Assignment 3: Addendum (Granular Synthesis)

For question 1, here are some hints on how to create the audio output file from granular data.

The question suggests the following user-supplied parameters:

a) duration of individual grains (in seconds; usually between 1ms and 100ms).
b) rate of grain generation (density of cloud). This can be # grains/second.
c) pitch of grains
d) location of grain cloud within source wave table: center location, and span of cloud in the table (ie. min and max locations from center, from which grains are generated).
e) overall duration of grain cloud (total number of grains generated)

Firstly, you can skip parameter e if you compute grain generation as suggested below.

You can add one more parameter:

f) duration of output audio file (seconds). Note that the size of this buffer (array) should be:

output_buffer_size = sampling rate X duration.

For example, if the sampling rate is 44,000 samples/second, and the duration is 10 seconds, then the output buffer should be 440,000 samples in total.

Also, grain sizes (# samples in a grain) are computed as:
grain_size = grain duration X sampling rate.
For example, if a grain duration is 1ms, and 44,000 sampling rate:
grain_size = 0.001 sec X 44,000 samples/sec
= 44 samples.

There are multiple steps in generating grains for output:

(i) selecting the raw sample for the grain from the source wave table. This may involve finding a random central location, then its extents before and after that position, and then randomly copying a grain from within that window. Note that the 'ends' of the grain may go outside the bounds of the window. This is OK, so long as you don't go outside the wave table (out of array bounds!).
(ii) reformulating the grain by playing it at a new pitch (see assignment 1), and then enveloping the played grain so that it starts and stops at value zero.
(iii) putting the grain in the output buffer for eventual output to an audio TXT file.

The rest of this will explain step (iii) above. There are two basic ways this can be done.

a) The easiest way to do this is to copy the reformulated grain ANYWAY in the output buffer. You should ADD the samples to existing samples in the output, because grains can overlap. Although this is easy, one problem with this is that it doesn't permit easy real-time playback, should you decide to enhance your assignment question into a stand-alone granular synth application with real-time feedback and user controls.

b) A more involved technique is to pretend the output buffer is similar to a "real-time" audio output port. By scanning the output buffer's array entries, you can decide at each sample

point whether a new grain will be added into the buffer at that point. This is done probabilistically (using random number generators). Here's how you do it. (over)

You need these parameters:

Sample playback rate: # samples/sec
grain generation rate: # grains/sec
Grain duration: seconds (between 1ms and 100ms)
Total output file duration: seconds

1. Your output buffer will need to hold this many samples:

Output buffer size = output file duration X output sampling rate.

eg. If sampling rate is 44,000 samples/sec, and duration is 10 seconds, then the buffer must hold 440,000 samples.

2. You then compute the probability that a new grain will start at any sample point in the output buffer.

Probability grain starts = (grain generation rate) / (sampling rate)

Grain size is computed as: Grain size = (grain duration) X (sampling rate)

e.g. Let the sample rate = 44,000 samples/sec
and grain generation rate = 100 grains/sec

So: Probability grain starts = (100 grains/sec) / (44,000 samples/sec)
= 1/440
= 0.00228

Then, if you are scanning the output file start-to-end (sample by sample, as if going forward in time), the following will randomly generate grains at approximately the specified grain generation rate. Note that grains can overlap!

```
Initialize output BUFFER to 0.
LOOP:  i = 1 to Output_buffer_size
            IF rand() < prob_grain_starts
            THEN
                  Create a grain from source wave.
                  Mix grain to BUFFER[i,..., i + grain_size - 1]
END LOOP;
```

Note: - Reformulated grain is enveloped, pitch-shifted.
- Beware of mixing outside the output_buffer! (memory crash in C, complaints in Java)
- Beware of clipping! You will need to weight grains before mixing them in the buffer.