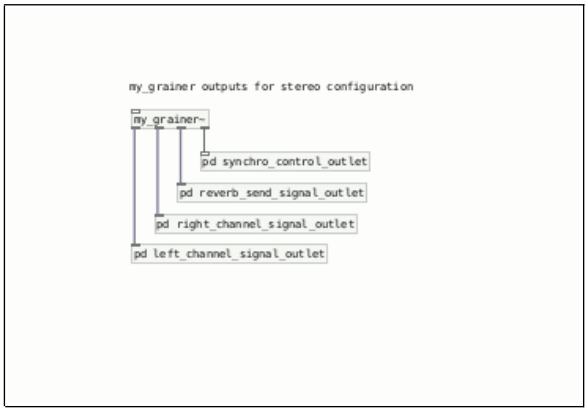
the_grainer~ a Pure Data External for granular synthesis with per grain 3D spatialisation

Pablo Di Liscia
odiliscia@unq.edu.ar

the_grainer~ is a *Pure Data* external for granular synthesis that is being developed by Oscar Pablo Di Liscia with the collaboration of Damián Anache, as a part of the research program *"Sistemas Temporales y Síntesis Espacial en el Arte Sonoro"* (Universidad Nacional de Quilmes, 2015-2019).
the_grainer~ is an improved version of *my_grainer~*:
-Di Liscia, Oscar Pablo, 2016: *Granular synthesis and spatialisation in the Pure Data environment*, PDCon 2016 Proceedings, Waverly Labs, NYU, New York, USA. pp.25-29.
http://www.nyu-waverlylabs.org/pdcon16/proceedings/

Initialization

the_grainer~ must be initialized with a value of 1, 2, 4, 9 or 16. These values mean respectively mono, stereo (*intensity panning*), quad (*Ambisonic B Format First Order*), nine channel (*Ambisonic B Format Second Order*) or sixteen channel (*Ambisonic B Format Third Order*) output.
The *quad, nine channels* and *sixteen channels* outputs consists respectively of the four (W, X, Y, Z) or the nine (W, X, Y, Z, R, S, T, U, V) or the sixteen (W, X, Y, Z, R, S, T, U, V, K, L, M, N, L, O, P, Q) encoded signals of an *Ambisonic B* first, second or third order format, which must be further decoded by the user for the available number of channels/loudspeakers of the system using appropriate externals or abstractions. If no initialization arguments are delivered, the default is 2 (stereo output). An extra (*the last signal outlet after the "direct" signal outlets*) signal outlet is always created to deliver a reverberation send signal which can be used to achieve distance cues (see below). Finally, a control outlet is added (the *rightmost one*) in order to send *bang* messages that may be useful to synchronize processes (see the *gcount* method below). The following image shows the aforementioned outlets for a stereo configuration:



General purpose messages

start: starts the audio rendering of the unit. The unit is initialized with the default value parameters (see next section), but note that there are no default for the audio and envelope tables. These must be provide by the user in order to start the audio rendering.

stop: stops the audio rendering of the unit.

pause (boolean): pauses/resumes (1/0) new grains creation. While *start* and *stop* messages controls audio rendering, *pause 0* message keeps audio active and just disable new grains creation.

gap_restart: restarts the gap between grains counter in order to synchronize multiple instances of the_grainer~.

seed (*float)*: seeds the random number generator with the *float* value delivered.

print: prints the actual parameter values of the unit in the *PD* prompt.

post_ctrl (boolean): enables/disables (1/0) console messages. These actions ignores the posting control: initialization messages; *start, stop* and *print* messages; table's loop mode; and most of input values errors.

gcount (*float)*: a nonzero value for *float* will make *the_grainer~* to send a *bang* message to its control outlet each time that *float* grains are generated. This may be useful to synchronize other processes using *asyncrhonous* granular synthesis whereas the gap time for the grains is unknown. Example: *gcount 10* will produce that a bang will be sent each 10 grains generated. Note that *gcount 1* , though possible, may result in high CPU overloading if a high grain rate is used. Default: *gcount 0* (no bangs are sent).

Distance cues messages

sound_speed (*float)*: sets the sound speed velocity in meters per second to the *float* value delivered. This is used to calculate the delay due to distance for each grain. Default= 340 m/s. Cannot be smaller than 300 m/s.

dist_exp (*float)*: sets the exponent at which the distance value will be raised for distance amplitude scaling to the *float* value delivered. Default=1. The resulting attenuation factor equals:

$$1 / D^{\text{dist\_exp}}$$

dist_delay (boolean): enables/disables (1/0) the distance time delay calculation. If the user is not going to use changing distances, switching off the delay computing will speed up the performance of the unit.

Synthesis messages

a-Audio and envelope tables management

the_grainer~ uses a “pool” of tables (max. 24 tables for audio and 24 tables for envelopes). Each table is requested by means of a special message (see below). The last table requested (either for audio or for the envelope) will be used in the next grain to be synthesized. If there are overlapping grains (i.e., that are not yet finished), these will be finished using the previous requested tables in order to avoid discontinuities. Note that if the user delete a table that is in use, audio discontinuities will happen, however.
The user is responsible for the creation of the tables either reading preexistent sound files, or with signals created “on-the-fly” by synthesis methods or real-time audio input.

gtab tablename: request the use of table *tablename* for the audio of the next grain. If the same *tablename* was previously requested, its contains is overwritten. If the user wants to request a previously requested table with its contain unchanged, then he/she should use the *usegtn* message.

usegtn (float): use the table number (*float)* for the audio of the next grain. The table (*float)* must have been requested previously. Tables are numbered starting from 0 in the order that were requested.

The two following messages will work indentically that the previous ones, except that will apply to amplitude (i.e., “envelope”) tables.

atab tablename: table name for the amp table.

useatn (float): use the table number (*float)* for the amp table.

b-Grain parameters

ga (float): gap between grains in secs., default=0.1
gar (float): gap random deviation, in secs., default=0.
N.B: The temporal gap between each grain will be: *ga* + birand(*gar*)

gs (float): grain size in secs., default=0.05.
gsr (float): grain size random deviation, in secs., default=0.
N.B: The size of each grain will be: *gs* + birand(*gsr*)

gf (float): grain read increment of the audio table (affects the frequency of the signal in the grain), default=1.
gfr (float): grain increment random deviation of the audio table, default=0.
N.B: The increment of each grain will be: *gf* + birand(*gfr*)

gli (float): grain read increment of the audio table at the end of the grain duration (affects the frequency of the signal in the grain as a glissando). Default=1, means no increment, higher values (*float* > 1) means upward glissandis, lower values (*float* < 1) means downward. (only positive values are allowed)
glir (float): grain increment random deviation of the audio table at the end, default=0.
N.B: The glissando of each grain will be: *gli* + birand(*glir*).

loop_t (float): sets the loop type of the next grain to be synthesised to *(float)*. Default=0.
(float)=0 means no loop. If the grain size exceeds the duration of the table, it will be read cyclically.
(float)=1 means forward loop.
(float)=2 means forward-backward loop.
N.B.: these must be used according the *gst, gstr, gen* and *genr* messages (see below) and the *gs* and *gsr* messages (see above). Any inconsistency between the resulting values will set the loop type to “0” (no loop).

gst (float): starting time of the function table for grains in seconds, default=0.
gstr (float): random deviation of the function table starting time, in seconds default=0.
N.B: The starting read time for each grain will be: *gst* + birand(*gstr*).

gen (float): ending point of the function table for grains in seconds, default=0.

genr (float): random deviation of the function table ending time, in seconds default=0.
N.B: The ending read time for each grain will be: *gen* + birand(*genr*).

ag (float): grain amplitude, default=1.
agr (float): grain amplitude random deviation, default=0.
N.B.: The amplitude of each grain will be: *ag* + birand(*agr*)

az (float): grain *azimuth* angle in degrees.

-When the output is set to Mono, this message is disregarded.

-When the output is set to Stereo, the range for the *azimuth* angle is from 45 to 135 degrees (45=right, 90 centre, 135=left, counterclockwise), default=90 Deg.

The distribution of the signal in the two channels is achieved using the intensity panning technique.

-When the output is set to Ambisonic B Format, the *azimuth* angle is wrapped around so as to keep it in the range of 0 to 360 Deg. Note that 90 Deg. is always centre (facing the listener and counterclockwise) also in this case, which is different than the usual angle conventions for *Ambisonics*.

azr (float): grain azimuth angle random deviation in degrees, default=0.
N.B.: The azimuth angle for each grain will be: *az* + birand(*azr*).

el (float): grain elevation angle in degrees (0=middle to 360), default=0. This is taken in account only for quad or nine channel (*Ambisonic*) output.
elr (float): grain elevation angle random deviation in degrees, default=0. This is taken in account only for quad or nine channel (*Ambisonic*) output.
The elevation angle is wrapped around so as to keep it in the range of 0 to 360 Deg.
N.B.: The elevation angle for each grain will be: *el* + birand(*elr*).

dis (float): grain distance, in arbitrary units. Minimal distance was set to 0.1. Default=1.
disr (float): grain distance random deviation.
N.B.1: The distance for each grain will be: *dis* + birand(*disr*).
N.B.2: At present, the distance cues are achieved: 1-Scaling the output by $1./\text{distance}^{\text{dist\_exponent}}$ (see the *dist_exp* method above). 2-Producing a delay for each grain which is computed taken in account a sound speed value provided by the user (see the *sound_speed* method above) and its distance. The *rightmost* signal outlet always delivers a copy of the audio grains without neither the delay nor the distance scaling to eventually feed a reverberator unit in order to reinforce the distance cue by means of the ratio between the “dry” and the “wet” signals as well as the time gap between them.

c-Arrays of discrete values

In addition to the former messages, *the_grainer~* allows the use of arrays of discrete values that are selected for each grain using the three following ways:

1)Just random with uniform probability distribution function (if only the list of values is delivered). Example: *gap_table* .5 .25 .125 .0625 will select at random a gap time out of these four.

2)Cyclic (if after the list of values the symbol CYC is included). Example: *gap_table* .5 .25 .125 .0625 CYC will cycle these four gap times.

3)Random shuffle mode: (if after the list of values the symbol SHU is included). Example: *gap_table* .5 .25 .125 .0625 SHU will select at random a gap time out of these four without any repetition until the four values are selected.

gap_table float1 float2...floatn: switches the gap size (*ga*) to discrete gap values, alternating them at random. The same message without any *float* value switches to the gap values selection mode on the basis of (*ga*, *gar*).
N.B.: The resulting gap values in this case will be a random choice out of the delivered gap values plus the *gar* deviation. Useful to create rithmic patterns.

dur_table float1 float2...floatn: the same as the former, but in this case applies to the duration of the grains.

pitch_table float1 float2...floatn: the same as the former, but in this case applies to the reading increment of the audio table for the grains (i.e., affects the frequency of the grains).
Useful for creating pitched sequences of grains. As an example, being the table fundamental frequency C5, the message *pitch_table 1 4 2 7* will produce pitched grains with their frequency being selected at random between C#5, E5, D5 and G5.
If *gf* value is different from "1", *pitch_table* pitches will be transposed according to the *gf* value, considering it as the reference.

ptr_table float1 float2...floatn: the same as the former, but in this case applies to the starting read time of the audio for the grains in the audio table.
Useful to create grains from specific regions of the audio table.
N.B.: this will set automatically the loop type to “0” (no loop).

ptr_dur_table float1 float2...floatn: the same as the former, but in this case applies both to the starting read time of the audio for the grains in the audio table and for the grains duration. The list must contain an even number of values in pairs being the first a starting read time and the second a duration value. This message overrides a random selection of starting values or duration values previously set, if any.
Useful to create grains from specific regions of the audio table with a specific duration associated.
N.B.: this will set automatically the loop type to “0” (no loop).

gtab_table float1 float2...floatn: causes a random selection of the audio table for the grains out of the values given in the list of floats delivered. The values must correspond to audio tables that are active (note that the tables are numbered starting from 0 in the order that were requested by a previous *gtable* message). The size of this list is limited to 24 and the exceeding values (if any) will be disregarded.
Example: *gtab_table 1 2 5 10* will produce grains using an audio table number being selected at random out of 1, 2, 5 or 10.
The *usegtn* or *gtable* (see above) messages will interrupt the random selection.

N.B.:
The *pause, post_ctrl, gap_restart, gli* and *glir* methods were added to the code of *the_grainer~* by Damián Anache.

(Information Updated by Oscar Pablo Di Liscia, 12/04/2018)