my_grainer~ a Pure Data External for granular synthesis

Pablo Di Liscia
odiliscia@unq.edu.ar

my_grainer~ is a *Pure Data* external for granular synthesis that is being developed by Oscar Pablo Di Liscia with the collaboration of Damián Anache and Esteban Calcagno, as a part of the research program *"Sistemas Temporales y Síntesis Espacial en el Arte Sonoro"* (Universidad Nacional de Quilmes, 2015-2019).

Initialization

my_grainer~ must be initialized with a value of 1, 2 or 4. These values mean respectively mono, stereo (*intensity panning*), or quad (*Ambisonic*) output.
The *quad* output consists of the four signals of an *Ambisonic B* first order format (W, X, Y, Z) and must be further decoded for the available number of channels/loudspeakers. If no initialization arguments are delivered, the default is 2 (stereo output). An extra (*rightmost*) signal outlet is always created to deliver the reverberation send signal (i.e., without distance scaling, see below).

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.

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.

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

Synthesis messages

a-Audio and envelope tables management

my_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.

gtable 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.

atable 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.
-When the output is set to Ambisonic 1st Order 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 (*Ambisonic*) output.
elr (float): grain elevation angle random deviation in degrees, default=0. This is taken in account only for quad (*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 cue is achieved by merely scaling the output by 1./distance. The *rightmost* signal outlet always delivers a copy of the audio grains without 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.

c-Arrays of discrete values

In addition to the former messages, *my_grainer~* allows the use of arrays of discrete values that are selected randomly for each grain. The random selection probability distribution function is uniform. The size of the arrays is limited to 24 and the exceeding values (if any) will be disregarded.

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).

(Information Updated by Oscar Pablo Di Liscia, 03/05/2016)