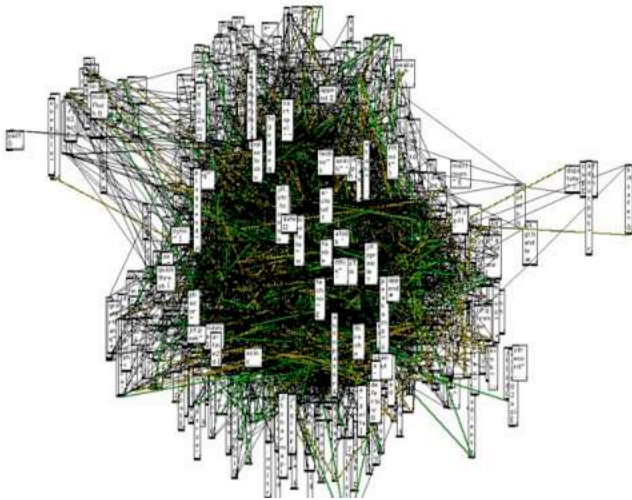


# a-objects<sub>0x10</sub>

by André Sier | [www.s373.net](http://www.s373.net)



atan\_sol092734 (André Sier, 2005)

the a-objects are an externals set for max msp  
jitter by cycling74. ranges from utility  
objects to some 3d synthetic motion physics  
sonic undulation

this file is in ongoing construction..  
suggestions appreciated at alpha @ s373.net

copyright: andré sier, 2001-2007  
documentation credits: alpha  
documentation version: 03c 03a 02 01 00  
published by s373.net[planalto de agenciamento maquinaico] oct 2007  
2007-10-27 07:00:36 AM

## table of contents

the a-objects overview patch

utils:

a-12many	a delay line for 1 float to a list of many floats
a-3602azi	convert angles system 360->azimuth
a-a	returns golden proportion
a-azi2360	convert angles system azimuth->360
a-change	output i,f,l only if change within fuzzy
a-count	a floating counter
a-dir	calculates azi,ele,dist from 2 successive 3d points
a-delta	difference from previous input: i, f, l
a-fiddle	(pos,amp) list building
a-flu	microbes on floats
a-hglide	list compression/expansion
a-hv	map a point to a rect drawing command
a-kin3d	3d conjugates
a-kinema	2d conjugates
a-len	length to
a-listscramble	shuffle the elements of a list
a-map	another linear mapping
a-mmm	min, max, mean of a stream of numbers
a-pad	entwine a list with values
a-polys	0-1 stream to multiple selections
a-rand	rand numbers mapped to intervals
a-smooth	one pole low pass smoother on i, f, l
a-waiter	serves numbers by several ways

3d utils:

a-2quad	convert a 3d point to a quad with rotation
a-3dkernel	compute a 4x3 matrix kernel & transforms points
a-cam	1st person 3d camera navigation
a-camera	quaternion based camera
a-terr	a terrain model in max
gl_grid	simple jitter 3d grid
gl_kinescope	imax cinema for gl
gl_ruttetra	ruth/etra synthesizer: 1-channel matrix to surface
gl_terrain	simple gl 3d terrain in jitter
gl_terrain2	simple gl 3d terrain in jitter
gl_terreno	um terreno 3d simples no jitter

geometry / movement:

a-celerador	thrust physical model
a-circulo	outputs a circle <xy> momentum
a-elips	outputs elliptical <xy> coords
a-fifo	fifo method for groups of y points in x dimensions
a-grav	orbit around a point
a-hemisphere	maps a xy list to a hemisphere
a-hsplines3d	hermite splines interpolation method in 3 dimensions
a-hypercube	hypercube xyz coords
a-line2d	bidimensional line momentum
a-line3d	tridimensional line momentum
a-lissa	lissajous xy pattern generator
a-mar	wave propagation
a-spring	a spring
a-spring+	a spring
a-swarm	a swarm of bees

msp synthesis:  
 a-gaussnoise~ msp gaussian noise generation  
 a-lorenz~ msp lorentz noise generation  
 a-pt~ msp poisson trigger noise generation  
 a-pt2~ msp poisson trigger interp noise generation  
 a-random~ msp rand noise generation  
 a-randomwalk~ msp random walk generation  
 a-rednoise~ msp red noise generation  
 a-rossler~ msp rossler noise generation  
 a-spring~ msp spring model noise generation  
 a-string~ msp string model noise generation

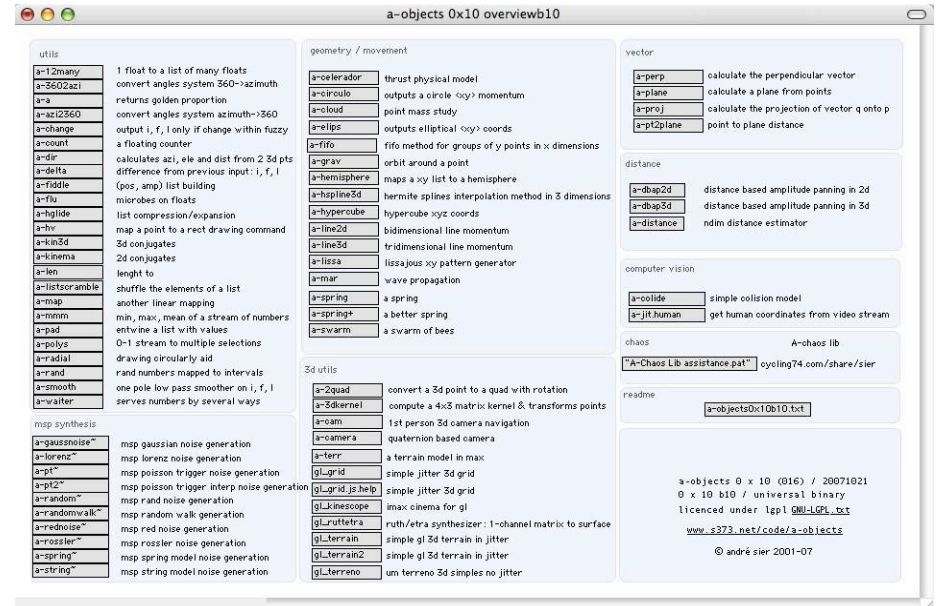
vector:  
 a-perp calculate the perpendicular vector  
 a-plane calculate a plane from points  
 a-proj calculate the projection of vector q onto p  
 a-pt2plane point to plane distance

distance:  
 a-dbap2d distance based amplitude panning in 2d  
 a-dbap3d distance based amplitude panning in 3d  
 a-distance ndim distance estimator

computer vision:  
 a-colide simple colision model  
 a-jit.human get human coordinates from video stream

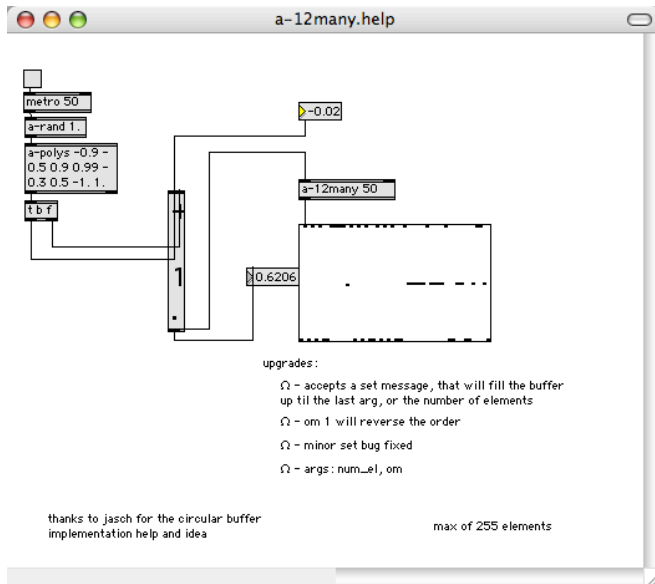
chaos:  
 see A-Chaos Lib (cycling74.com/share/sier)

## the overview patch



[a-objects->utils]

**a-12many** a delay line for 1 float to a list of many floats



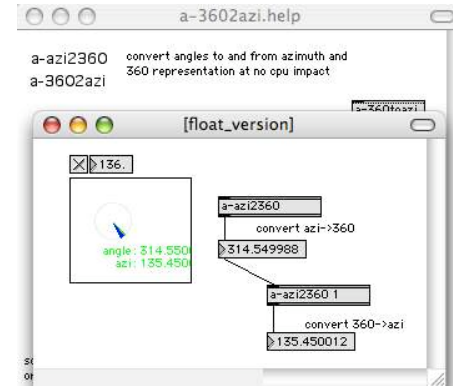
arguments:  
 1 int specifies number of elements in delay line

messages:  
 float next number for the delay line  
 set set next number for the delay line  
 om sets mode (0 to right, 1 to left)

outputs:  
 list N elements specified in argument

[a-objects->utils]

**a-3602azi** convert angles system 360->azimuth



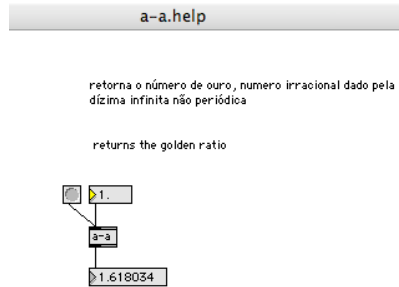
arguments:  
 na

messages:  
 float angle in 360. to be converted

outputs:  
 float angle in azi

[a-objects->utils]

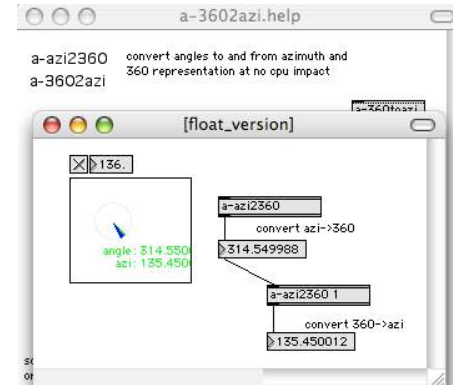
**a-a** returns golden proportion



arguments:  
na  
messages:  
float multiplier  
outputs:  
float result

[a-objects->utils]

**a-azi2360** convert angles system azimuth->360

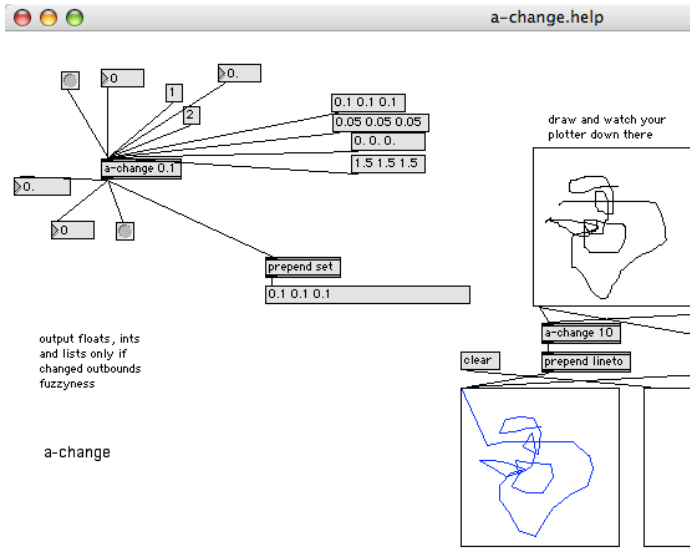


arguments:  
na  
messages:  
float angle in azi to be converted  
outputs:  
float angle in 360

[a-objects->utils]

### a-change

output i,f,l only if change within fuzzy



arguments:  
float/int

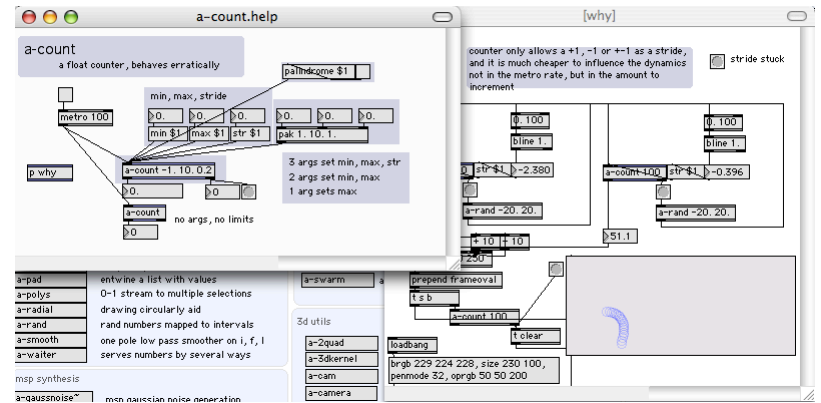
messages:  
fuzzy amount of fuzzyness to consider

outputs:  
i,f,l changed input

[a-objects->utils]

### a-count

a floating counter



arguments:  
1 float/int sets max  
2 float/int sets min,max  
3 float/int sets min,max, stride

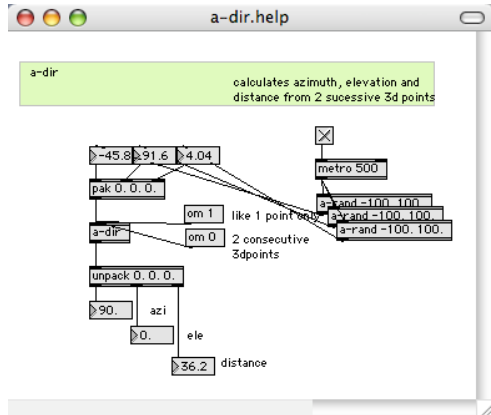
messages:  
min starting point for counter  
max ending point for counter  
str stride for counter  
palindrome flag to toggle palindrome mode

outputs:  
float counter value

[a-objects->utils]

### a-dir

calculates azi,ele,dist from 2 successive 3d points



arguments:  
na

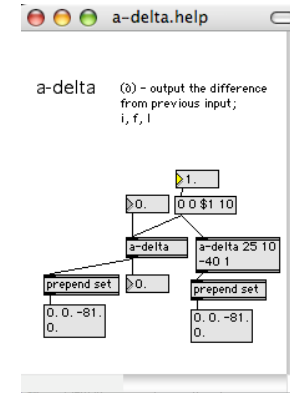
messages:  
list 3d point coords  
om mode

outputs:  
list azi, ele, dist

[a-objects->utils]

### a-delta

difference from previous input: i, f, l



arguments:  
na

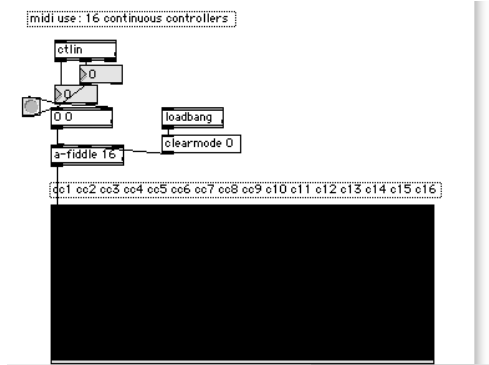
messages:  
list  
int, float

outputs:  
list, int, float

[a-objects->utils]

### a-fiddle

(pos,amp) list building



arguments:  
 int           number of list members

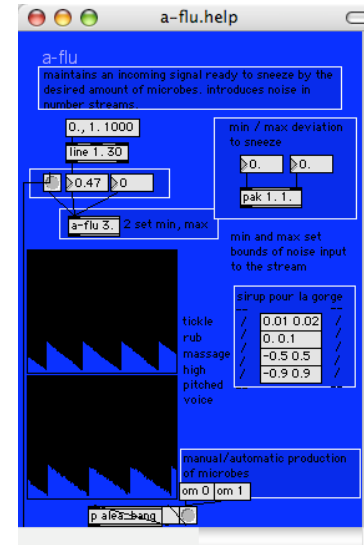
messages:  
 list           first element is pos on list, second is value

outputs:  
 list

[a-objects->utils]

### a-flu

microbes on floats



arguments:  
 1 float/int   sets max  
 2 float/int   sets min,max

messages:  
 list           min/max to deviate  
 int,float     reference value

outputs:  
 float         deviated value

[a-objects->utils]

## a-hglide

list compression/expansion

a-hglide

Linear List Expansion: Input Args are expanded or compressed linearly into a list of N arguments

how long is the list

how many points to interpolate from in int, float methods

numpoints 100

1 2 -2 3.8 -6.1415

metro 50

a-rand 100 1000

>0

t b i

random

-500

\* 0.01

a-hglide

p cascade

like int, float,deline (or a-12many ....)

gate

numpoints \$1

prepend set set

```

-2. -1.992 -1.984 -1.976 -1.968 -1.96 -1.952 -1.944
-1.936 -1.928 -1.92 -1.912 -1.904 -1.896 -1.888 -
1.88 -1.872 -1.864 -1.856 -1.848 -1.84 -1.832 -
1.824 -1.816 -1.808 3. 3.008 3.016 3.024 3.032 3.04
3.048 3.056 3.064 3.072 3.08 3.088 3.096 3.104
3.112 3.12 3.128 3.136 3.144 3.152 3.16 3.168
3.176 3.184 3.192 8. 7.977374 7.954747 7.932121

```

arguments:  
int numpoints in final list

messages:  
list reference list  
int,float reference value

outputs:  
list expanded/compressed list

[a-objects->utils]

## a-hv

map a point to a rect drawing command

a-hv

receives a 2d list and transforms it in a rect center at received point with a desired horizontal and vertical offset

um objecto utilitário que pega num par de coordenadas <x> e adiciona-lhes o width e a height pretendidas.

automaticamente arranja as coordenadas de destino como sendo <x> do CENTRO e não do ponto superior esquerdo, ie, os pontos que saem correspondentes a um conjunto XY são (x-W/2, y-H/2, x+W/2, y+H/2). se quiserem o comportamento normal, ie, (x, y, x+w, y+h) basta "fix 0"

André Sier 20040124

arguments:  
int,float default width and/or height for the point

messages:  
hv width and/or height for the point  
list point to be rect'ized

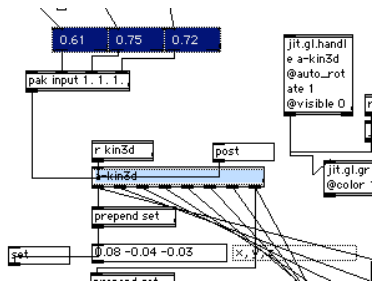
outputs:  
list rect drawing list



[a-objects->utils]

### a-kin3d

3d conjugates



arguments:  
 int,float      default width and/or height for the point

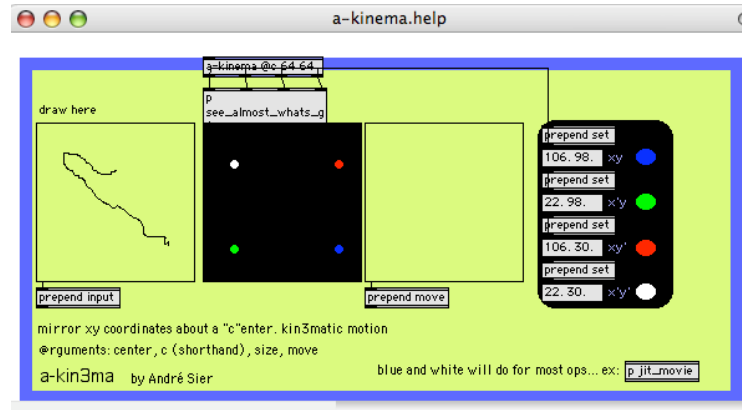
messages:  
 hv              width and/or height for the point  
 list             point to be rect'ized

outputs:  
 list             rect drawing list

[a-objects->utils]

### a-kinema

2d conjugates



arguments:  
 @center          defines center to make conjugates from

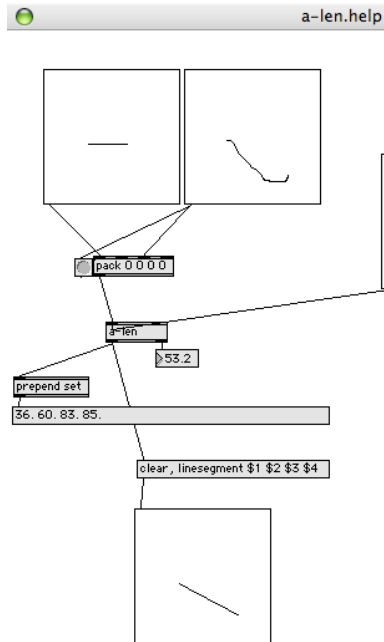
messages:  
 input            point to be cojugated

outputs:  
 4 lists          conjugate points

[a-objects->utils]

### a-len

length to



arguments:  
na

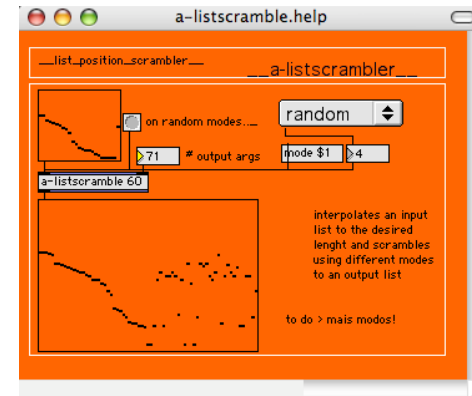
messages:  
list                    2 points

outputs:  
list                    difference of 2 points  
float                   length to

[a-objects->utils]

### a-listscramble

shuffle the elements of a list



arguments:  
int                    default list length

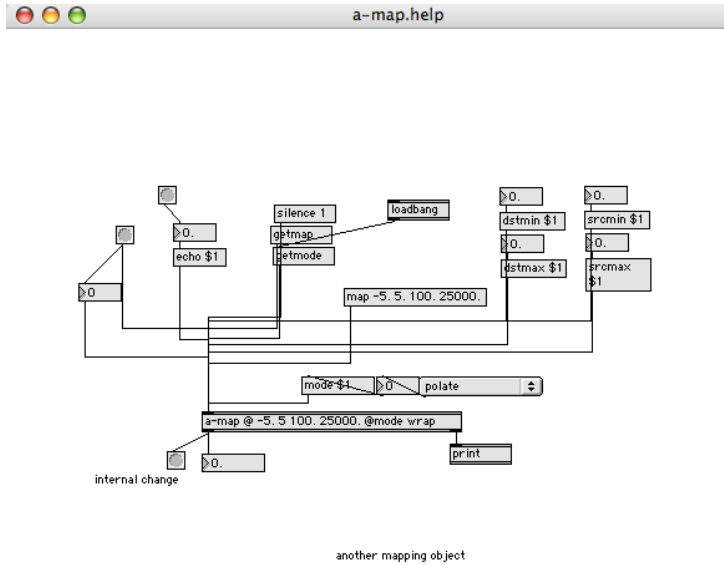
messages:  
mode                   change the mode  
int                    # of args

outputs:  
list                   shuffled list

[a-objects->utils]

### a-map

another linear mapper



arguments:  
 @  
 @mode mapping list: src min, src max, dst min, dst max  
 bound mode: interpolate, clip, wrap, fold

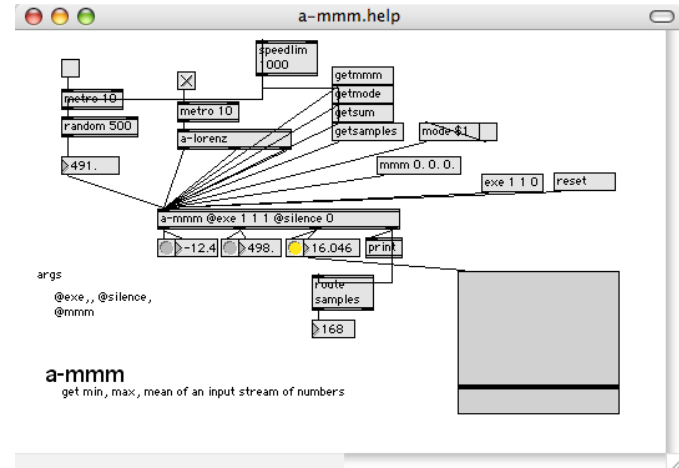
messages:  
 map mapping list: src min, src max, dst min, dst max  
 float input to be scaled

outputs:  
 float scaled value

[a-objects->utils]

### a-mmm

min, max, mean of a stream of numbers



arguments:  
 @exe config ops to be performed  
 @silence new output on parameter change

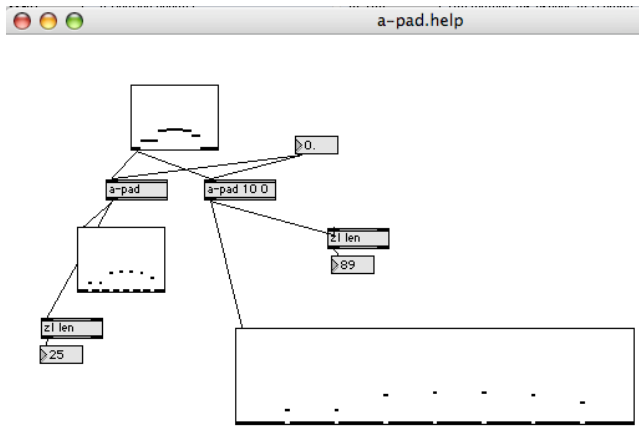
messages:  
 exe mapping list: src min, src max, dst min, dst max  
 silence input to be scaled  
 float input to be analyzed

outputs:  
 float min value  
 float max value  
 float mean value

[a-objects->utils]

### a-pad

entwine a list with values



arguments:  
 list num elements to be padded, and value to pad with

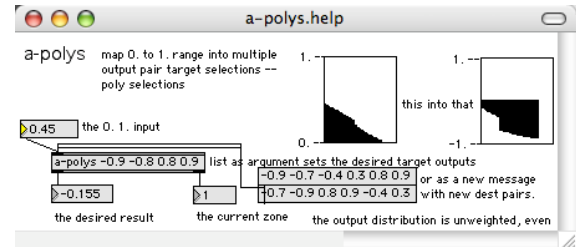
messages:  
 list list to be padded

outputs:  
 list padded list

[a-objects->utils]

### a-polys

0-1 stream to multiple selections



arguments:  
 list desired pairs of output values

messages:  
 float input

outputs:  
 float value between multiple selections

[a-objects->utils]

## a-rand

rand numbers mapped to intervals

The screenshot shows a window titled "a-rand.help" with the following text:

**a-rand**  
updated random algorithm

This random number generator originally appeared in "Toward a Universal Random Number Generator" by George Marsaglia and Arif Zaman. Florida State University Report: FSU-SCRI-87-50 (1987) It was later modified by F. James and published in "A Review of Pseudo-random Number Generators"

**args:**  
1 arg sets range(0, arg)  
2 arg sets range(arg1, arg2)

**messages:**  
range or list (1 or 2 args): sets dstmin and dstmax  
bang: next random number  
seed (2 ints) : re-init random engine. see p more.

The patch diagram shows a 'get random number' object connected to a 'seed 25503 11324' object, which then feeds into an 'a-rand 31768' object. The output of 'a-rand' is connected to a 'range 3.' object, which outputs '-5.7.'. A '3697.' object is also shown at the bottom left.

### arguments:

1 float/int	sets max
2 float/int	sets min,max

### messages:

bang	next pseudo-number
seed	seed random engine
range	set new min max random range
list	set new min max random range

### outputs:

float	pseudo-number
-------	---------------

[a-objects->utils]

## a-smooth

one pole low pass smoother on i, f, l

The screenshot shows a window titled "a-smooth.help" with the following text:

**a-smooth**  
fast one pole smoothing lowpass filter for ints, floats and lists

The patch diagram shows a 'metro 50' object connected to an 'a-rand 50.' object, which feeds into a '>0.' object. The output of '>0.' is connected to a 'smooth \$1' object, which also receives input from a 'p lists\_too' object. The output of 'smooth \$1' is connected to a 'metro 5' object, which feeds into four 'a-smooth' objects with different arguments: 'a-smooth 0.9', 'a-smooth 0.7', 'a-smooth 0.4', and 'a-smooth 0.5 0.99'. The outputs of these objects are connected to a 'key keyup' object, which feeds into 'sel 32' objects, which then feed into a 'metro 5' object.

**new:** smooth now takes 2 arguments, like the standard max slide object. first arg is smooth up and second is down. if only 1 arg, both smooths share the same value

### arguments:

1 float/int	sets smooth up and down
2 float/int	sets smooth up,sets smooth down

### messages:

bang	smooth
float	sets new input and smooth
list	sets new input list and smooth

### outputs:

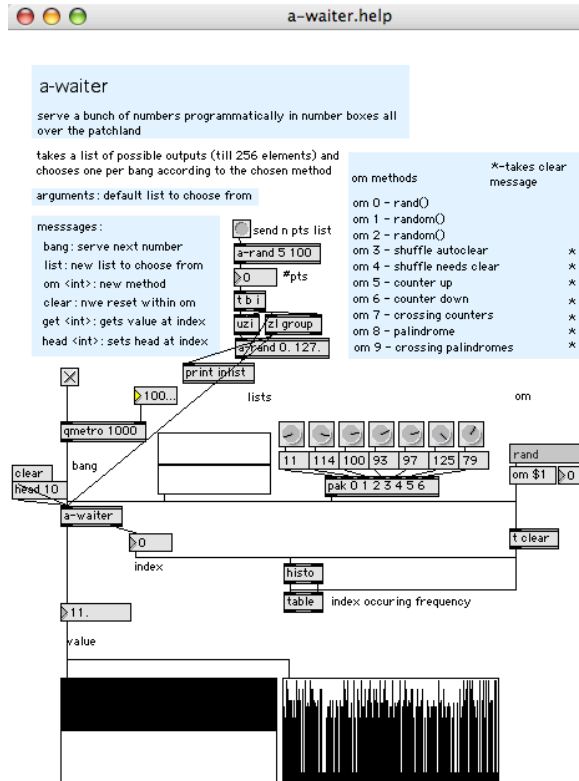
float	smoothed value
list	smoothed list

[a-objects->utils]

[a-objects->3d utils]

### a-waiter

serves numbers by several ways



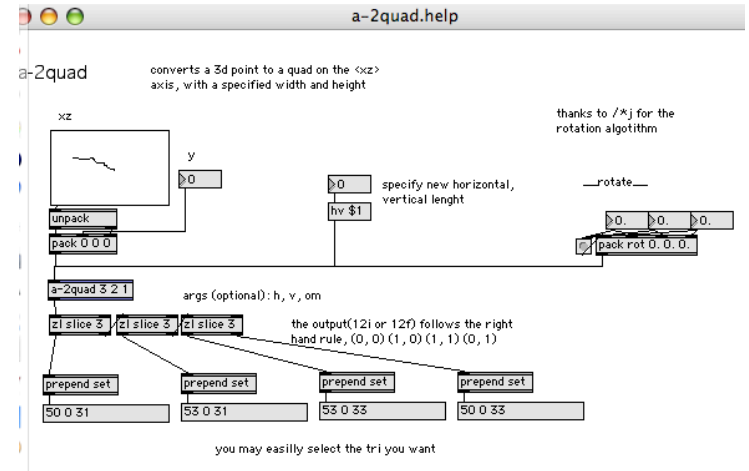
arguments:  
 list default list to choose from

messages:  
 see patch image

outputs:  
 float chosen value

### a-2quad

convert a 3d point to a quad with rotation



arguments:  
 h, v, om

messages:  
 hv new horizontal vertical width height  
 list 3d point coords

outputs:  
 list 4 3d points

[a-objects->3d utils]

### a-3dkernel

compute a 4x3 matrix kernel & transforms points

The screenshot shows the 'a-3dkernel.help' window. On the left, there's a patch of objects including 'in point xy', 'pack.translate', 'pack.scale', 'pack.rotate', 'gettransform.kernels', 'getcombined', 'getworld', 'getrotate', 'gettranslate', 'getscale', 'a-3dkernel', 'n.format', 'the 4x3 matrix kernel', 'inverted kernel', 'a-3dk', 'getrotate.obj', 'gettranslate.obj', 'getscale.obj', 'getobject', 'p.more-kernel', and 'p.notes'. The 'the 4x3 matrix kernel' is displayed as a 4x3 grid of numbers. On the right, a 3D visualization shows a red sphere and a green sphere connected by a red line, with a green vertical line extending upwards from the green sphere.

compute a 4x3 matrix kernel with some 3d affine transformations(translation, rotation, scaling);  
output kernels and transformed points;

added object methods and combined transforms

use a-3dkernel also to compute the original point from jit.la inverse's inverse matrix kernel transform

the 4x3 matrix kernel

-0.8660	-0.1294	-0.4830	-0.4403	-0.8660	-0.0000	-0.5000	-0.4800
0.0000	-0.9659	0.2588	-0.1835	-0.1294	-0.9659	0.2241	-0.1900
-0.5000	0.2241	0.8965	-0.1974	-0.4830	0.2588	0.8965	-0.0000
0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000

transformed point  
-0.533913 -0.1956 -0.247736

original point reversed  
-0.533913 -0.1956 -0.247736

arguments:  
na

messages:

translate	translate and compute kernel
rotate	rotate and compute kernel
scale	scale and compute kernel
translate.obj	translate object kernel and compute kernels
rotate.obj	rotate object kernel and compute kernels
scale.obj	scale object kernel and compute kernels
list	point to be transformed
getcombined	getcombined kernel
getworld	getworld kernel
getrotate	getrotate kernel
gettranslate	gettranslate kernel
getscale	getscale kernel

outputs:

list	transformed point
list	kernel

[a-objects->3d utils]

### a-cam

1st person 3d camera navigation

The screenshot shows the 'a-cam.help' window. It contains a complex patch of objects for camera navigation, including 'pack.setpos', 'pack.setlook', 'pack.setup', 'pack.setright', 'fwd', 'strafe', 'up', 'pack.move', 'pack.circle.up', 'pack.circle.right', 'a-cam', 'unpack', 'lookat position', 'camera position', and 'up vector'. The patch is organized into several sections, with some objects having arguments like '\$1' or '\$!'. A note indicates 'circle modes still disabled'.

arguments, [!], @pos @look @up @right

arguments:  
na

messages:

fwd	amount to go forward in set direction
strafe	amount to go sideways in set direction
up	amount to go up in set direction
rotate.x	rotate around x
rotate.y	rotate around y
rotate.z	rotate around z
bang	advance

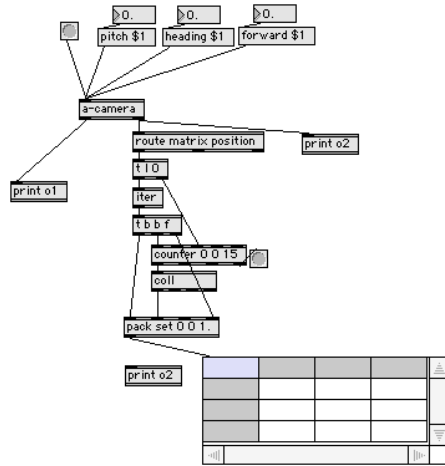
outputs:

list	lookat position
list	camera position
list	up vector

[a-objects->3d utils]

### a-camera

quaternion based camera



arguments:  
na

messages:

fwd	amount to go forward in set direction
strafe	amount to go sideways in set direction
up	amount to go up in set direction
rotate.x	rotate around x
rotate.y	rotate around y
rotate.z	rotate around z
bang	advance

outputs:

list	lookat position
list	camera position
list	up vector

[a-objects->3d utils]

### a-terr

terrain model in max

arguments:  
na

messages:

fwd	amount to go forward in set direction
strafe	amount to go sideways in set direction
up	amount to go up in set direction
rotate.x	rotate around x
rotate.y	rotate around y
rotate.z	rotate around z
bang	advance

outputs:

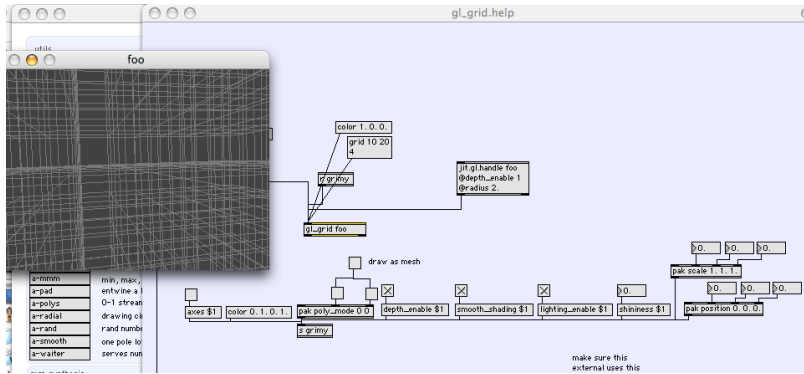
list	lookat position
list	camera position
list	up vector



[a-objects->3d utils]

### gl\_grid

simple jitter 3d grid



arguments:  
na

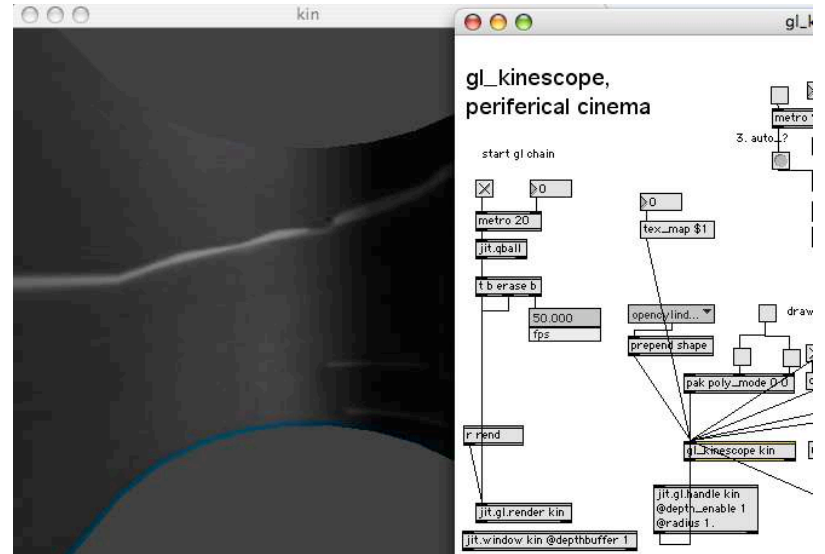
messages:  
grid x,y and z subdivisions of rendered grid

outputs:

[a-objects->3d utils]

### gl\_kinescope

imax cinema for gl



arguments:

messages:

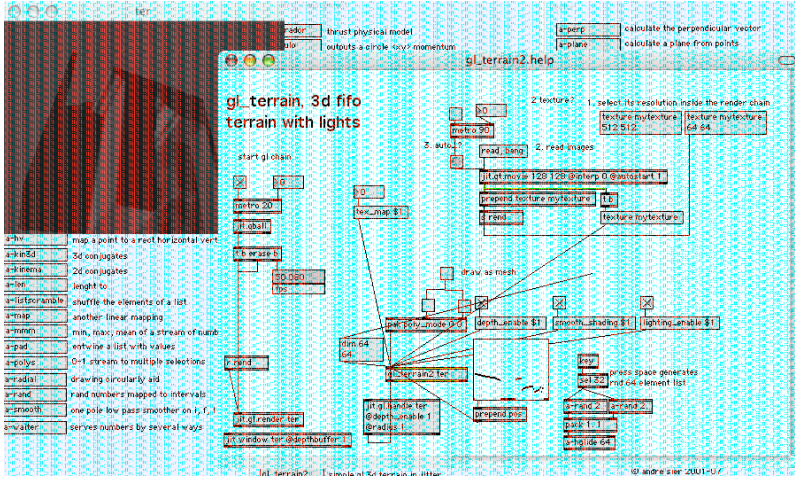
outputs:



[a-objects->3d utils]

### gl\_terrain2

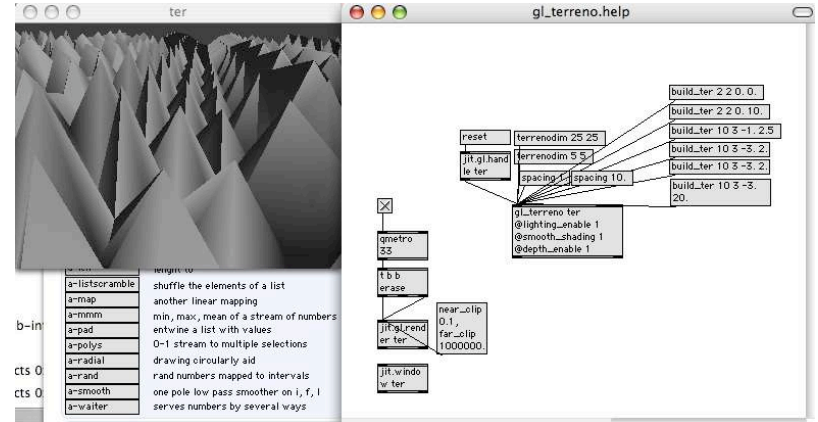
simple gl 3d terrain in jitter



[a-objects->3d utils]

### gl\_terreno

um terreno 3d simples no jitter



arguments:

messages:

outputs:

arguments:

messages:

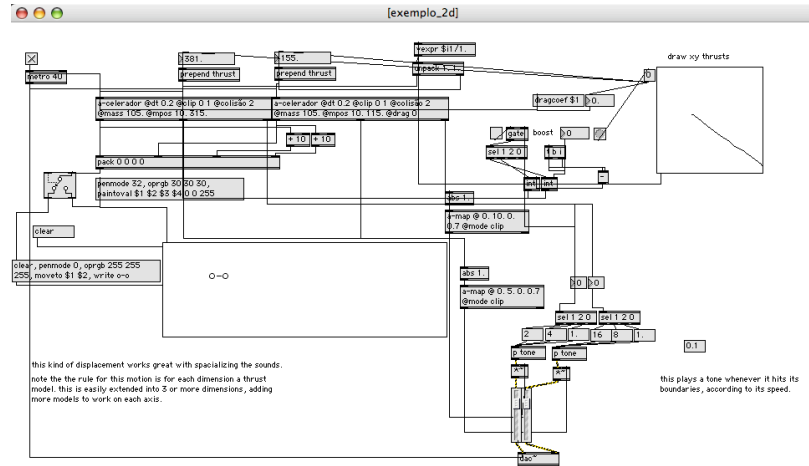
outputs:

[a-objects->geometry / movement]

[a-objects->geometry / movement]

## a-celerador

thrust physical model



### arguments:

same as messages prepended with @

### messages:

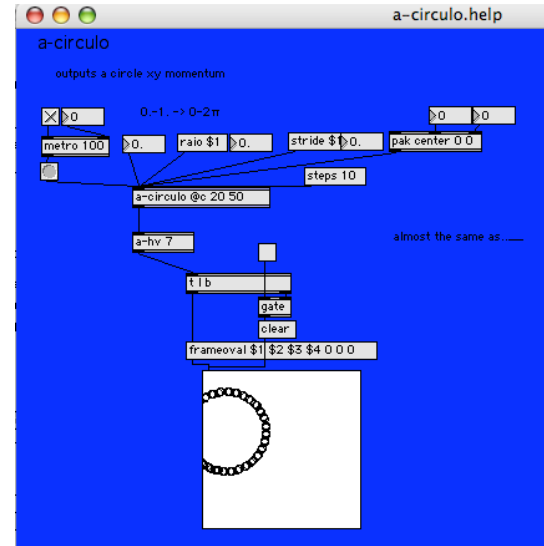
thrust	set acceleration
dt	set dt
mass	set mass
mvel	set max and min velocity
mpos	set max and min position
colisãõ	flag to be notified at bounds
colide	colide now
bang	advance

### outputs:

float	position
float	velocity
int	boundary state

## a-circulo

outputs a circle <xy> momentum



### arguments:

same as messages prepended with @

### messages:

center	set circle center
raio	set circle radius
stride	set circle stride
steps	set circle stride
float	sync to circle position
bang	advance circle by stride

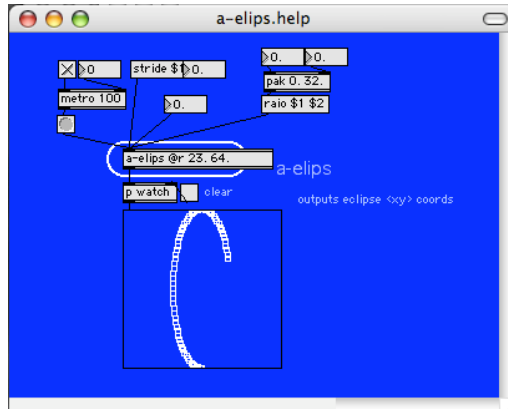
### outputs:

list	circular point position
------	-------------------------

[a-objects->geometry / movement]

### a-elips

outputs elliptical <xy> coords



arguments:  
 same as messages prepended with @

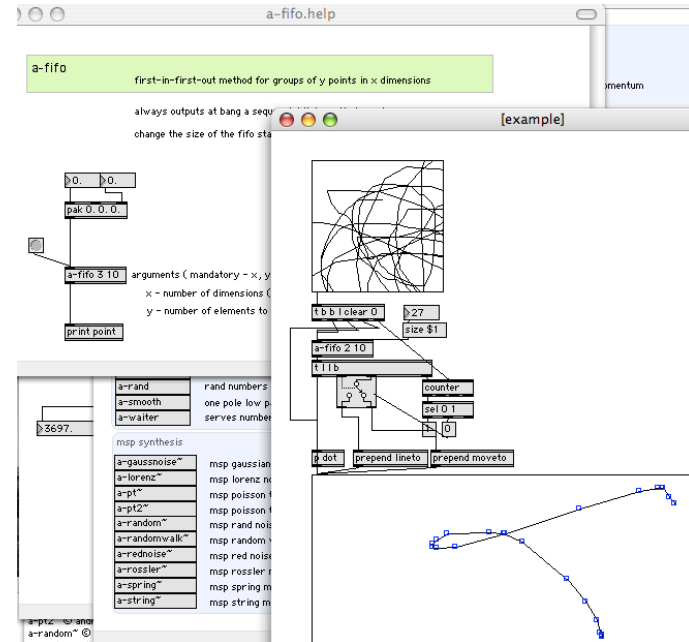
messages:  
 center            set ellipse center  
 raio             set ellipse radius  
 stride           set ellipse stride  
 steps            set ellipse stride  
 float            sync to ellipse position  
 bang             advance ellipse by stride

outputs:  
 list             elliptical point position

[a-objects->geometry / movement]

### a-fifo

fifo method for groups of y points in x dimensions



arguments:  
 2 ints            number of dimensions, number of elements in each dim

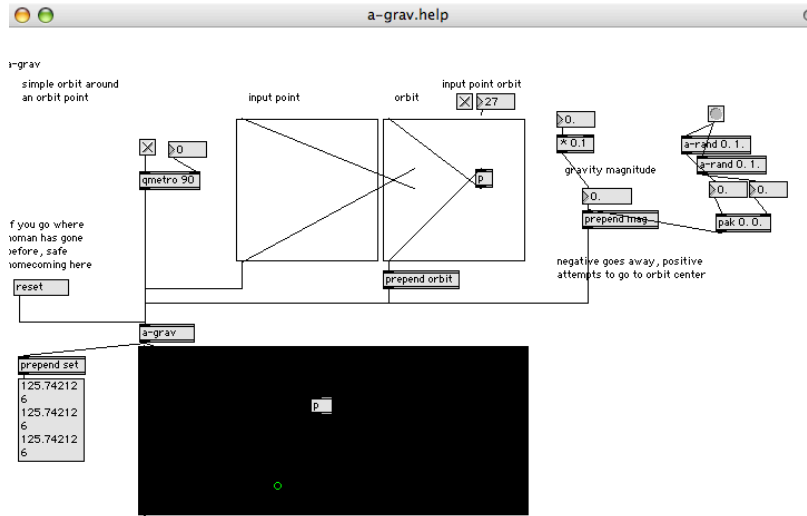
messages:  
 list             set next fifo element list  
 bang            get fifo chain

outputs:  
 list             fifo chain

[a-objects->geometry / movement]

### a-grav

orbit around a point



arguments:  
 2 ints            number of dimensions, number of elements in each dim

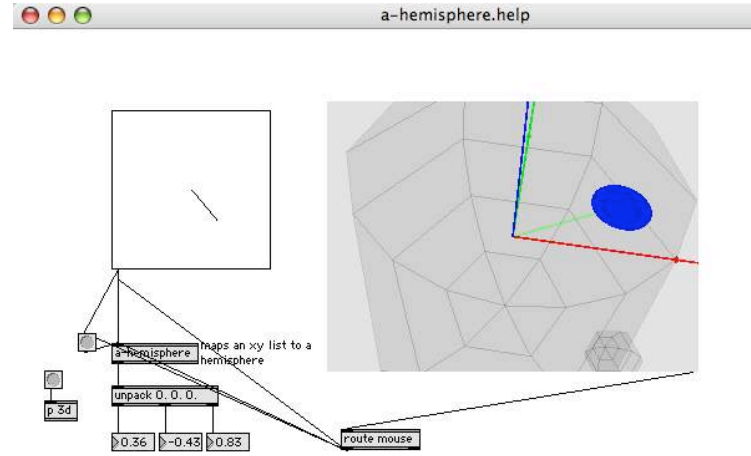
messages:  
 mag              gravitation forces magnitude  
 bang             calc point position

outputs:  
 list              point position

[a-objects->geometry / movement]

### a-hemisphere

maps a xy list to a hemisphere



arguments:

messages:  
 list              xy point  
 bang             get xyz pos

outputs:  
 list              xyz pos

[a-objects->geometry / movement]

### a-hsplines3d hermite splines interpolation method in 3 dimensions

Code snippets from the help window:

```

// hermite spline's method,
// 0 < t < 1;
// p(t) = (2t^3 - 3t^2 + 1)p0 + (t^3 - 2t^2 + 1)m0 + (t^3 - t^2)m1 + (-2t^3 + 3t^2)p1;
// m1 = ((1-t)/2)(p1 - p0) + (p1 - p0);
  
```

1. make control points (sequential list of xyz positions)

2. Iterate over curved surface

alpha controls the tension of the tangent to the spline; defaults at 0.05 is near linear; high values curl; stride \$t \$b 0; how much to advance each frame, or use steps

updates:  
running stack method  
double alpha  
p stack\_message

Sequential list of 3d control points (xyz pos)

arguments:  
list sequential list of 3d control points  
stack infinite control point seeding  
alpha tension of tangent to spline  
stride stride of spline calculations  
bang get xyz splined point pos

outputs:  
list xyz pos

[a-objects->geometry / movement]

### a-hypercube hypercube xyz coords

Code snippets from the help window:

```

// hermite spline's method,
// 0 < t < 1;
// p(t) = (2t^3 - 3t^2 + 1)p0 + (t^3 - 2t^2 + 1)m0 + (t^3 - t^2)m1 + (-2t^3 + 3t^2)p1;
// m1 = ((1-t)/2)(p1 - p0) + (p1 - p0);
  
```

1. begin \_gl\_

2. rotate \_gl\_ structure

outputs hypercube 3d coordinates into max

angle of rotation should yield a perfect ordinary cube

after a screen saver by Richard Asbury, thanks!

angle of rotation should yield a perfect ordinary cube

arguments:  
rotation angle  
get hypercube data

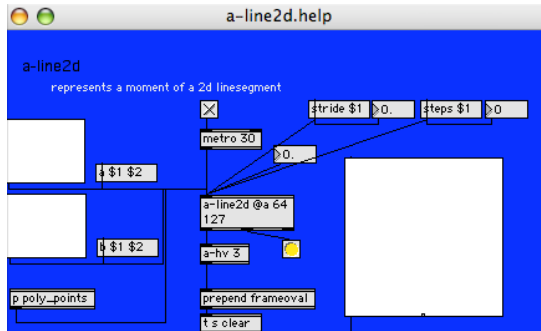
messages:  
angle rotation angle  
bang get hypercube data

outputs:  
list hypercube data

[a-objects->geometry / movement]

### a-line2d

bidimensional line momentum



arguments:  
 same as messages prepended with @

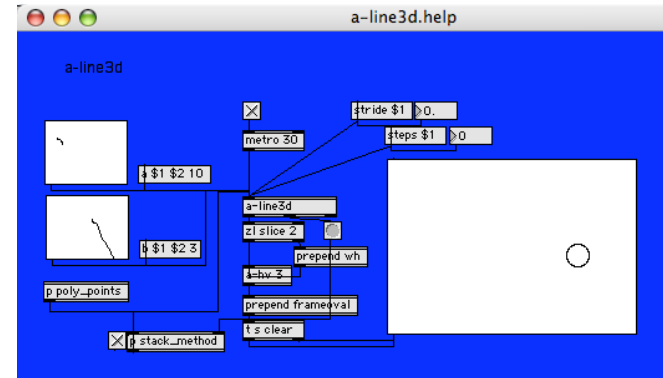
messages:  
 a set point a center  
 b set point b center  
 list sequential list of control points for line  
 stride set line stride  
 steps set line stride  
 float sync to line position  
 bang advance line by stride

outputs:  
 list line point position

[a-objects->geometry / movement]

### a-line3d

tridimensional line momentum



arguments:  
 same as messages prepended with @

messages:  
 a set point a center  
 b set point b center  
 list sequential list of control points for line  
 stride set line stride  
 steps set line stride  
 float sync to line position  
 bang advance line by stride

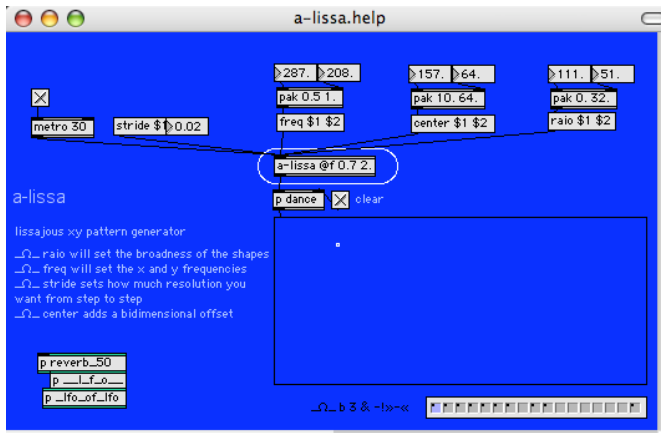
outputs:  
 list line point position



[a-objects->geometry / movement]

### a-lissa

lissajous xy pattern generator



arguments:  
 same as messages prepended with @

messages:

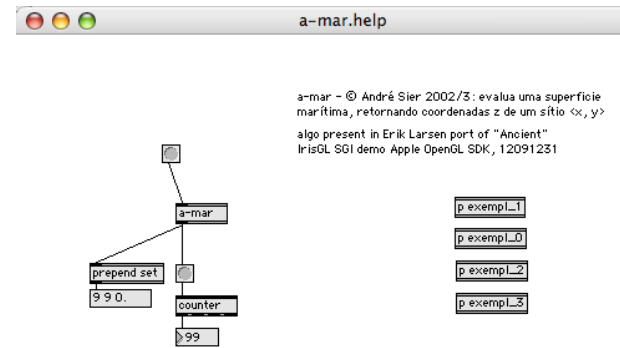
freq	set frequency
center	set center
raio	set radius
stride	set lissa stride
steps	set lissa stride
bang	advance line by stride

outputs:  
 list            lissajous point position

[a-objects->geometry / movement]

### a-mar

wave propagation



arguments:

messages:

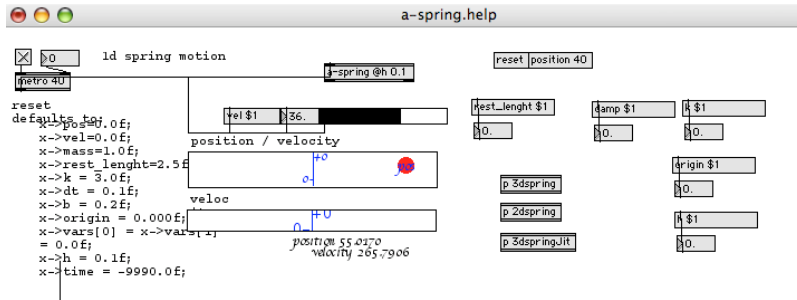
om	set mode
xy	set xy grid pos
grid	define grid size
dt	set dt of simulation
bang	advance simulation

outputs:  
 list            grid of points

[a-objects->geometry / movement]

## a-spring

a spring



arguments:  
same as messages prepended with @

messages:

k	set spring stiffness
damp	set spring damping
rest_lenght	set spring rest lenght
origin	set spring origin
vel	set spring velocity
pos	set spring position
dt	set simulation time factor
bang	advance spring

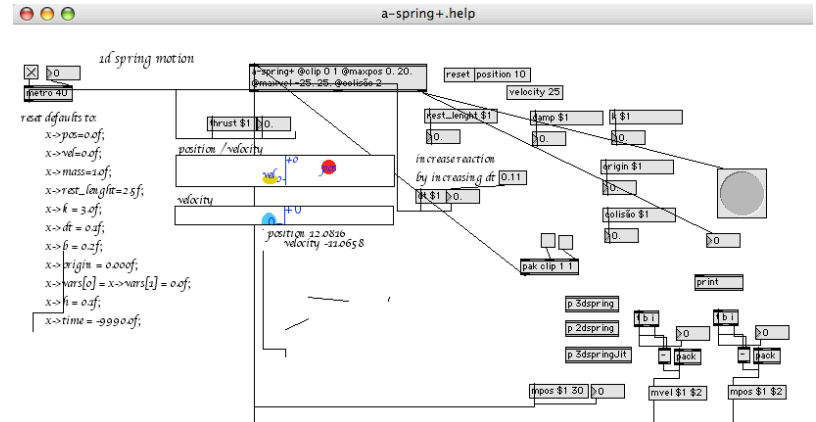
outputs:

float	point position
float	point velocity

[a-objects->geometry / movement]

## a-spring+

a spring



arguments:  
same as messages prepended with @

messages:

k	set spring stiffness
damp	set spring damping
rest_lenght	set spring rest lenght
origin	set spring origin
vel	set spring velocity
pos	set spring position
dt	set simulation time factor
bang	advance spring

outputs:

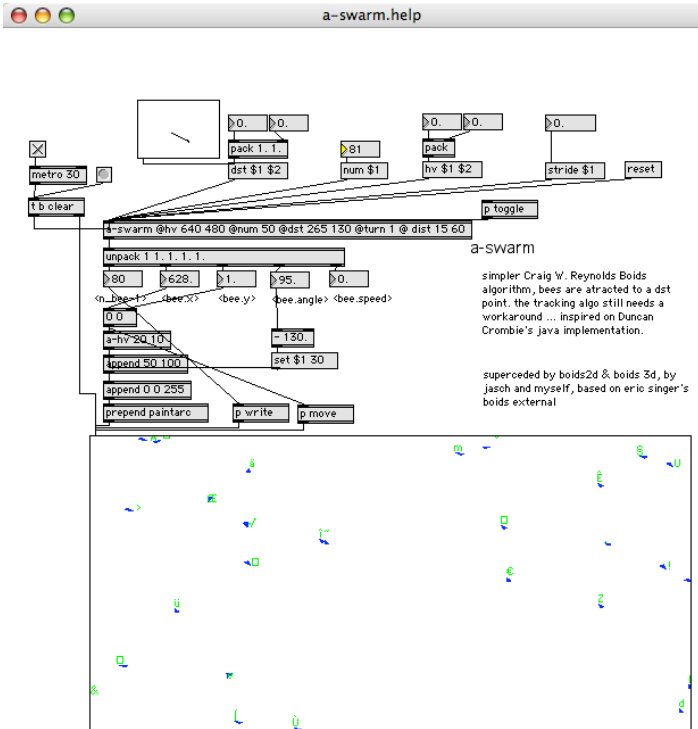
float	point position
float	point velocity

[a-objects->geometry / movement]

[a-objects->msp synthesis]

**a-SWARM**

a swarm of bees



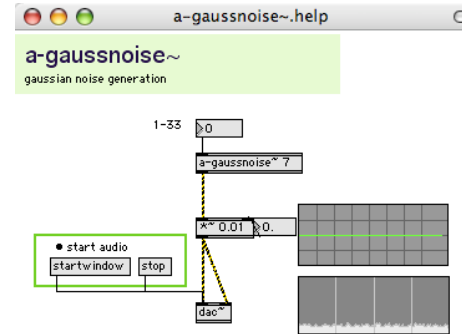
arguments:  
same as messages prepended with @

messages:  
num            number of bees  
hv             space  
stride         set stride simulation  
dst            set attract point  
bang          advance simulation

outputs:  
list           indexed bees point positions

**a-gaussnoise~**

msp gaussian noise generation



arguments:  
int            gaussnoise factor

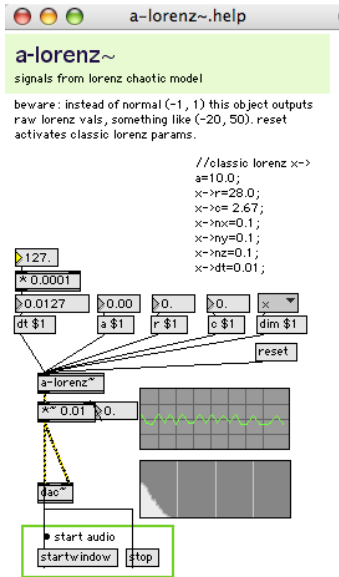
messages:  
int            gaussnoise factor

outputs:  
signal         gaussnoise signal

[a-objects->msp synthesis]

### a-lorenz~

msp lorenz noise generation



arguments:

messages:

- a set a constant of lorenz equation
- c set c constant of lorenz equation
- r set r constant of lorenz equation
- dim select output signal
- reset reset to classic lorenz params
- dt set simulation time factor

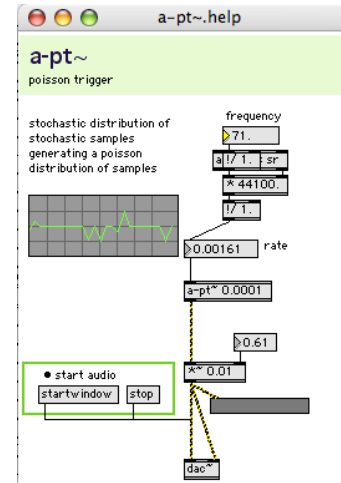
outputs:  
signal

lorenz noise signal

[a-objects->msp synthesis]

### a-pt~

msp poisson trigger noise generation



arguments:

float

set rate of poisson trigger

messages:

float

set rate of poisson trigger

outputs:

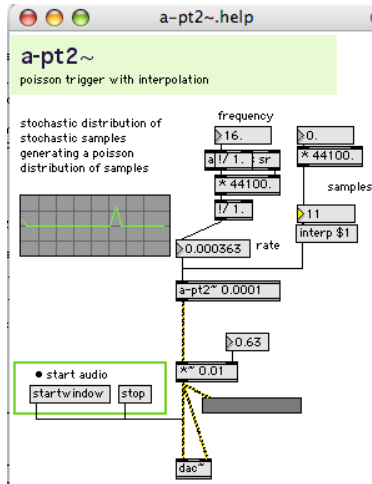
signal

poisson noise signal

[a-objects->msp synthesis]

### a-pt2~

msp poisson trigger interp noise generation



arguments:  
float set rate of poisson trigger

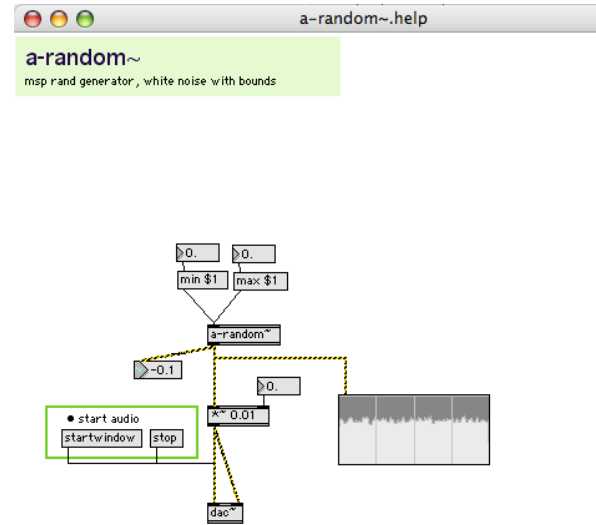
messages:  
float set rate of poisson trigger  
interp set num samples to interpolate

outputs:  
signal interped poisson noise signal

[a-objects->msp synthesis]

### a-random~

msp random noise generation



arguments:

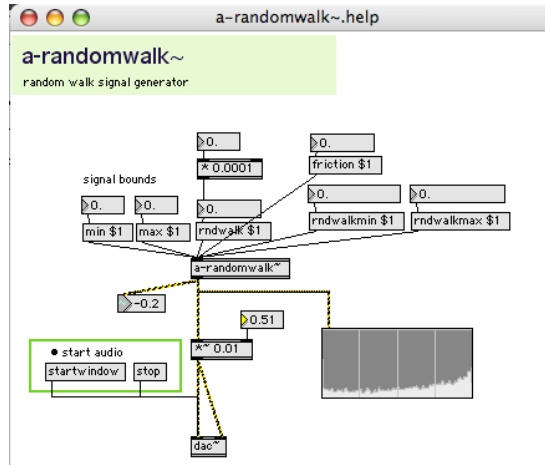
messages:  
min set min bounds of signal  
max set max bounds of signal

outputs:  
signal white noise

[a-objects->msp synthesis]

### a-randomwalk~

msp randomwalk noise generation



arguments:

messages:

min set min bounds of signal  
 max set max bounds of signal  
 rndwalk set rndwalk min and max  
 rndwalkmin set rndwalk min  
 rndwalkmax set rndwalk max

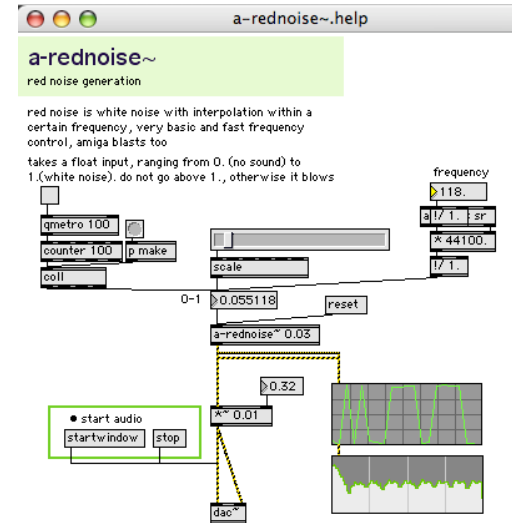
outputs:

signal rndwalk noise

[a-objects->msp synthesis]

### a-rednoise~

msp red noise generation



arguments:

float set rate of red noise gen

messages:

float set rate of red noise gen

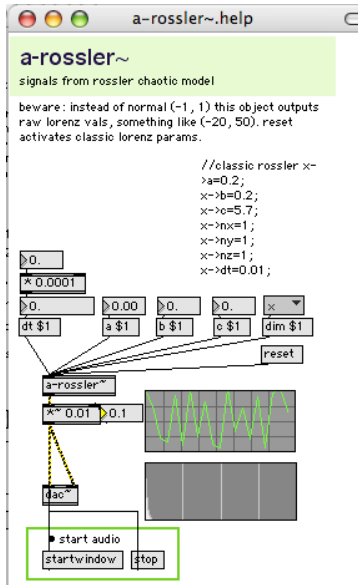
outputs:

signal red noise signal

[a-objects->msp synthesis]

### a-rossler~

msp rossler noise generation



arguments:

messages:

- a set a constant of rossler equation
- b set b constant of rossler equation
- c set c constant of rossler equation
- dim select output signal
- reset reset to classic rossler params
- dt set simulation time factor

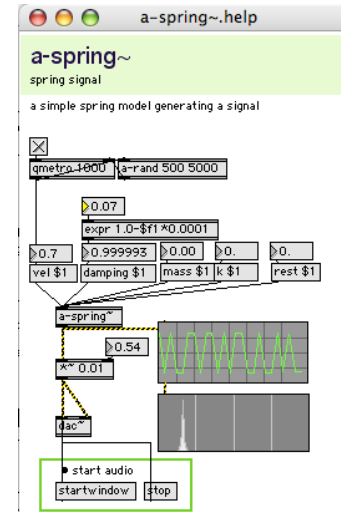
outputs:  
signal

rossler noise signal

[a-objects->msp synthesis]

### a-spring~

msp spring model noise generation



arguments:

messages:

- vel give the string a velocity
- damping damp the string
- mass the mass
- k spring stiffness
- rest rest position
- reset reset spring

outputs:  
signal

spring signal

[a-objects->msp synthesis]

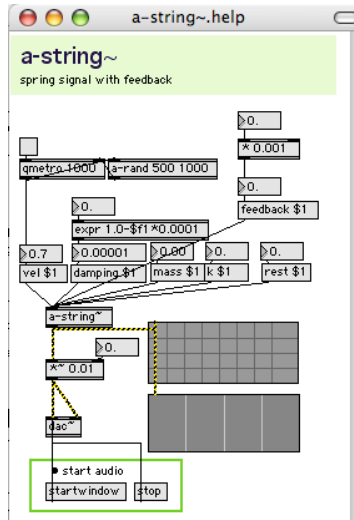
[a-objects->vector]

### a-string~

msp string model noise generation

### a-perp

calculate the perpendicular vector



arguments:

messages:

vel	give the string a velocity
damping	damp the string
mass	the mass
k	spring stiffness
rest	rest position
reset	reset spring

outputs:

signal	string signal
--------	---------------

arguments:

messages:

outputs:



[a-objects->vector]

**a-plane**

calculate the perpendicular vector

arguments:

messages:

outputs:

[a-objects->vector]

**a-proj**

calculate the perpendicular vector

arguments:

messages:

outputs:

[a-objects->vector]

## a-pt2plane

calculate the perpendicular vector

arguments:

messages:

outputs:

[a-objects->distance]

## a-dbap2d

distance based amplitude panning in 2d

welcome to this mini tutorial of a-dbap2d.

this tutorial will proceed in 2d, just be aware that to do this in 3d you have to send 3d vectors for sound coords and speaker placement. ok, ready to go.

first, pick a coordinate system, no matter which, just be consistent. i'll use floats with top left corner as (0, 0) and bottom right corner as (1, 1), and i'm looking at spatializing 3 sounds on 4 speakers arranged in a quad setup.

i'll start with a square quad test setup, with me at (0.5, 0.5), the front left speaker is (0, 0), the front right speaker is (1, 0), and the back left(0, 1), back right (1, 1). this gives me the info to config the speakers.

next, sequenciate the coordinates in the right order that you want, this will be important for the dac channel output... can be any order, just keep using it. i add a little offset to the extremes for the viz and coords engine.

speakers 0.1 0.1 0.9 0.1 0.1 0.9 0.9 0.9

this message configures the speaker space, ie, where they are located in our defined coordinate space. it is sent to a-dbap2d and also sent to the visualizer

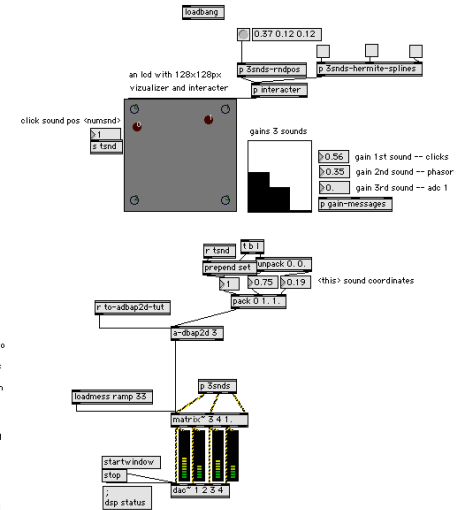
next, define how many max: input sounds you will spatialize, the message <num> sent to a-dbap2d, or instantiate the object with 1 argument, where that int is the number of sounds we will be using here, so lets start with 3.

num 3

a-dbap follows matrix style inputs and outputs -- sounds are indexed from 0 to (num-1). finally, now we have to send each sound its own set of coordinates in space, and also each sound's gain in the final mixture, by sending the coordinates you are asking a-dbap2d to calculate the levels to the speakers location, a-dbap2d is speed optimized so that it ONLY calculates when you send new position coordinates for each sound.

all done configuring, now just interact with the lod above, changing the number next to it allows you to place that source snd when you interact with it, this will scale the lod coordinate system of 0-128px to 0-1, range i am using here.

in this example there is also a sample of automated path control, using hermite splines across a path, at loadbang a path is configured and the object sends out coordinates that are sent. look inside the hermite splines subpatch for configuring the number of random points in the path and alpha values for the curveness.



arguments:

messages:

outputs:

[a-objects->distance]

### a-dbap3d

distance based amplitude panning in 3d

**a-dbap3d**  
distance based amplitude panning in 3d spatium

extended from trond lossius excellent tl.dbap; thanks to mathieu chamagne and trond for ideas in extending this object

a-dbap3d calculates distance coefficients for sound panning in custom multi-channel loudspeaker configurations. receives a list of (3d) of sound positions, outputs matrix" style input messages <i o v>;

steps:

1. configure the speaker setup via speakers message (a sequential list of xy coordinates - the number of speakers configured is outputted in the second outlet)

```
speakers 0.1 0.1 0.0 0.9
0.1 0.0 1.0 0.9 0.0 0.9 0.9
0.0 1.0 1.1 0.9 0.1 1.1
0.1 0.9 1.0 0.9 0.9 1.1
```

ready made cubic system, built with 8 speakers (two square quad setups, one at z=0., the other at z=1.)

2. feed the list of coordinates to the desired sound location, prepended by the sound index

3. the output is to be connected to [matrix" X Y 1.], where X is the num of sounds to be spatialized, Y is the num speakers to your dac"

see also: a-dbap2d **a-dbap2d**

arguments:

messages:

outputs:

[a-objects->distance]

### a-distance

ndim distance estimator

**a-distance**  
ndim distance estimator

messages:

- numnodes <int> : set the number of nodes to calculate distance
- numdim <int> : set the number of dimensions
- nodes <list> : set the nodal points coordinates sequentially
- <int>, <float> : send 1d input point and calculate distances to nodal points
- <list> : (number of args sets number of dimensions to calculate distances to nodal points)
- om <0> : city block distance (default operating mode)
- om <1> : squared euclidean distance
- om <2> : euclidean distance
- om <3> : normalized proximity (0-1.)
- om <4> : chebychev distance

arguments:

- 1 sets om
- more set nodes coordinates

euclidean distance from o->p:  $\sqrt{(p-x)^2+(q-y)^2}$

steps:

1. configure nodal points and dimensions if necessary (defaults to 1d)
  - send in how many nodes in 'numnodes', after, message 'nodes' insert the list of your nodal points coordinates
2. send input list or val according to dimensions to calculate distances

1d scenario example

1. configure nodes
2. send point to test for distances

2d scenario

1. configure dim & nodes
2. send point to test for distances

arguments:

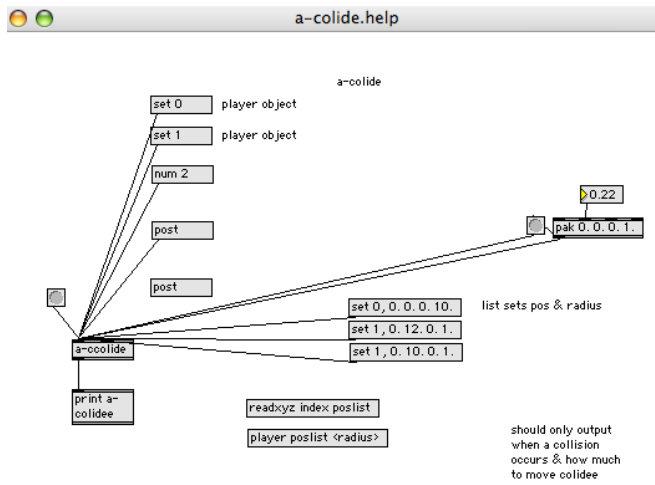
messages:

outputs:

[a-objects->computer vision]

## a-colide

simple colision model

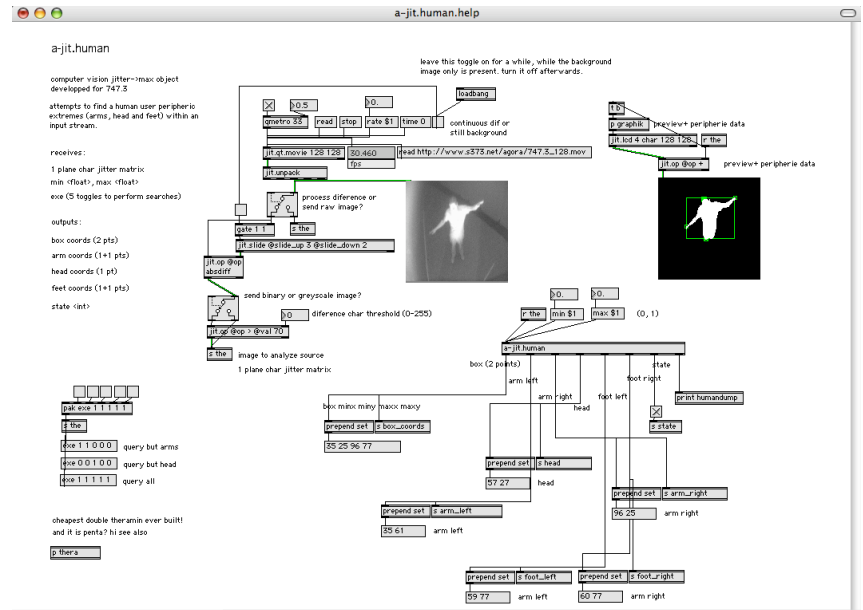


arguments:  
messages:  
outputs:

[a-objects->computer vision]

## a-jit.human

get human coordinates from video stream



arguments:  
messages:  
outputs: