Synthesis of background chatter

Fergus Jack Doherty-Eagles (260852322)

Introduction

Early in the year, I was watching a video by one of my favourite musicians on YouTube in which he improvised a conversation between a couple in a restaurant. For the first few minutes, he recorded a few clips of himself speaking nonsense, and used a looping device to layer them over the top of one another. With only three clips, he was able to generate fairly realistic background chatter. It did have a repetitive quality to it (due to the looping), but it got me thinking about the potential of this approach. How realistically, and simply, can background voices be generated?

For my project, I wanted to create some kind of user-friendly program to generate background chatter. In theory, a program in which the user could record many different tracks and add them together would be the most realistic - but this would be very time-consuming. I decided to restrict the user input to only one voice track, and use digital effects and manipulation to create the background chatter from that. Under this limitation, the project became quite experimental, and several questions arose in my mind:

1. How can one conversational track have a noise-like quality to it?
2. Can the setting/room be varied?
3. What parameters are the most important?
4. How can the qualities of a single voice be changed to sound like multiple different voices with different timbres?

This report summarizes my approach, which mainly took principles from granular synthesis. To round out the project, I also used some simple synthesis techniques we discussed in class to create a coffee machine sound effect that can be controlled by the user. This gives the program a specific ‘sub-function’ - generation of background cafe ambiance! It can be recorded into an output audio file and then used externally in whatever context the user needs.

Initial Ideas

After learning so much about the ‘fundamentals’ of synthesis in MUMT 307, and building sounds from the ground up, I thought that formant synthesis might be a possible approach to achieve the goal of background chatter. It quickly became apparent that this would be far too difficult, and that in the absence of the plosives and fricatives of natural speech, it would not sound believable.

Granular synthesis of a pre-recorded audio file seemed like a more ideal solution - by playing back different snippets of the file in quick successions, I would be able to preserve the complex timbre of the human voice, but also introduce a noise-like quality in the randomness. I wasn’t sure what grain duration would be the most ideal - by introducing it as a parameter, I would be able to test different durations and hear for myself what the effect was!

I thought that the apparent number of voices in the room could be changed simply by changing the number of instances of a ‘random snippet player’ operating simultaneously.

Implementation

Granulation - The ‘snippets’ patch

The granulation of the sound file was the trickiest part of the design. I fixed the buffer length at 40 seconds, and had to figure out a way of randomly playing through different parts of the file in quick succession. To avoid pops and clicks, each grain also had to have a window applied to it - but since the grains were of constantly varying size, the domain of the window had to be constantly updated. This led to some difficulties in synchronization, but after reviewing Max/MSP’s messaging order, I was able to resolve the issue.

The start and end times of each grain were randomly selected by first choosing an end time, and subtracting an arbitrary grain duration. The grain duration was lower-bounded by the minimum grain duration, and upper-bounded by five seconds. If the start time ended up negative, I took the absolute value. The grain duration set the domain of the envelope and triggered it, while the start and end times were sent to a “play” message that triggered the playback of a snippet of the buffer. When the line~ object of the window finished, it started the process over again, allowing for indefinite repetition. To stop the voices playing, there is a gate in the snippets patch which disconnects the line completion bang.

Creating multiple instances of the snippets patch

The next step in the implementation was to have many snippets patches playing at once. Luckily, Max is a very powerful program, so creating multiple instances using the poly~ object was the simplest and most effective way to achieve this. A slider in the UI controls the number of patches operating simultaneously, and they are each input the same minimum grain duration number.

Rounding out the features of the patch

The patch allows the user to record a sound file to be used, or select one from their file system. The global gain of this file can be adjusted - this is relevant when a lot of instances of the snippets patch are being used (upwards of 20). Without lowering the gain of the file, the addition of multiple playbacks creates significant distortion.

One problem was that with only one voice, it sounded very monotonous. To remedy this, I added optional pitch shifting. The pitch shift was implemented using the pitchshift~ object in max - I limited the range so that it sounded like it could feasibly be several different voice sources.

Another feature that I added was a simple Schroeder reverb, using cascaded all pass filters and feedback comb filters. This was an attempt to simulate a more ‘echoey’ room. The patch can be found within the snippets patch, on the right next to the pitch-shift patch.

Coffee Machine

Finally, I created a subpatcher ‘coffee’, which uses a simple feedback comb filter and high-pass filter in series to replicate the sound of a coffee machine. This can be added to the chatter signal to create the cafe ambiance (it is turned off by setting the slider to zero).

<u>Results</u>

Here is the base sound file I used for the examples (there are a few seconds of silence at the beginning): My Audio - 0019.wav

Overall, I think the sound is fairly realistic! Here are some simple examples of changing the number of effective voices playing. These samples all have the same properties except a different number of snippet patch instances running simultaneous

Basic chatter, no pitch-shift/reverb, minimum grain 500ms, 5 instances:chatter1.wav
Basic chatter, no pitch-shift/reverb, minimum grain 500ms, 15 instances:chatter2.wav
Basic chatter, no pitch-shift/reverb, minimum grain 500ms, 30 instances:chatter3.wav

I found that beyond 20 voices, it became much less believable. My theory for this is that, in any sound file there will be certain plosives/fricatives that particularly stand out, and with so many instances of snippet patch playing at once, there is almost always one playing one of these stand out moments. The ‘s’ sound is too ubiquitous in the 30 instance sample, for example.

One happy coincidence of the gate controlled by the ‘stop/reset’ button is that the sound doesn’t instantly cut off - each grain will finish playing. This (remarkably effectively!) simulates the effect of a room full of people stopping talking, where several voices may take slightly longer to peter out. I laughed the first time it happened out of surprise! Here is an example:

Voices petering out:chatter4.wav

Here is a comparison with the different minimum grain durations. To make the effect more clear, I set the voice number to only 3:

3 voices, no pitch-shift/reverb 10ms minimum grain duration:chatter5.wav
3 voices, no pitch-shift/reverb 3000ms minimum grain duration:chatter6.wav

Above 10 voices, the difference isn’t nearly as perceptible. Here is a similar pair of examples:

3 voices, no pitch-shift/reverb 10ms minimum grain duration:chatter7.wav
3 voices, no pitch-shift/reverb 3000ms minimum grain duration:chatter8.wav

This answers one of my questions about the important parameters controlling background chatter generation. It turns out that the minimum grain duration really doesn't make too much of a difference. I had initially thought that with longer grain durations the sound would be more believable (since it is closer to just layering several sound clips). In hindsight, it would be important to also have a control for the MAXIMUM grain duration to test this hypothesis properly and come to a confident conclusion.

Below are examples of the various effects I implemented:

8 voices, 500ms minimum grain duration, pitch-shifting:chatter9.wav
8 voices, 500ms minimum grain duration, reverb (not good):chatter10.wav

The pitch-shifting works quite well to give an impression that there is not just one voice source in the room.

For the reverb, I found that the Schroeder implementation did not allow for small changes in the amount of reverb, and when the effect was applied it sounded, frankly, quite demonic. This was a feature I added close before the presentation day so I did not have time to improve it.

Finally, here is an example with the coffee machine playing! This is the program operating in what is its most realistic manner (after my experimentation): chatter11.wav

Further Improvements/Final Thoughts

Several things could be improved in future versions of the patch. One particularly effective feature would be to have a random gain set on each instance of the snippets patch. I think this might effectively capture the sense that some people in the room are further away. Another thing I would like to add is the automation of global gain of the file depending on the number of voices selected. At the moment it is purely a matter of trial and error, adjusting the level until distortion is removed. Finally, I would like to be able to change the window shape, to see if that makes a difference - unfortunately I only think of this modification as I am writing the report.

I asked some of my friends how many voices it sounded like, to see if the number of snippets instances was closely correlated with the apparent number of people in the room. The responses were very varied, however typically for 5 instances, the guesses were almost perfect ! I would often play it without explaining that it was my project first however, so often the responses were 'it's the same person sped up!' or other similar (somewhat true) observations.

I had a great time experimenting with this idea!