**Music Programming Project – Granulator**

**Supporting Documentation**

Author: Ian Baxter

Year: 2015

Title: v9f08 - Granulator

Table of Contents

Design and Structure of Code2

User Manual4

Understanding The Program4

Troubleshooting5

Examples………………………………………………………………………………………………5

Design and Structure of Code

First the name and number of arguments input by the user is declared and then the global functions and variables are declared so that they can be utilised throughout the program.

Next in main the required variables are declared and psf props is declared. This allows for the reading and conversion of audio files along with relevant variables such as opening input and output files. The number of arguments input by the user is then checked otherwise print a usage error. Next define the variables and convert them to floats and gain from SPL to DB. Then check that values input by the user are suitable, for example attack/decay not zero and smaller than grain size and that pan and warp are greater than zero. For an unsuitable value the process is stopped and a corresponding error message is printed.

Start portsf and open input audio file from input argument. Then check if the input is in stereo, if yes, make the variable strereoIn equal 1. Then set output as stereo and create output file from output argument. Check both input and output are greater than 0.

Create a buffer for the whole file by calculating the total number of samples then converting this to the number of bytes required and then allocating the buffer from the number of bytes. Check if the buffer is null, if so clean up files and end the process. Then clean buffer.

Next create a grain and loop the process until written\_frames equals total\_frames. This is done by first calculating random grain duration from the minimum and maximum grain size and then calculating the number of frames in the grain. If the grain tries to write over total duration of the file then break the loop to prevent crashing. Next convert the number of samples to bytes and allocate another buffer from the number of bytes which will be used for each grain loop. If stereoIn equals 1 then a stereobuffer is required where the number of bytes is multiplied by the number of channels. Next calculate the size of the attack and decay frames from the user input values. Then calculate a random read positon and seek which takes a random part of the original file with which to make the grain. Copy the input to the output using the copy function and then free the grain buffer and make the buffer value null so that it can be used again for the next grain.

After total\_frames is filled write the output file from Bigbuffer and total\_frames and check that frames equals total frames otherwise end the process. Finally clean up the files and buffers to save processing power, finish portsf and send a completion message letting the user know that the granulation has been successful.

The functions are written after main. Open\_input opens and reads the properties of the input audio file (name, stereo or mono). Open\_output creates an output audio file with relevant properties (name, stereo or mono). An error is printed if either audio file is less than zero. Allocate buffer/stereobuffer allocates the number of bytes needed and then makes sure the buffer does not equal zero.

In the copy function the relevant variables are declared, such as the ones needed for panning. If the input is in stereo the input file is read and then the buffer is calculated as an average value between consecutive frames in the stereobuffer. If the input is in mono then it just reads the input file in to buffer. Check that frames which is the number of frames expected, is equal to num\_frames from the buffer otherwise return negative. For the random panning of grains a random x value between -1 and 1 is first calculated. This is then used to calculate a left and right balance value which determines the random spatial placement. A series of checks are then made to make sure that the balance values are within a specific range for the speakers. The final panning factor is then calculated using the balance values and determines the levels of the left and right channels for each grain. Next implement mixing of grains from a position until num\_frames is reached. This is done by implementing the panned grains into two consecutive positions in Bigbuffer. After the grain has been mixed increase the position so that when the next grain is created it does not overwrite the previous one.

The clean\_up function clears previous values for in and out files and also clears the buffer so that the looping processes can repeat using the same buffer and minimal processing power. Attack increases the attack factor from zero by an increment calculated by the user input attack value. This occurs over the length of the value for attack\_frames and has the effect of steadily increasing volume by multiplying it by the buffer value for a given position. The buffer is also multiplied by the warp value to randomise the calculated length of the attack here. The decay function operates in the same way as attack function except it works to decrease the endBuf value and steadily lower the volume.

User Manual:

1. Open command prompt and navigate to the folder containing the granulator file and input audio files. (Input file names must not have spaces and files must be in either WAV of AIFF file format)
2. Type the name of the granulator program “granulator” followed by the name of the input file and the name of the output. Remember to include the file extension after the file name. Then input the desired duration, minimum grain size, maximum grain size, attack, decay, sampling rate, density, gain, pan and warp (a space is required between each argument).
3. Press enter and then the program will produce a granulated copy of the original file in the same folder.

Understanding the Program

**Duration** is the length of the output file in seconds.

**Minimum grain size** is the smallest possible size of grain created in milliseconds.

**Maximum grain size** is the largest possible size of grain created in milliseconds.

**Attack** is the time in milliseconds for each grain to go from 0 to maximum volume.

**Decay** is the time in milliseconds for each grain to go from its maximum volume to 0.

**Sampling rate** determines the number of samples created in the output file. The higher the value the larger the file size but the better the sound quality. The recommended value is 44100 however other rates such as 48000, 96000 can be used. Other lower values may be used and this can be used to create interesting glitches in the sounds.

**Density** determines how closely spaced together each grain is to each other.

**Gain** alters the maximum volume of the output.

**Pan** determines how wide the output is panned across the speakers

**Warp** randomly alters the attack and decay of each grain.

Troubleshooting

If output file is clipping decrease the gain value.

Example Parameters

1. granulator input1.wav output1.wav 5 180 300 80 80 44100 15 1 2 4
2. granulator input2.wav output2.wav 5 50 80 10 30 44100 30 -3 5 2
3. granulator input3.wav output3.wav 5 120 150 20 30 44100 15 2 3 5