Procedural Breath Sounds in Csound

Ken Kobayashi

MU610A

Intro:

The human voice is an artform studied and perfected by many. However, recent technologies have begun to imitate this artform. Vocal synthesis is a subsection of sound synthesis that generates human singing with technology, often using computer algorithms or models to mimic vocal sounds. This innovation represents a unique intersection of art and science, where the intricacies of human vocalization are replicated or even enhanced through machines. As vocal synthesis continues to evolve, it spans a wide array of techniques and technologies, each with its own unique approach to recreating the nuances of human voice.

The development of vocal synthesis started at the 1939 World's Fair when Bell Labs revealed the 'VODER,' an early electronic speech synthesis device.[1] Bell Labs' innovation in synthetic speech led to the first computer to sing in 1961, performing 'Daisy Bell.'[2] This performance was created using an IBM 704 computer by John L. Kelly, with accompaniment from Max Matthews.[3] The 'VODER' was operated in real time by a performer pushing pedals with their hands and feet to trigger consonant and vowel sounds.

The development and adoption of vocal synthesis have grown rapidly, notably leading to products such as *Vocaloid* by Yamaha. The *Vocaloid* program began development in the year

---

[1] "Background: Bell Labs Text-to-Speech Synthesis: Then and Now." *Bell Labs: Background: Bell Labs Text-to-Speech Synthesis: Then and Now*, 7 April 2000, <web.archive.org/web/20000407081031/www.bell-labs.com/news/1997/march/5/2.html> Accessed 25 Oct. 2024.

[2] O'Dell, Cary. "Daisy Bell (Bicycle Built for Two)." *"Daisy Bell (Bicycle Built for Two)"—Max Mathews, John L. Kelly, Jr., and Carol Lochbaum (1961)*, www.loc.gov/static/programs/national-recording-preservation-board/documents/DaisyBell.pdf. Accessed 25 Oct. 2024.

[3] "Background: Bell Labs Text-to-Speech Synthesis: Then and Now."

2000, releasing in 2003. It is a vocal synthesizer which relies on a soundbank, a collection of recordings of a singer or voice actor singing every possible vowel and consonant combination in a language. Around 500 combinations are required for Japanese vocal synthesis, while English requires 2,500.[4] These recordings are then concatenated together to generate words and have the pitch and timbre adjusted spectrally to synthesize singing.[5] Voice banks must be provided MIDI data (musical pitch and timing information) as well as phonetic information for lyrics by the user.

Vocaloid's popularity is most evident in Japanese pop music. One of the most popular songs that feature *Vocaloid* singers, "Vampire" by producer DECO*27, has garnered over 80 million views on *Youtube*.[6] "Vampire" uses the "Hatsune Miku" voicebank, which has been personified as a blue haired virtual singer, and is the most widely used and popular *Vocaloid* voicebank. *Vocaloid's* continued popularity can be seen by accessing the *Billboard Japan Hot 100*, a ranking of the most popular songs in Japan. Among the hottest Japanese artists, J-pop bands, and K-pop groups, numerous *Vocaloid* songs can be found. Today, "Tetoris" by Hiiragi Magnetite and "Monitoring" by DECO*27 are found in the top 100.[7] *Vocaloid's* immense popularity within Japanese pop music is proof that synthesized vocals can resonate in today's music trends..

After the rapid rise in popularity of *Vocaloid* in Japan, 'UTAU,' a free, shareware alternative was created in 2008. Through 'UTAU,' one is able load a voicebank, and play back

---

[4] Kenmochi, Hideki. "Singing Synthesis System VOCALOID and Hatsune Miku." *Research on Intellectual Property Rights on Digital Content*, Mar. 2008, pp. 38–42, http://www.dcaj.or.jp/project/report/pdf/2007/dc08_03.pdf#page=38

[5] Kenmochi, Hideki, and Hayato Oshita. *Vocaloid – Commercial Singing Synthesizer Based on Sample Concatenation*, homes.esat.kuleuven.be/~spch/www.interspeech2007.org/Technical/ssc_files/Yamaha/VOCALOID_Interspeech.pdf. Accessed 25 Oct. 2024

[6] DECO*27. "DECO*27 - The Vampire Feat. Hatsune Miku." *YouTube*, YouTube, 9 Mar. 2021, www.youtube.com/watch?v=e1xCOsgWG0M.

[7] "Billboard Japan Hot 100: Charts." Billboard JAPAN, Billboard, www.billboard-japan.com/charts/detail?a=hot100. Accessed 15 Jan. 2025.

their voicebank similar to *Vocaloid*. [8] However, voicebanks for 'UTAU' are created by users, unlike *Vocaloid* voicebanks, which are created by Yamaha. This has fostered an online community that shares and distributes user made voicebanks for anyone to download and use.[9]

In the past couple of years, AI has been trained to sing. AI technology functions on machine learning, in which a computer is fed vast amounts of data to learn from until it is able to mimic what it was given and produce a desired output.[10] AI has empowered vocal synthesis to new heights, its abilities extending throughout the entire musicmaking process. The website, 'Suno,' displays this technology in an accessible manner. 'Suno' takes a short description of a song, then generates a full song matching the user input.[11] [12]

Another branch of AI voice synthesis is with voice models. Voice models are created through machine learning with data restricted to a desired singer or voice. Once the training of the voice bank is complete, the user provides the model with a vocal sample. The voice bank can then be replicate the sample in the style of the singer it was trained on. Unlike 'Suno', a proprietary AI model[13] based on astronomical computation power, voice models can created by anyone with a quality data set and computation power. Online AI communities like 'AI Hub' host in-depth online guides on creating one's own voice model using open source models running on desktop PC's and cloud computing services.[14]

---

[8] cheesum. "Voicebanks 101." Utau.Us, 20 Feb. 2016, utau.us/vb.html.
[9] Anonymous. "原音ファイルセット（UTAU用音声ライブラリ）." 歌声合成ツール *UTAU*, アットウィキ, 11 May 2018, w.atwiki.jp/utau2008/pages/35.html.
[10] Ibm. "What Is Machine Learning (ML)?" *IBM*, 17 Oct. 2024, www.ibm.com/topics/machine-learning
[11] *Suno*, suno.com/. Accessed 25 Oct. 2024
[12] "Ai Music." *Microsoft Research*, 9 Sept. 2022, www.microsoft.com/en-us/research/project/ai-music/
[13] Hiatt, Brian. "A CHATGPT for Music Is Here. inside Suno, the Startup Changing Everything." Rolling Stone, Rolling Stone, 22 Mar. 2024, www.rollingstone.com/music/music-features/suno-ai-chatgpt-for-music-1234982307/.
[14] Julia, and Eddy. AI Hub Docs, AI Hub, 21 Oct. 2024, docs.aihub.gg/.

Many techniques are used by singers to enhance their performance, many of which are implemented within *Vocaloid* and AI. The basics of singing, such as pitch and lyrics are clean and clear, with more advanced techniques such as vibrato, whispers, vocal fry, and breathiness can be achieved with some work.

However, despite many innovations, a clear weakness to synthetic vocals has reamined: synthetic vocals sound distinctly robotic. One of the most obvious issues that these programs face is breath. Breath is a limit for humans, forcing singers to pause for air, confining the length of phrases. Breath has defined human perception for music, as the pause for air shapes phrasing in singing, and, as a consequence, has spread to all of music. Instruments that do not have such physical limits often mimic phrasing styles of singing, as singing—and therefore breathing—has defined our musical experience.

For this project, I will focus on a small part of this issue. The sound of a singer breathing for air is a small but critical element of a vocal performance. In the song, 'Slowly' by Asu, one can hear 3 different types of breaths in the first 10 seconds alone: a short, sharp breath, a medium breath, and a long, soft breath.[15]

Modern vocal synthesis techniques make attempts to mimic breath sounds. *Vocaloid* libraries often come packaged with wave files of breath samples. However, these breath samples are limited in variation and sound unnatural in some contexts.

Voice models are able to generate breaths sounds in specific conditions. The primary requirement is for the voice model to be trained on data that includes breath sounds. This is required in order for the model to be able to produce breath sounds. With voice models often

---

[15] Asu. "Asu Op. 2 - Slowly [Original MV]." *YouTube*, YouTube, www.youtube.com/watch?v=enFLXbbXWdk. Accessed 25 Oct. 2024

being created by members of online communities, there is no standardization or quality control that ensures breath sounds are included in training data. By listening to demos of voice models posted on the AI Hub discord server, clear differences in quality can be heard, with some models having clear and natural breath sounds, while others do not have any.[16] Furthermore, the output of AI models is heavily dependent on prompts. Prompts are "your input into the AI system to obtain specific results." [17] For a generative AI like 'Sona,' the prompt is the user input text, while for voice models synthesis, it is the vocal sample that the user inputs. Obtaining ideal breath sounds from a AI model requires careful AI prompting that is infeasible for a program like 'Sona.' However, breath sounds generated from voice models can be manipulated through manually editing the breath sounds in the input vocal sample. This process, as with any form of AI prompting, is extremely tedious and fickle, as small changes can drastically change the output of an AI model in a unpredictable fashion.

Therefore, in this project, I will be creating a Cabbage instrument that uses Csound to create breath sounds. I will first analyze breath samples, using audio analysis tools to observe the audio spectra of the samples. Through this analysis, I will create formant tables based on the samples. A Cabbage instrument will synthesize breaths through additive synthesis based on these formant tables. This instrument will include customizable parameters, adjusting the envelope and fileter of the breath. This instrument will be able to supplement synthetic vocals, enhancing their human-like quality by streamlining the process of adding breath sounds.

---

[16] "AI Hub Discord." *AI Hub Discord*, Discord, discord.com/invite/aihub. Accessed 15 Jan. 2025.

[17] "Effective Prompts for AI: The Essentials." *MIT Sloan Teaching & Learning Technologies*, MIT, 16 Sept. 2024, mitsloanedtech.mit.edu/ai/basics/effective-prompts/.

Research

In order to synthesize breath sounds, a formant table is used. A formant table consists of a series of frequencies that are shaped by the vocal tract's configuration. These resonant frequencies appear as peaks on the frequency spectrum of a person's voice.[18] Importantly, formants are independent from the fundamental frequency. [19] In other words, regardless of the pitch of a sung vowel, the resonant frequencies remain consistent for each singer. Each vowel sound has its own formant table, characterized by three major resonant frequencies, distinguishing each vowel sound. [20]

Basic vocal synthesis can be achieved with a formant table using additive synthesis. Additive synthesis involves amplifying specific frequencies corresponding to the vocal tract's resonant formants, thereby recreating the peaks in the frequency spectrum defined by the vowels formant table. In terms of synthesis, vocal synthesis through additive synthesis is relatively simple, requiring only a resonance filter per each entry in a formant table.

For breath sound synthesis, I hypothesize that a similar approach can be applied. The primary objective of this study is to create accurate formant tables for replicating breath sounds. This process will require audio analysis of reference breath sounds and extraction of the frequency, band width (a measure of how wide the peak is), and gain of each resonant frequency. The second objective is to create an instrument that uses the formant tables created to perform additive synthesis to generate breath sounds. The additive synthesis will require exploration to design a signal processing method that achieves desirable output of breath sounds. Finally,

---

[18] "Formants." *Acoustic Phonetics: Formants*, University of Manitoba, home.cc.umanitoba.ca/~krussll/phonetics/acoustic/formants.html. Accessed 19 Jan. 2025.
[19] Martin, Sean Thomas. "Formants Explained and Demonstrated." *YouTube*, 2019, youtu.be/jpbFnsusfp0?si=GTaCHP28B9kq0fyY.
[20] "2.2. Formants of Vowels." *Phonetics and Phonology*, corpus.eduhk.hk/english_pronunciation/index.php/2-2-formants-of-vowels/. Accessed 19 Jan. 2025.

further development of the instrument will be done to streamline its usage for intuitive and flexible generation of breath sounds. The user of the instrument should have controls available, allowing users to generate a wide range of breath sounds suited to different vocal performances.

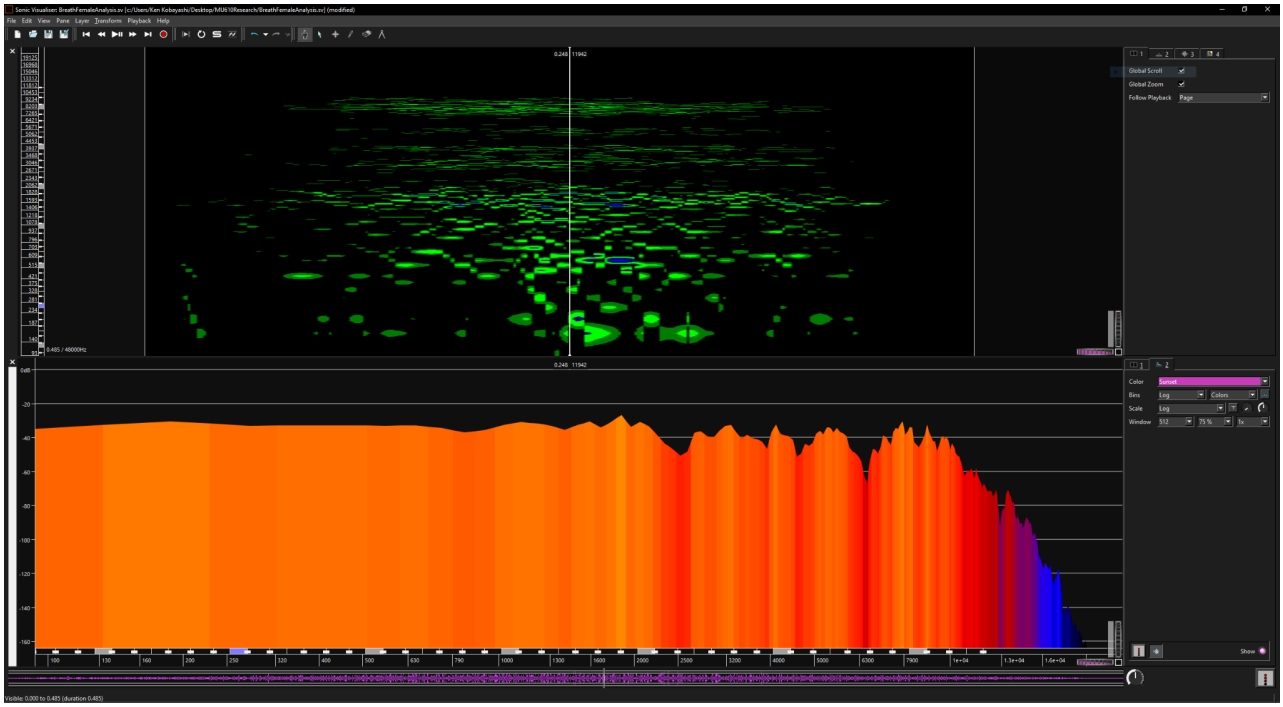
The primary difference with breath synthesis from traditional vocal synthesis is the absence of pitch in breathing, as breaths are typically noise-based with no defined fundamental frequency. While vocal synthesis uses an oscillator with a fundamental such as a square wave to produce a pitch, breath sounds can be generated using noise as the input oscillator. A key aspect of this project will be investigating the best type of noise to use, given the variety of noise types available for synthesis.

Audio Analysis

The first step in this project was to create a formant table for a breath sound by analyzing a reference audio sample.

A suitable sample was found in the online audio library Splice by searching for "breath sounds." The sample, titled *PantFemale_BW.16137.wav* by Blastwave FX, is a 16-second recording of heavy breathing, offering a diverse selection of breaths from the same individual.

To analyze the audio, the software *Sonic Visualiser* was used. This free, open-source application is designed for audio analysis and visualization, providing tools such as spectrograms to inspect the frequency content of audio files. After importing the sample into *Sonic Visualiser,* a single breath that sounded most representative was selected. I carefully scrubbed through the audio and identified a frame where the breath was loud and clear, with observable resonant frequency peaks. Capturing a frame where the breath was prominent ensured stable formants and reduced noise-induced error. Below is a screenshot of the *Sonic Visualiser* interface:



The bottom half of the screen is the spectogram, a graph that shows the frequencies contained in the audio.The horizontal axis represents frequency in Hertz (Hz), while the vertical axis shows each frequency's amplitude in decibels (dB). This visualization provides a clear picture of the wide range of frequency content present in the audio, spanning from the lower limit of 0 Hz to the upper limit of 22 kHz, which is the maximum range of the file type.

From this data, one can make an informed decision on the type of noise to use as an input oscillator. One option for the input noise is white noise, which is pure and random noise. White noise contains an equal amount of energy across all frequencies, resulting in a flat line on a spectrogram. In this project, white noise could be processed through additive synthesis to amplify specific resonant frequencies that match the reference audio sample. Afterward, a low-pass filter could be applied to attenuate the higher frequencies, replicating the gentle slope observed in the high-frequency region on the spectrogram.

Another option for the input noise is pink noise, also known as fractional noise. Pink noise has a power spectral density that decreases as frequency increases. This characteristic distributes energy more evenly across octaves, making it sound more balanced and natural to the human ear compared to white noise. Spectrally, pink noise is useful for this study as it inherently mimics the attenuation of higher frequencies seen in the reference sample. By applying additive synthesis to pink noise, specific frequencies could be amplified to match the spectral flatness of the reference audio.

Based on this analysis, both white noise and pink noise could serve as input oscillators for this project. Experimentation with both types of noise will be performed to determine which is more fit this study.

The analysis also revealed distinct formants, visible as bright orange peaks on the spectrogram. Unlike vowel sounds, which typically exhibit three key resonant frequencies, the breath sample showed four prominent peaks, with additional ones discernible upon closer inspection. Determining whether these extra peaks are necessary for accurate breath synthesis required further experimentation.

To extract numerical values of each resonant frequency from *Sonic Visualiser*, one can hover over a point on the spectrogram and see the frequency and amplitude value of the point. Using this feature, I analyzed six formants for their center frequency, bandwidth, and gain, constructing the following formant table for female breaths:

Formant Number	Center Freq. (Hz)	Band Width (Hz)	Gain (dB)
f1	1600	200	0
f2	3100	300	-6
f3	3950	200	-7
f4	5350	500	-8
f5	8525	1000	-6
f6	13400	150	-15

Using this formant table, additive synthesis with a noise generator can theoretically recreate a breath sound resembling the reference.

Testing

Before programming the instrument, a suitable digital signal processing (DSP) strategy needed to be identified. This was achieved using *Ableton Live 11*, a digital audio workstation

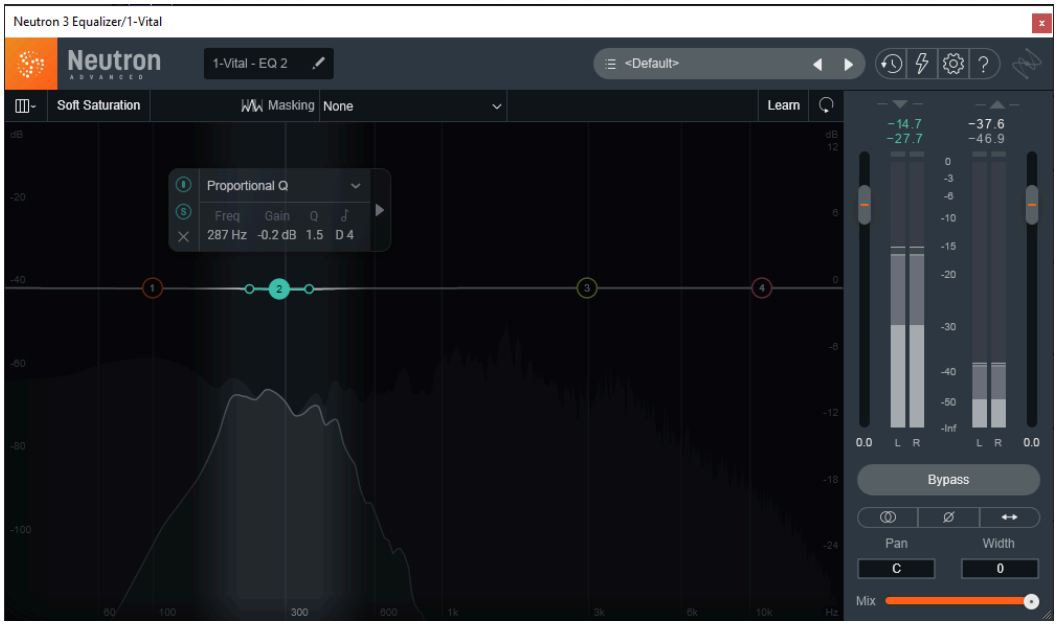
(DAW) with real-time audio feedback and an intuitive interface, allowing for quick iterations and testing.[21]

In *Live*, several plugins were used to streamline the testing process. For a sound generator, the synth plugin *Vital* by Vital Audio is used. *Vital* is a free to use, wavetable synth.[22] This synth includes many noise generators, including white and pink noise that is used in this project.

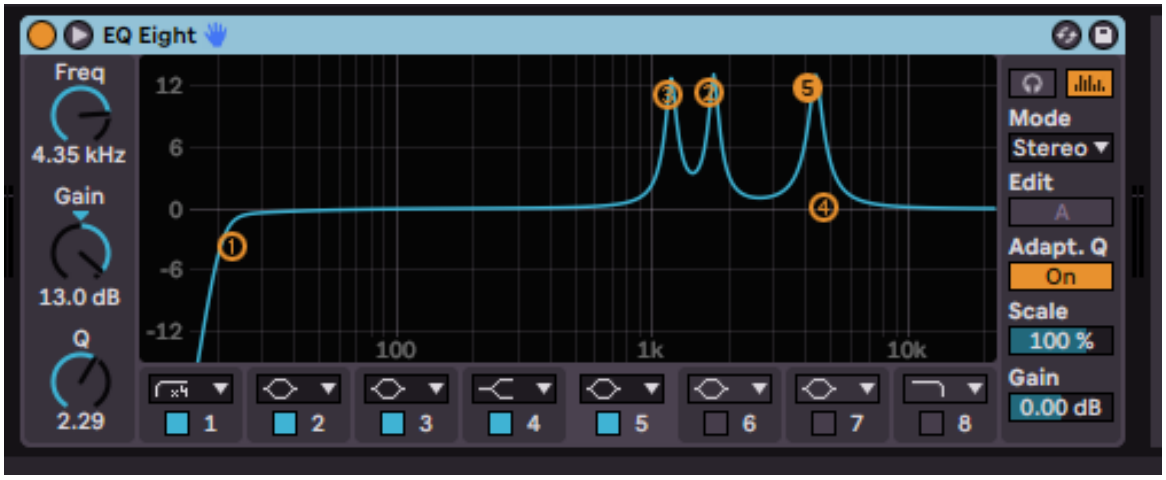
Another useful plugin is *Neutron 3 EQ* by iZotope. *Neutron 3 EQ* is a graphical EQ, a type of DSP which allows precise manipulation of specific frequency bands. Specifically, the plugin's ability to solo a EQ band and listen to the band in isolation allowed close monitoring of formants. Using *Neutron 3* ensured that the synthesized breath sound closely matched the

[21] "Ableton Live." *Ableton*, www.ableton.com/en/live/what-is-live/. Accessed 21 Jan. 2025.
[22] "Spectral Warping Wavetable Synth." *Vital*, vital.audio/. Accessed 20 Jan. 2025.

spectral characteristics of the reference sample.



Armed with these softwares, different methods of additive synthesis were tested. The first technique I tried was a graphical EQ to amplify the formants. In this method, the first 3 formant frequencies were boosted in order to create peaks in white noise.



This produces a tinny, hollow wind sound (ref. LiveOutput1.wav), and is not similar to breathing in any sense. There were many issues with this approach, most due to lazy and quick testing. The band width of each formant was ignored, as the graphical EQ used in the test

accepted Q-values, a number derived from band width. The gain was also ignored, with all 3 formant frequencies set to max amplification.

The second method that was attempted was using parallel band filters. This technique filters out the desired frequencies, and adds them back to the original sound, amplifying the frequencies. This technique was most true to the basis of additive synthesis. Each frequency's volume was adjusted to approximately match the formant table, and the inherent wide bandwidth of band pass filters would create smoothen the gaps between each formant frequency.

This technique creates a sound (ref. LiveOutput2.wav) more similar to a breath, although still tinny and wind-like. Though an improvement from the first method, the output is far from what could be called a breath sound.

Through spectral analysis within *Live*, it was observed that this method causes very shallow peaks in the spectrum, but smoothens the amplitudes between resonant frequencies.

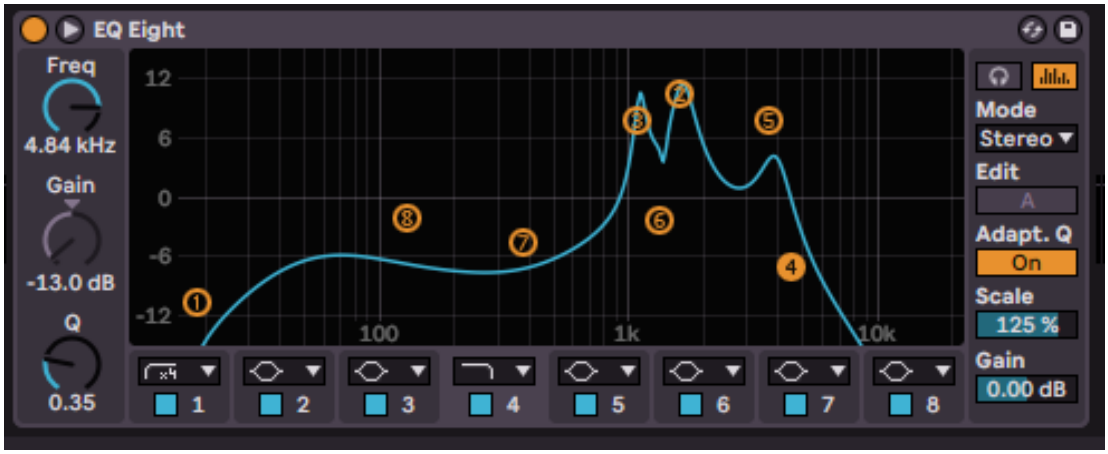
In the third technique, first two techniques were combined. By adding highly amplified signal of the resonant frequencies to the band pass filters, the resonant frequencies were further

enunciated while keeping the smoothness between them.

This (ref. LiveOutput3.wav) sounded fairly similar to the output of the second technique with the breath sound much more enunciated. However, the amplified frequencies were very harsh sounding, so the output of the EQ was lowered. Overall, the output sounded like a blurry breath sound with noise.

Looking closer at the spectral analysis of the reference sample as well as the outputs of the prior attempts of synthesis, some significant differences could be seen. Some crucial elements from the reference, such as a attenuation of high frequencies and a dip in low frequencies, were not reflected in the effect chains.

In the forth attempt in *Live*, these issues were tackled using a complex EQ. By using many nodes in a graphical EQ, the spectrum could be carefully sculpted to match the spectrum of the reference sample. By comparing the two spectra, adjusting the EQ, and repeating until the spectra matched, a intricately shaped EQ was created.



Listening to the output sound of this EQ is disappointing. It is a very dark and hollow sounding gust of wind with no semblance of a breath sound. The failure of this technique could be attributed to applying a frame of the reference to the entire synthesis. Analysing different moments of the reference show small but noticeable changes in the spectrum throughout a breath.

For the sake of experiment, the vocoder in *Live* was used. A vocoder analyzes the frequency content of a sound, decomposes it into several frequency bands, and then uses the amplitude of these bands to modulate a carrier signal. In this case, it analyzes the reference breath sound and applies modulation to white noise. As the vocoder is able to accurately match the reference with dynamic spectral changes, it resynthesized the reference well, the output easily identifyable as a human breath (ref. LiveOutput5.wav).

What was interesting about the vocoder was is accuracy despite low resolutions. Even after setting the bin count (the number of bands the spectrum is separated into) to the minimum of four, the output was still very clearly a human breath (ref. LiveOutput6.wav). The vocoder was able to replicate the reference analyzing and amplifying four frequencies. This was proof that breath synthesis was possible with a dynamic spectrum and at least four frequencies.

Programming

With the lessons learned from experimenting in *Live* and evidence that breath synthesis was feasible, I transitioned to programming in *Csound*, a language designed specifically for sound synthesis. *Csound* allowed for precise numerical control, such as directly using values from the formant table and setting filter parameters to exact numbers.

For the initial prototype of the instrument, I used *CsoundQt*, an open-source frontend for *Csound*.[23] *CsoundQt* is both stable and efficient, making it ideal for building and testing *Csound* instruments.

The first prototype implemented basic additive vocal synthesis with a formant table. The formant table values were assigned to variables:

```
// Formant Table
kcf1 = 1600
kcf2 = 3100
kcf3 = 3950
kcf4 = 5344
kcf5 = 8530
kcf6 = 13400

kbw1 = 200
kbw2 = 300
kbw3 = 200
kbw4 = 500
kbw5 = 1000
kbw6 = 150

ka2 = -6
ka3 = -7
ka4 = -8
ka5 = 3
ka6 = -15
```

[23] Heintz, Joachim, et al. *CsoundQt*, csoundqt.github.io/. Accessed 21 Jan. 2025.

A amplitude envelope was created based on the p3 field (duration input):

```
kamp linseg 0, p3/2, 0.8, p3/2, 0 // amp
```

A white noise generator was then modulated by the amplitude envelope:

```
asig noise kamp, 0 // white noise generator
```

Next, resonance filters were applied for all resonant frequencies in the formant table. The reson opcode allowed for accurate assignment of frequency and bandwidth:

```
// Resonance filters
af1 reson asig, kcf1, kbw1, 1
af2 reson asig, kcf2, kbw2, 1
af3 reson asig, kcf3, kbw3, 1
af4 reson asig, kcf4, kbw4, 1
af5 reson asig, kcf5, kbw5, 1
af6 reson asig, kcf6, kbw6, 1
```

The outputs of the resonance filters were summed into a single audio bus and sent to the output:

```
amix = af1 + af2*ampdb(ka2) + af3*ampdb(ka3) + af4*ampdb(ka4) +
af5*ampdb(ka5) + af6*ampdb(ka6)  // combining reson outputs

out amix // output
```

The Csound instrument was then instantiated to perform a breath with an amplitude envelope length of 0.5 seconds:

```
schedule(1,0.25,0.5) // run instrument
```

Running this code produced a WAV file output (ref. QtOutput1.wav) containing a surprisingly realistic breath sound. Unlike earlier trials in *Live*, this method produced appropriate amounts of high-frequency noise. The breath was characterized by firm mid frequencies and wispy high frequencies, enhancing its realism.

Testing Noise Generators

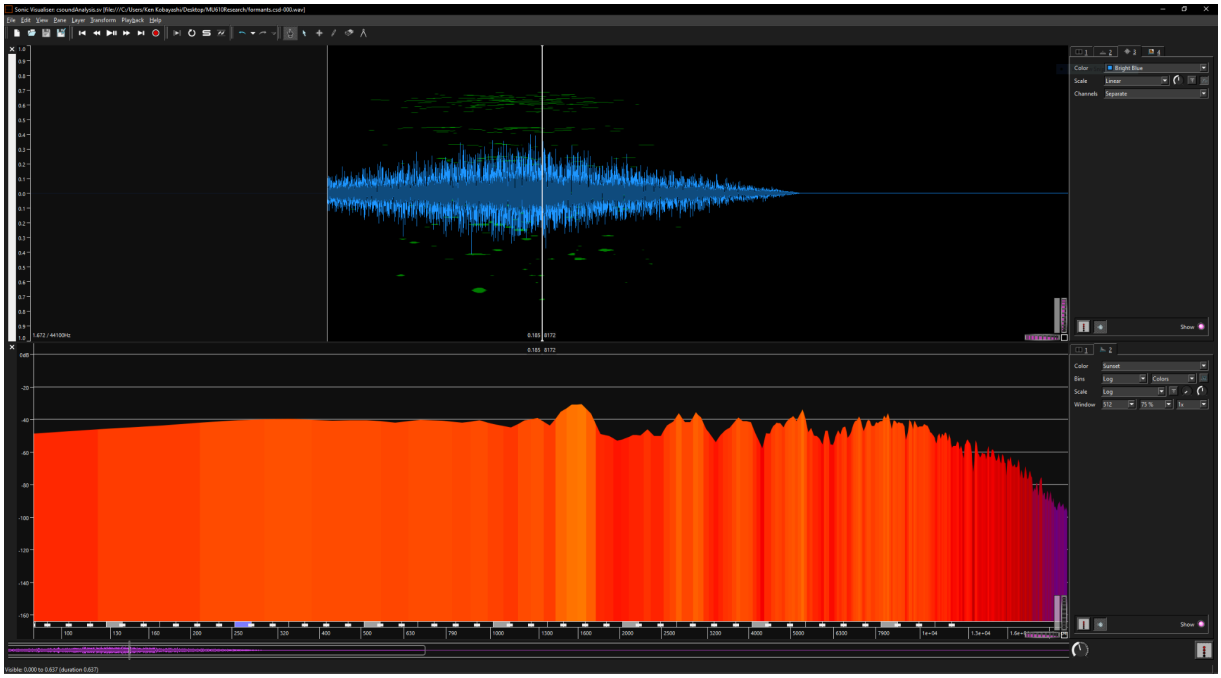
To test the viability of pink noise, the noise opcode was replaced with the pinkish opcode, which generates pink noise:

```
asig pinkish kamp // pink noise generator
```

Listening to the output (ref. QtOutput2.wav) revealed that white noise was superior. Pink noise produced excessive mid frequencies that masked the high-end noise, resulting in a distorted and unpleasant sound. The change was reverted, and white noise was used for the remainder of the project.

Basic Filtering

To more closely investigate potential improvements, the output of the initial Csound script was analyzed in *Sonic Visualiser*.



The spectrogram showed an overall flat frequency curve with clear resonant peaks and gradual high-frequency attenuation. However, the high-frequency attenuation in the output was sharper in the reference sample. To address this, a low-pass filter was added to the signal. Additionally, a high-pass filter was applied to remove sub-bass frequencies below 100 Hz, which were indistinguishable on headphones and unnecessary for the application.

Filters were implemented in series as follows:

```
asig noise kamp, 0 // white noise generator
asig1 butterhp asig, 110 // high-pass filter
asig2 zdf_ladder asig1, 12000, 1 // low-pass filter
```

The filtered output (ref. QtOutput3.wav) had reduced noise and improved clarity. Removing unnecessary frequency content enhanced the breath's realism.

Filter Envelope

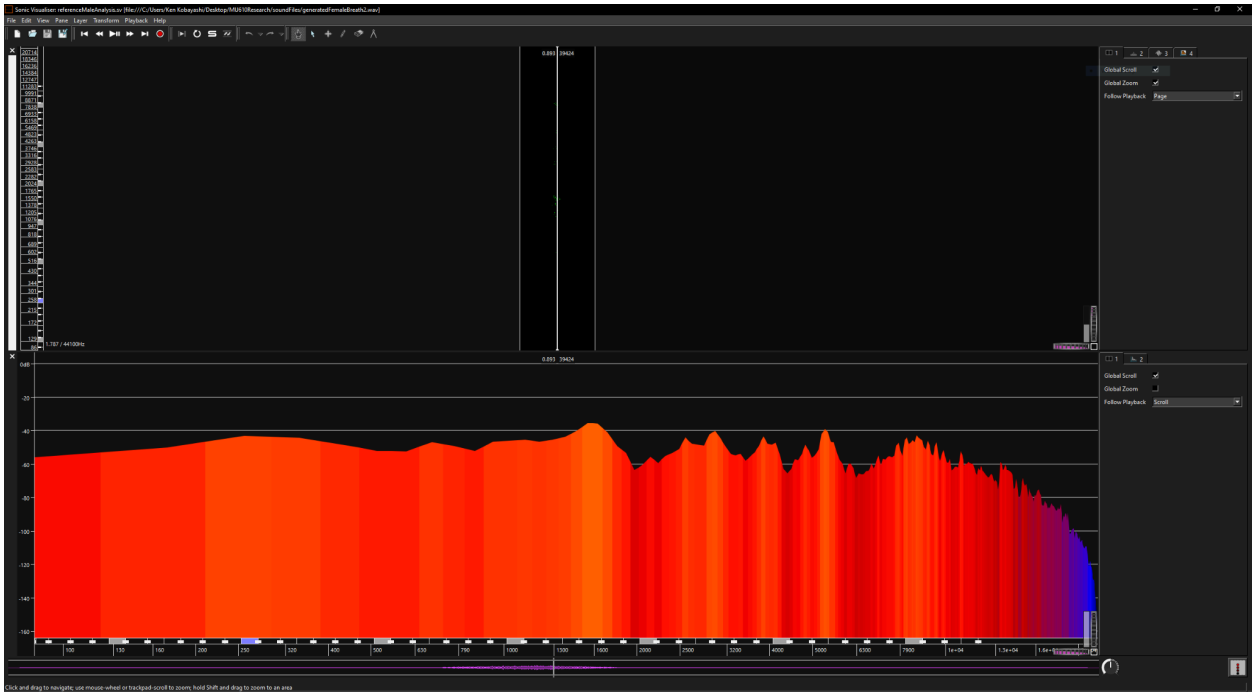
Despite improvements, the output still sounded overly compressed. The static spectrum of the breath caused the dynamic range to suffer, making it difficult to distinguish between inhalation and exhalation. Scrubbing through the reference sample in Sonic Visualiser revealed that the low-pass filter effect varied over time, starting at lower frequencies, moving higher as the breath grew louder, and lowering slightly as it finished.

This phenomenon was replicated in Csound using a filter envelope:

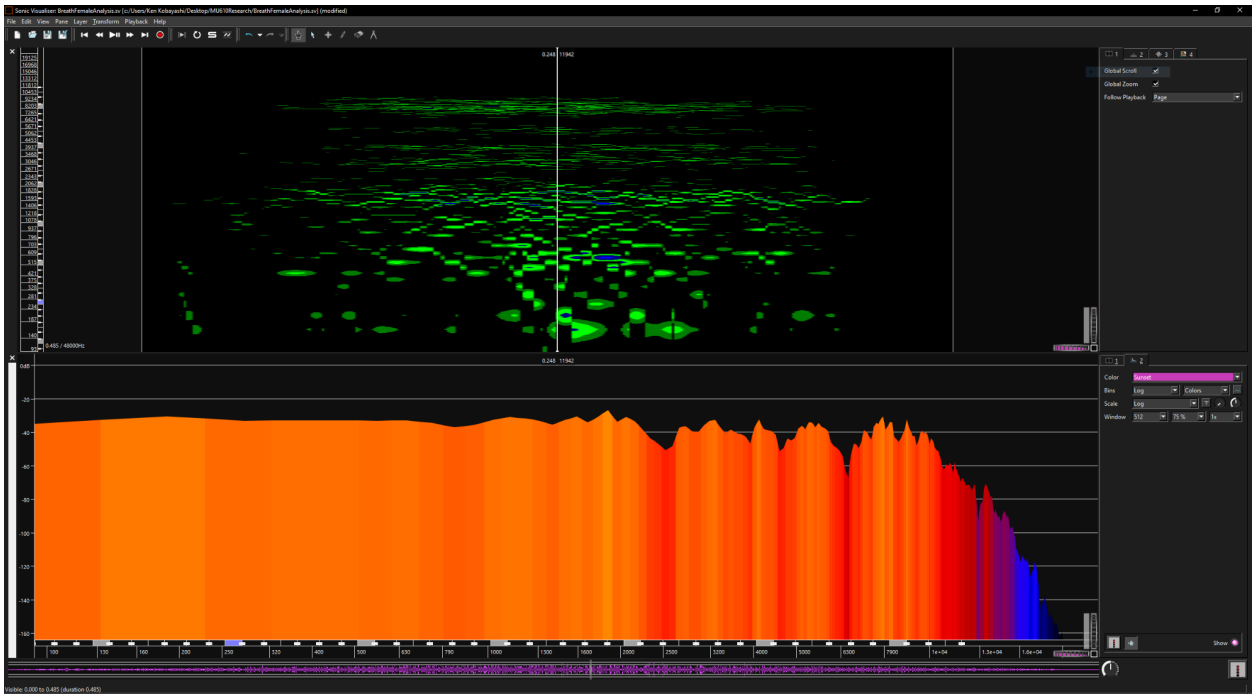
```
kfilt = linseg:k(3000, p3/0.5, 15000, p3, 15000) // filter envelope
amix butterlp amix, kfilt // low-pass filter w/ filter envelope
```

The addition of the filter envelope significantly improved the breath's realism. The sound (ref. QtOutput4.wav) became more nuanced, with a gradual, natural beginning and end. Adjusting the envelope's speed allowed for variations, such as a gentle intake or a sharp gasp.

The accuracy of the output of the Csound code can be observed by analyzing it in *Sonic Visualiser*:



This can be compared to the spectogram output of the reference sample:



The spectrogram showed a flat frequency curve, gradual high-frequency attenuation, and resonant peaks matching the reference in frequency and width. These similarities confirmed the breath sound's realistic synthesis, achieved through precise control and iterative refinement.

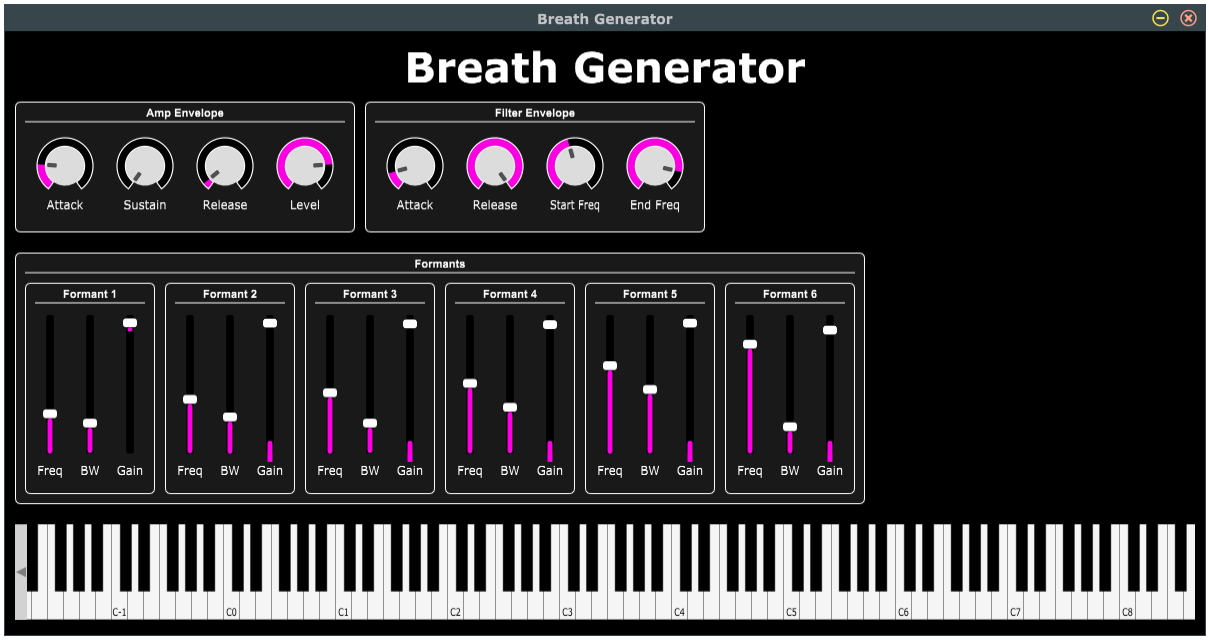
User Interface

With the DSP synthesis completed, a user interface (UI) was needed to enable users to manipulate and customize the breath sounds to their preferences, streamlining the instrument's use in musical applications.

The UI was created using *Cabbage*, a "framework for audio software development" built on *Csound.*[24] *Cabbage* facilitates the creation of VST or AU plugins and provides an intuitive tool for designing graphical user interfaces (GUIs). The Csound instrument developed within *CsoundQt* was imported into *Cabbage,* where the interface development took place.

The initial iteration of the GUI included controls for all the essential functions of the synthesizer:

[24] Walsh, Rory. "Cabbage." *Home | Cabbage Audio*, www.cabbageaudio.com/. Accessed 21 Jan. 2025.



These controls allowed users to adjust the amplitude envelope, filter envelope, and parameters of each of the six formants. This design exposed all synthesizer parameters to the user, enabling them to modify every attribute of the synth via a clear and intuitive interface. A MIDI keyboard was also included at the bottom of the GUI, allowing users to trigger the breath sound by clickin any key.

The functionality of the UI was programmed within the Csound code. Global variables were first initialized for each slider in the UI:

```
// global variables
// amp envelope
gkatk init 0
gkrel init 0
gksus init 0
gklevel init 0

// filter envelope + formants
... // repeated for all other values
```

A new Csound instrument was created to manage the slider values. This instrument was set to "alwayson," ensuring it continuously fetched the UI slider values and assigned them to their respective global variables:

```
// channel input
instr 2
gkatk = chnget:k("atk")
gkrel = chnget:k("rel")
gksus = chnget:k("sus")
gklevel = chnget:k("level")
... // repeated for all other values
endin
alwayson(2) // keeps instrument 2 always running
```

These global variables were then used to replace the input parameters for the opcodes in the main Csound instrument. For instance, the amplitude envelope was implemented as follows:

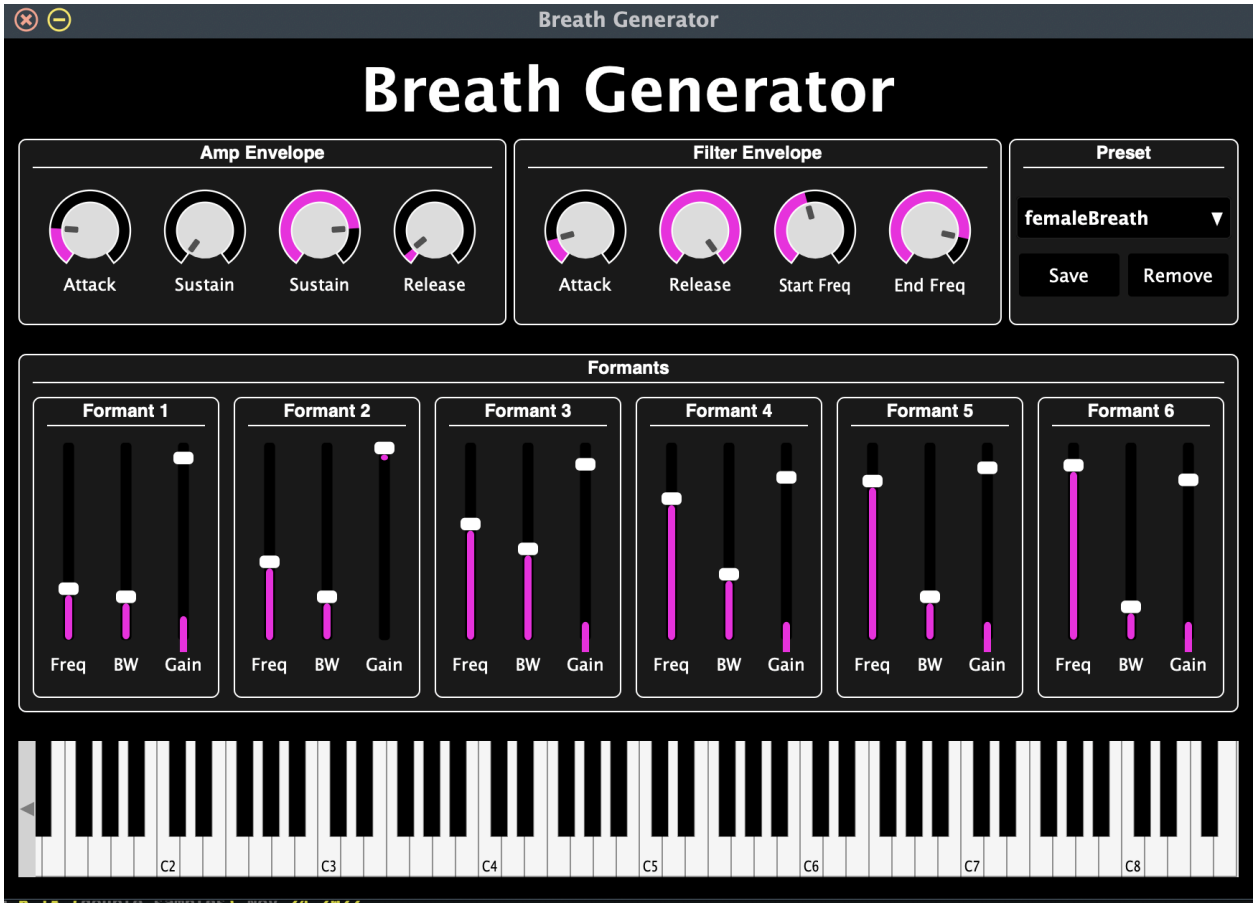
```
// amp envelope variables, casted to i-rate variables to match input
of linseg
iatk = i(gkatk)
irel = i(gkrel)
isus = i(gksus)
ilevel = i(gklevel)

kamp linseg 0, iatk, ilevel, isus, ilevel, irel, 0 // amp
```

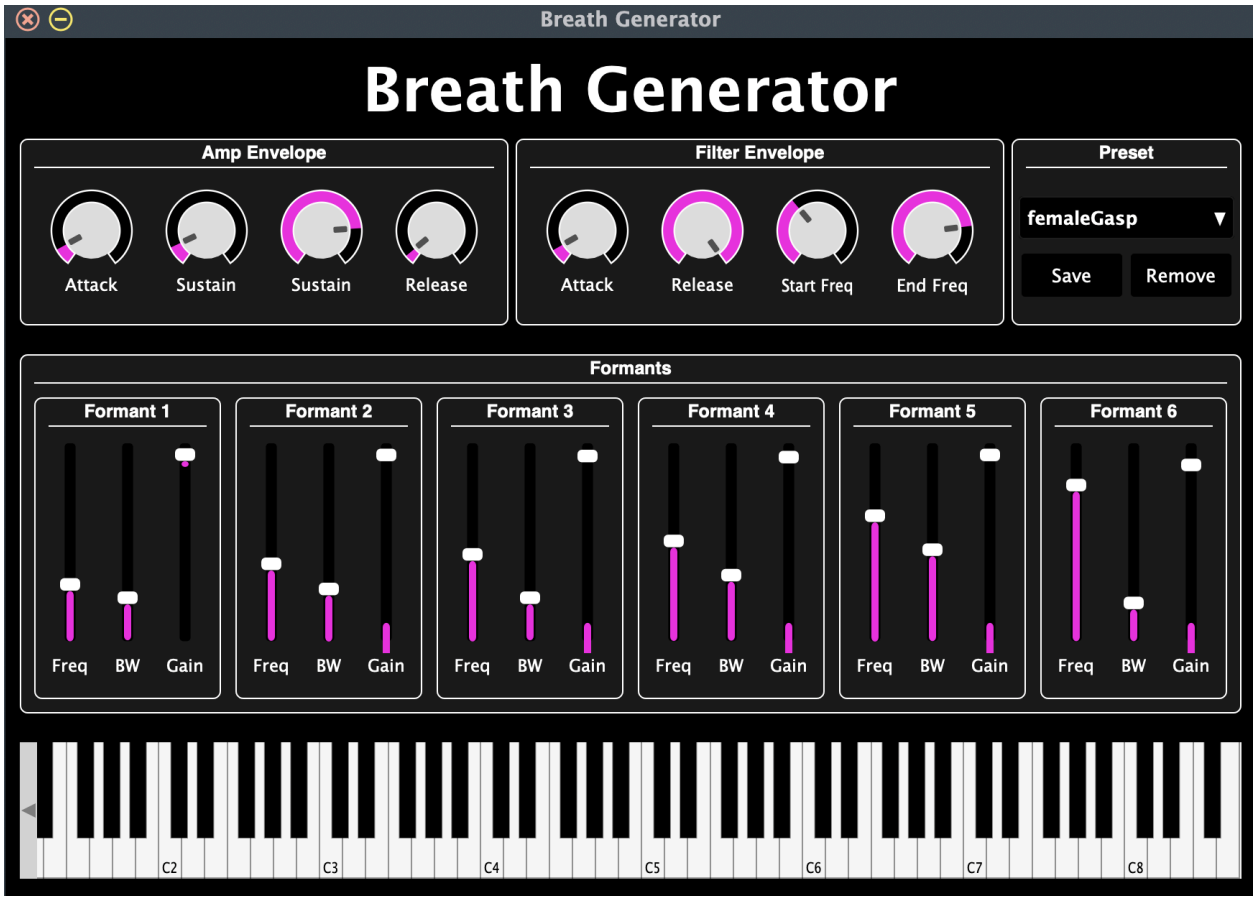
This process was repeated for all slider values, allowing every parameter of the synthesizer to be modified via the GUI.

The extensive customization options necessitated a preset feature, enabling users to save and reload their work. This feature allowed users to quickly access previously crafted breath sounds, significantly enhancing the plugin's practicality.

The preset functionality was implemented using the "filebutton" UI element in *Cabbage*:



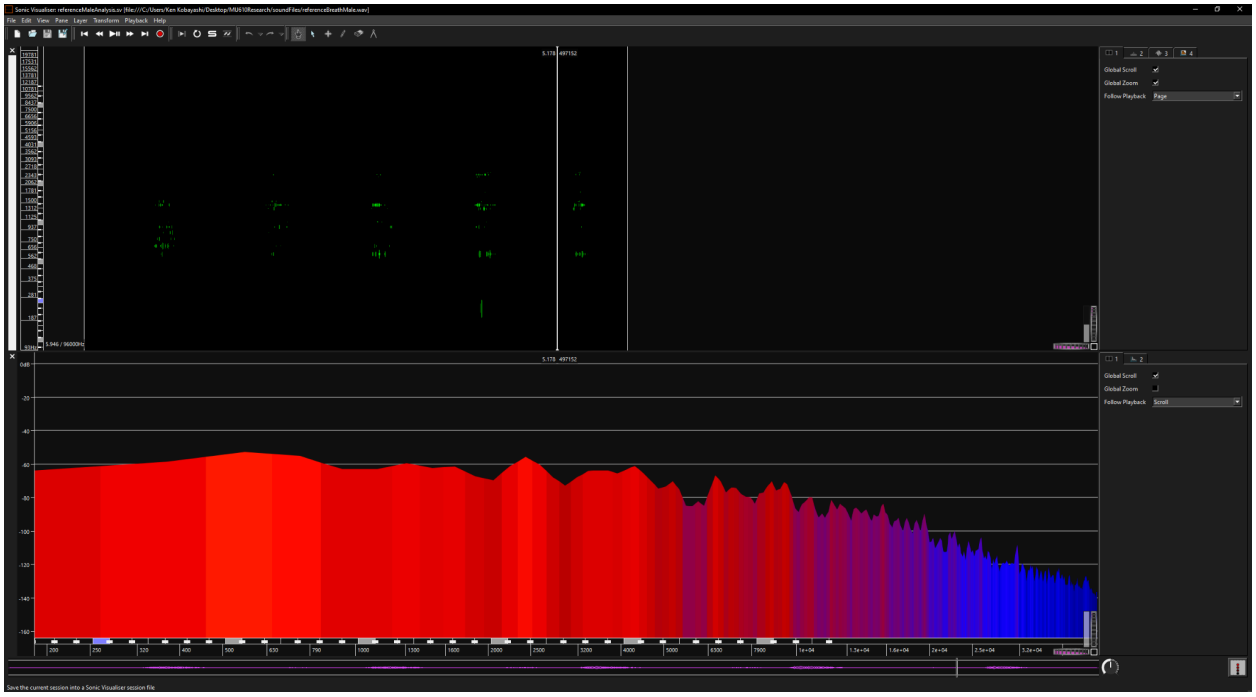
For example, the formant table and envelope settings previously created were saved under the "femaleBreath" preset, which was included with the plugin. Users could then modify these settings to create variations. For instance, by adjusting the amplitude and filter envelopes to be faster, a quick, sharp gasp-like sound was created (ref. generatedFemaleGasp.wav) and saved as the "femaleGasp" preset.



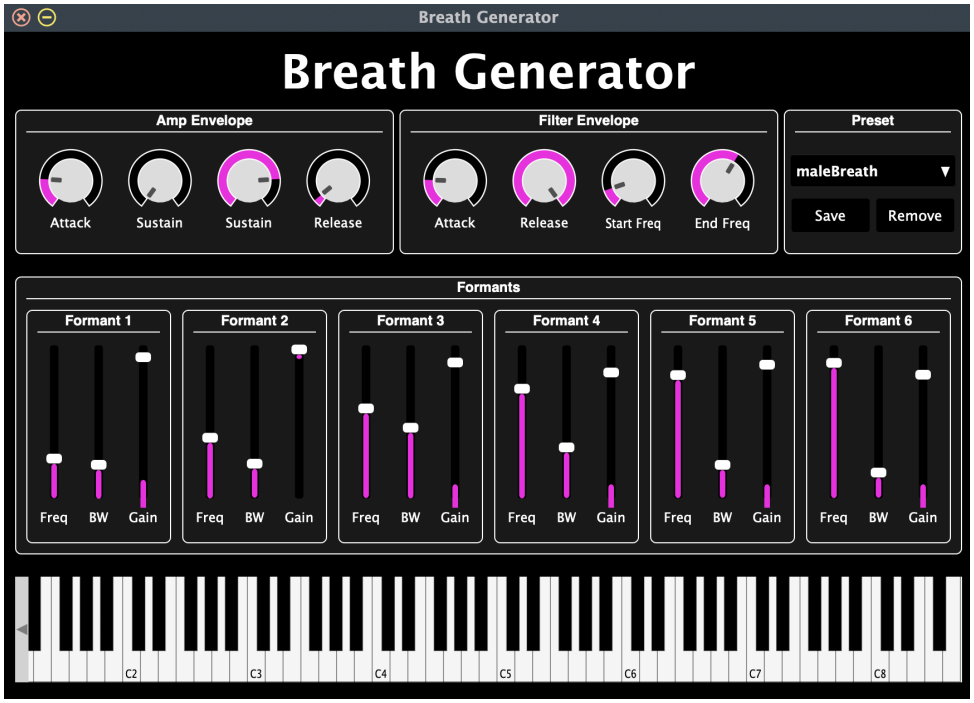
Male Breath

With the GUI of the instrument fully developed, all sound design for the breaths could now be completed within the GUI, eliminating the need for further code modifications to test breath sounds. As a proof of concept, I chose to replicate a male breath within the instrument.

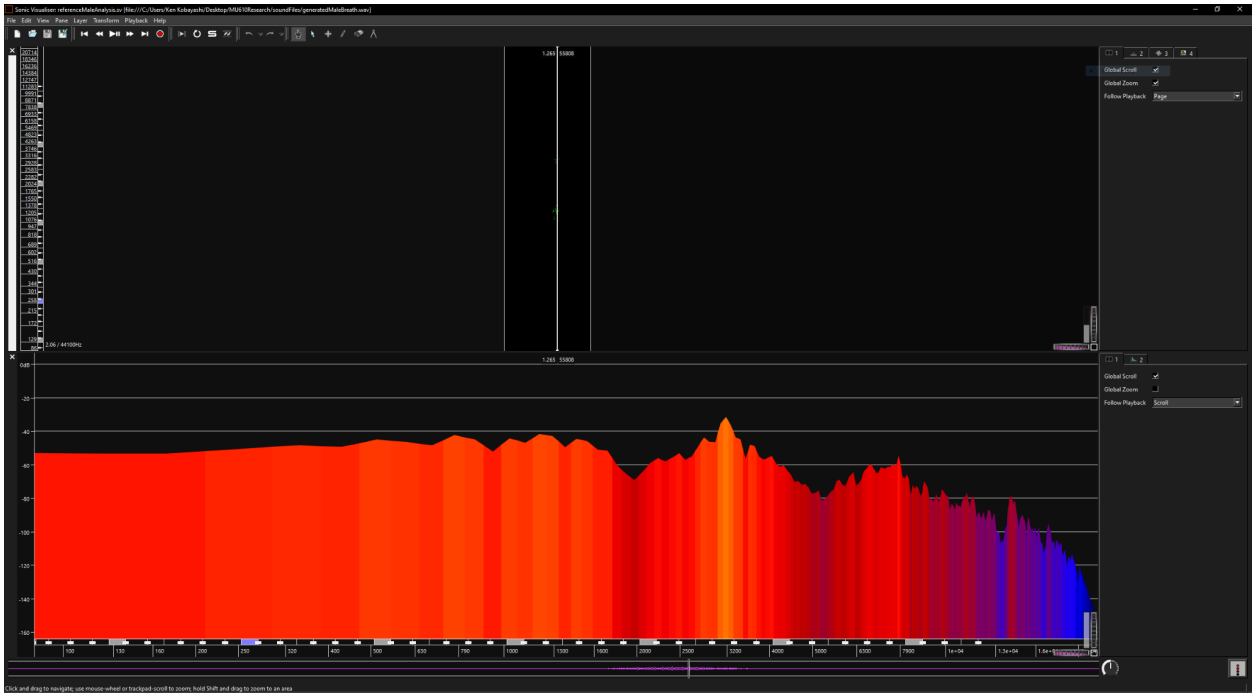
The process of recreating a male breath began similarly to that of a female breath. First, a reference sample was located on *Splice*. The sample "BreathHeavyMale_SFXB_1980.wav" from Blastwave FX was selected. This file was analyzed in Sonic Visualiser to extract spectral data:



The resonant frequencies observed in the spectrogram were input into the instrument through the UI. After making slight adjustments to the formant table values and envelopes, a male breath was successfully recreated in the instrument and saved as a preset, “maleBreath."



This process was quick and straightforward, requiring minimal intervention. Most of the effort involved transferring data from Sonic Visualiser into the instrument. However, the simplicity of this approach resulted in reduced accuracy in the recreation of the breath. A spectral analysis of the generated breath file revealed irregularly large spikes not present in the reference sample:



These irregularities likely contributed to the noise and harshness heard in the output of the preset (generatedMaleBreath.wav).

Conclusion

The synthesis of breath sounds represents a unique intersection of acoustic analysis, digital signal processing, and user-centric instrument design, aimed at bridging a significant gap in vocal synthesis. This project has addressed the often-overlooked dimension of breath sounds in synthetic vocal performance by leveraging detailed formant analysis and additive synthesis.

The challenges of replicating breath sounds were approached methodically, beginning with the spectral analysis of human breath samples, which revealed key resonant frequencies and their roles in defining realistic breath characteristics. The transition from white and pink noise inputs to a meticulously filtered and dynamically modulated output enabled the generation of nuanced breath sounds that closely matched human breath spectra. Furthermore, the implementation of low- and high-pass filters and a dynamic filter envelope enhanced the natural progression of the synthesized breath, adding depth and realism to the final sound.

The use of *Csound* and *Cabbage* as development platforms not only facilitated precise sound design but also enabled the creation of a versatile user interface. This interface empowers users to customize breath sounds through adjustable parameters and save their configurations as presets, streamlining the integration of synthetic breath into musical projects. The incorporation of both male and female breath presets serves as a proof of concept for the flexibility and scalability of this approach, highlighting its potential for further refinement and broader application.

Ultimately, this project underscores the importance of breathing as a critical element in the perception of natural vocal performances. By synthesizing breath sounds that align with human expectations, this work bridges the gap between robotic and human-like qualities in synthetic vocals. As advancements in AI and synthesis technologies continue, the techniques developed here can serve as a foundation for future innovations, enriching the expressiveness and authenticity of vocal synthesis in diverse musical and creative contexts.

Bibliography

"2.2. Formants of Vowels." Phonetics and Phonology, corpus.eduhk.hk/english_pronunciation/index.php/2-2-formants-of-vowels/. Accessed 19 Jan. 2025.

Anonymous. "原音ファイルセット（UTAU用音声ライブラリ）." 歌声合成ツール UTAU, アットウィキ, 11 May 2018, w.atwiki.jp/utau2008/pages/35.html.

Asu. "Asu Op. 2 - Slowly [Original MV]." YouTube, YouTube, www.youtube.com/watch?v=enFLXbbXWdk. Accessed 25 Oct. 2024.

"Ableton Live." Ableton, www.ableton.com/en/live/what-is-live/. Accessed 21 Jan. 2025.

Bell Labs. "Background: Bell Labs Text-to-Speech Synthesis: Then and Now." Bell Labs: Background: Bell Labs Text-to-Speech Synthesis: Then and Now, 7 Apr. 2000, <web.archive.org/web/20000407081031/www.bell-labs.com/news/1997/march/5/2.html> . Accessed 25 Oct. 2024.

cheesum. "Voicebanks 101." Utau.Us, 20 Feb. 2016, utau.us/vb.html.

DECO*27. "DECO*27 - The Vampire Feat. Hatsune Miku." YouTube, YouTube, 9 Mar. 2021, www.youtube.com/watch?v=e1xCOsgWG0M.

Heintz, Joachim, et al. CsoundQt, csoundqt.github.io/. Accessed 21 Jan. 2025.

Hiatt, Brian. "A CHATGPT for Music Is Here. Inside Suno, the Startup Changing Everything." Rolling Stone, Rolling Stone, 22 Mar. 2024, www.rollingstone.com/music/music-features/suno-ai-chatgpt-for-music-1234982307/.

Ibm. "What Is Machine Learning (ML)?" IBM, 17 Oct. 2024, www.ibm.com/topics/machine-learning.

Julia, and Eddy. AI Hub Docs, AI Hub, 21 Oct. 2024, docs.aihub.gg/.

Kenmochi, Hideki. "Singing Synthesis System VOCALOID and Hatsune Miku." Research on Intellectual Property Rights on Digital Content, Mar. 2008, pp. 38–42, http://www.dcaj.or.jp/project/report/pdf/2007/dc08_03.pdf#page=38.

Kenmochi, Hideki, and Hayato Oshita. "Vocaloid – Commercial Singing Synthesizer Based on Sample Concatenation," homes.esat.kuleuven.be/~spch/www.interspeech2007.org/Technical/ssc_files/Yamaha/VOCALOID_Interspeech.pdf. Accessed 25 Oct. 2024.

"Microsoft Research." AI Music, Microsoft Research, 9 Sept. 2022, www.microsoft.com/en-us/research/project/ai-music/.

Martin, Sean Thomas. "Formants Explained and Demonstrated." YouTube, 2019, youtu.be/jpbFnsusfp0?si=GTaCHP28B9kq0fyY.

"Spectral Warping Wavetable Synth." Vital, vital.audio/. Accessed 20 Jan. 2025.

"Effective Prompts for AI: The Essentials." MIT Sloan Teaching & Learning Technologies, MIT, 16 Sept. 2024, mitsloanedtech.mit.edu/ai/basics/effective-prompts/.

O'Dell, Cary. "Daisy Bell (Bicycle Built for Two)." "Daisy Bell (Bicycle Built for Two)"—Max Mathews, John L. Kelly, Jr., and Carol Lochbaum (1961), www.loc.gov/static/programs/national-recording-preservation-board/documents/DaisyBell.pdf. Accessed 25 Oct. 2024.

"AI Hub Discord." AI Hub Discord, Discord, discord.com/invite/aihub. Accessed 15 Jan. 2025.

Formants. Acoustic Phonetics: Formants, University of Manitoba, home.cc.umanitoba.ca/~krussll/phonetics/acoustic/formants.html. Accessed 19 Jan. 2025.

"Billboard Japan Hot 100: Charts." Billboard JAPAN, Billboard, www.billboard-japan.com/charts/detail?a=hot100. Accessed 15 Jan. 2025.

Suno. Suno.com. Accessed 25 Oct. 2024.

Walsh, Rory. "Cabbage." Home | Cabbage Audio, www.cabbageaudio.com/. Accessed 21 Jan. 2025.