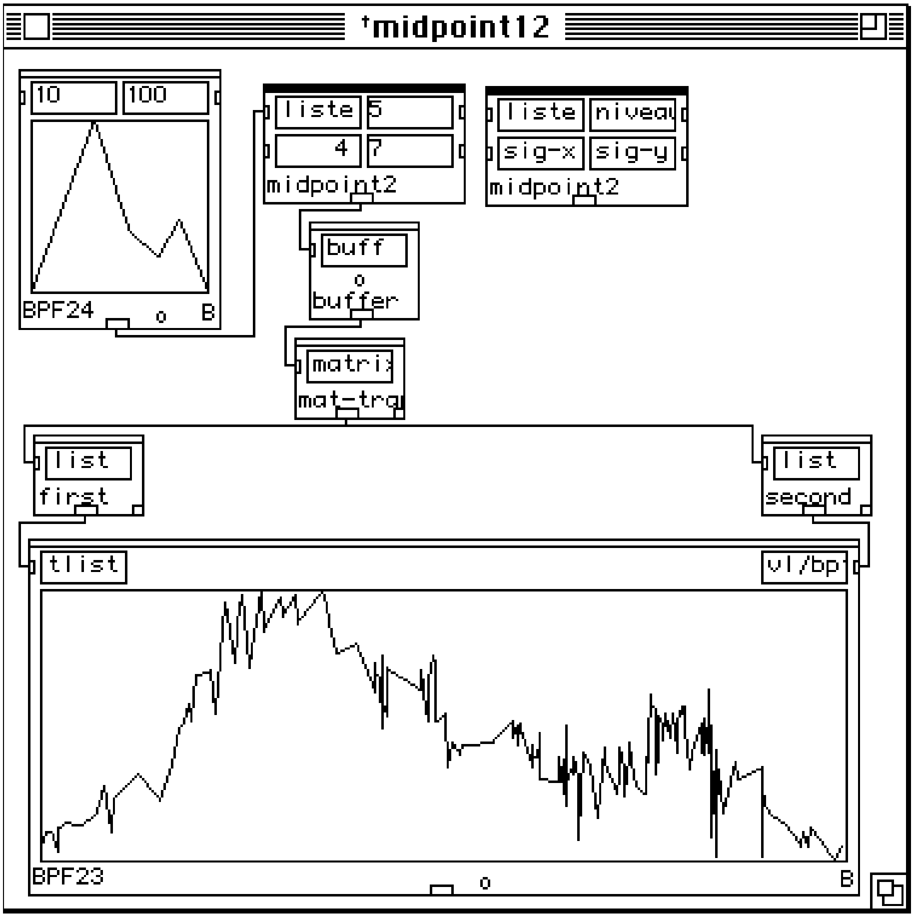
• *sig-x* is the parameter of dispersion for the gaussian variation introduced into the 'x' values;

• *sig-y* is the parameter of dispersion for the gaussian variation introduced into the 'y' values.

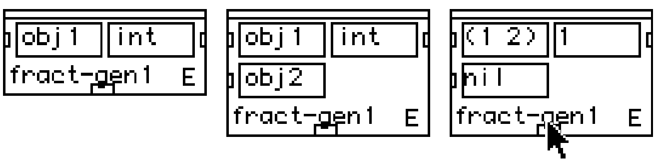
In this version the perturbation is based on a uniform distribution.

The output of this module is a list of coordinates in two dimensions of *list1* transformed : ((x_0 y_0) (x_1 x_2) ... (x_n y_n)).

Below is an example of the application of this algorithm on a curve contained within a BPF module :



3.3 fract-gen1



Syntax

(alea::fract-gen1 obj1 *int &optional obj2*)

Inputs

obj1 list of lists, or a BPF object

int whole number greater than or equal to one

&optional

obj2 list of lists, or a BPF object

Output

list of coordinates in two dimensions

Generates the coordinates of points on a fractal curve, based on graphical data.

• *obj1* is the pairs of coordinates or a BPF;

• *int* is the number of iterations;

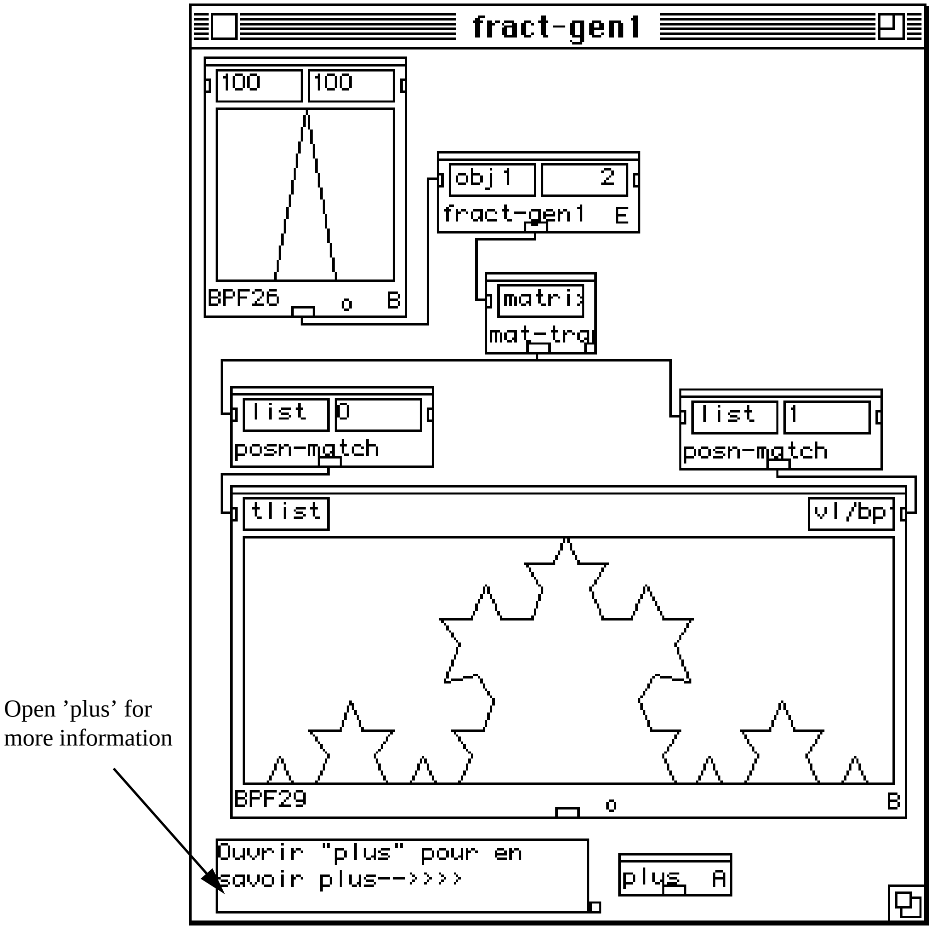
• *obj2* is the pairs of coordinates or a BPF.

The **fract-gen1** module applies the figure, or object, defined by *obj1* onto itself or onto a second object, *obj2*, if that optional input has been opened.

The output of this module is a list of coordinates in two dimensions :

((x_0 y_0) (x_1 x_2) ... (x_n y_n)).

Below is an example of the application of this algorithm on a curve contained within a BPF module :

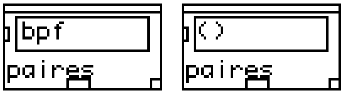


The following example shows the application of the input curve in the window *obj1* on the input curve in the window *obj2* :

5 Outils (Tools)

This section contains certain tools for manipulating geometry in a plane.

4.1 paires



Syntax

(alea::paires *bpf*)

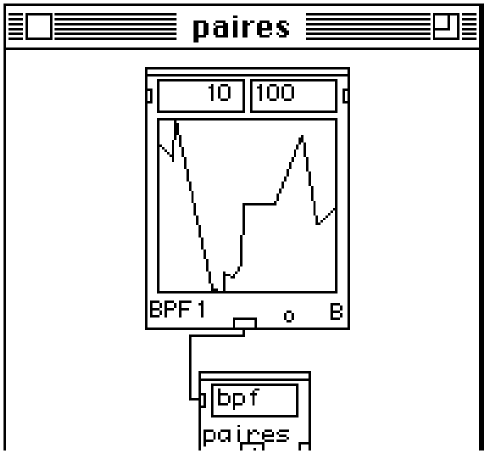
Inputs

bpf a **multi-BPF** module

Output

the coordinates of the points within a **multi-BPF**

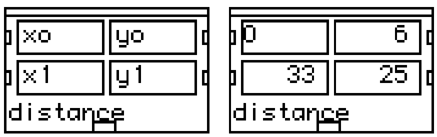
Outputs a list of the coordinates of the points within a **multi-BPF** :



Evaluating the patch below will produce the following result :

```
? PW->((10 81) (15 71) (16 93) (29 4) (34 2) (34 13) (37 10) (40 17) (41 48)
(52 49) (62 84) (67 38) (75 48))
```

4.2 distance



Syntax

(alea::distance *xo yo x1 y1*)

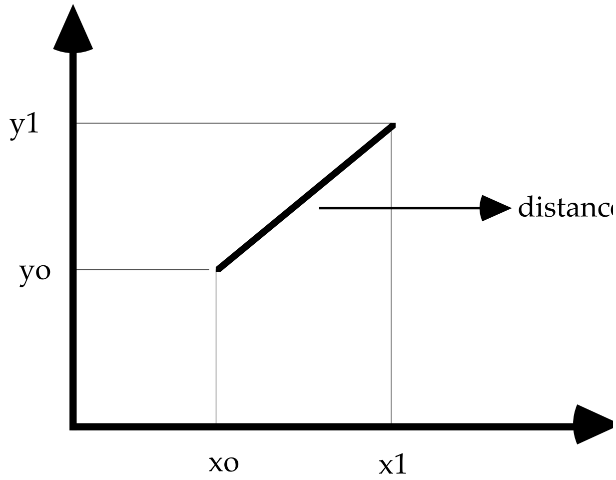
Inputs

xo	whole or floating-point number
yo	whole or floating-point number
x1	whole or floating-point number
y1	whole or floating-point number

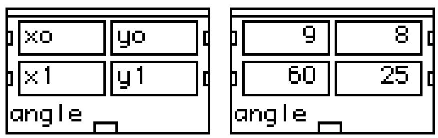
Output

the distance between *xo yo* and *x1 y1*

Calculates the Euclidean distance between two points in the same plane at coordinates *xo yo* and *x1 y1* .



4.3 angle



Syntax

(alea::angle *xo yo x1 y1*)

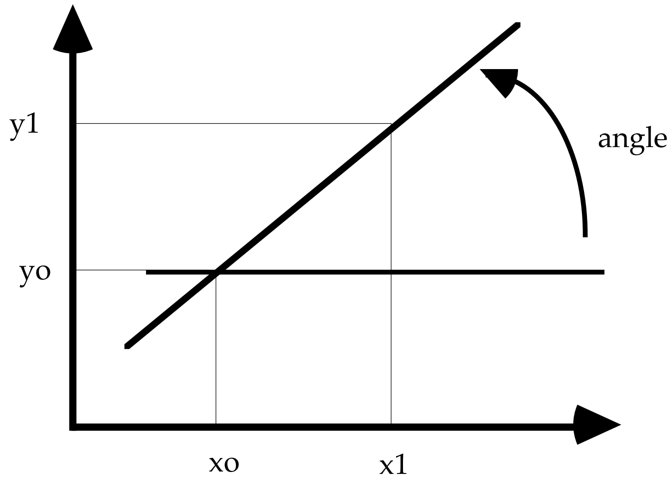
Inputs

xo	whole or floating-point number
yo	whole or floating-point number
x1	whole or floating-point number
y1	whole or floating-point number

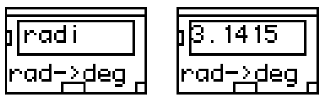
Output

angle in radians

Calculates the angle in radians in the plane formed from the line segment made by two points at coordinates (*xo yo*) and (*x1 y1*) and the x-axis.



4.4 rad-deg



Syntax

(alea::rad-deg *radi*)

Inputs

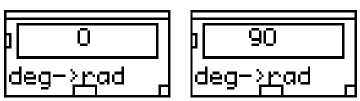
radi whole or floating-point number (angle in radians)

Output

angle in degrees

Converts radians into degrees.

4.5 deg-rad



Syntax

(alea::deg-rad *deg*)

Inputs

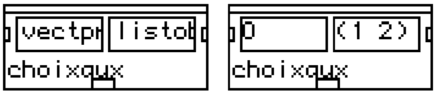
deg whole or floating-point number (angle in degrees)

Output

angle in radians

Converts degrees into radians

4.6 choixaux



Syntax

(alea:choixaux *vectprob listobjets*)

Inputs

vectprob list

listobjets list

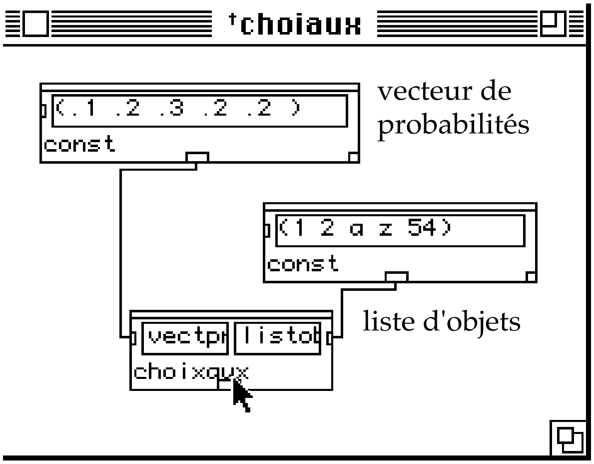
Output

an object (element of *listobjets*)

This module makes a choice between multiple alternatives (*listobjets*) based on a probability vector *vectprob*.

Example :

Evaluating the module **choiaux** will output one of the elements in the list connected to the input *listobjets* in function of the list of probabilities (probability vector) connected to the input *vectprob*.



choiaux
(.1 .2 .3 .2 .2)
const
vecteur de
probabilités
(1 2 a z 54)
const
vectpr
listob
choixaux
liste d'objets

Bibliography

Barnsley, Michael, *Fractals Everywhere,* Academic Press, Inc., 1988.

Barnsley, Michael F. et al, *The science of Fractal Images*, Springer-Verlag - New York, 1988.

Peitgen, H.O et al, *Chaos and Fractals* **-** *New Frontiers of Science,* Springer-Verlag, New York, 1993.

Bidlack**,** Rick**,** **"**Chaotic Systems as Simple (but complex) Compositional Algorithms", *Computer Music Journal*, vol 16, n° 3, Fall 1992.

Cvitanovic, Predrag (ed.), *Universality in Chaos*, Adam Hilger, 1989. Many articles, including one by Lorentz.

Dalmenico, A. D. , Chabert, J. L. et Chemla, K. (eds), *Chaos et Déterminisme*, Ed. du Seuil, Paris, 1992.

Feigenbaum, Mitchell J., "Universal Behavior in Nonlinear Systems", *Los Alamos Science* 1, 1980, pp. 4-27

Gogins, Michael, "Iterated Functions Systems Music", *Computer Music Journal*, vol 15, n° 1 1991.

Kojeve, Alexandre, *L'Idée du Determinisme dans la physique classique et dans la physique moderne*, Librairie Générale Française, Paris, 1990.

Helleman, Robert H. G., "Self-Generated Chaotic Behavior in Nonlinear Mechanics", *Fundamentals Problems in Statistical Mechanics,* vol 5, 1980, pp 165-233.

Rossler Otto E., "An equation for Continous Chaos", *Physics Letters* 57A, 1976, pp 397-398.

Ruelle, David, "Strange Attractors", *The Mathematical Intelligencer* 2, pp. 126-37, 1980.

Ruelle, David, *Hasard et Chaos***,** Ed. Odile Jacob, Paris, 1991.

Index

S

T

U

V