ECE 45 Project

Professor Massimo Franceschetti, Winter 2024

A special thanks to our TAs, Sandeep Chintada and And Yilmaz.

ABSTRACT

The **Audio Synthesizer** is a versatile tool designed for generating and manipulating audio signals in real time. It was designed by our team using **MATLAB**, including MATLAB's GUI (Graphical User Interface) and functions via MATLAB's library. It offers a range of features to customize sound characteristics such as oscillator type, amplitude, noise level, envelope shaping, pitch modulation, input sequence, tone delay, low-pass filtering, echo effects, and graphical representation of various parameters. For adjusting the values, we have added sliders and as well as text boxes that allow the users to choose their favored way to set values. The Audio Synthesizer provides a wide range of functionality, and can be used for music production, education, experimental sound design, game development, and much more:

- **Music Production**
    - Create custom sounds, melodies, and effects for music composition, sound design, and audio production.
- **Education**
    - Demonstrate principles of sound synthesis, envelope shaping, and waveform manipulation in educational settings, encouraging learning in music technology, signal processing, and digital audio.
- **Experimental Sound Design**
    - Explore unconventional sound textures, modulations, and tonalities for experimental music and audio projects.
- **Game Development**
    - Generate dynamic sound effects, background music, and interactive audio elements for video games, enhancing the gaming experience with tailored audio content.

The Audio Synthesizer offers a user-friendly interface and a comprehensive set of features. It is a great tool for creativity and exploration in sound synthesis and digital audio processing.

APPLICATION

Getting Started

1. Use MATLAB (or MATLAB Online) to run the installer `ECE_45_Synthesizer_Project.mlappinstall`.
2. Once the app is installed, go to the *Apps* tab to run it.
3. Install the *DSP System Toolbox* dependency as prompted.

Features

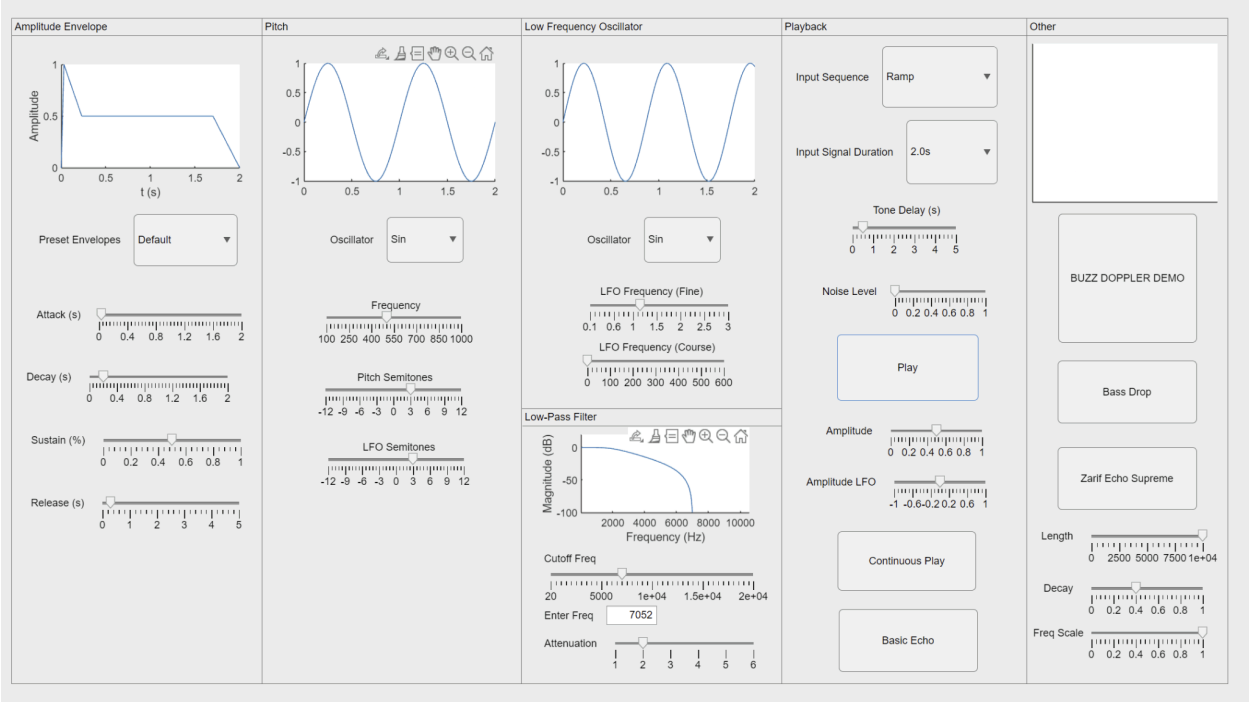

Fig. 1: Application Interface

Tooltips

Many of the buttons and sliders have mouse-over tooltips. Try it out!

Oscillator Selection

The synthesizer offers four types of oscillators: Sine, Square, Sawtooth, and Triangle. Each oscillator produces a distinct waveform, allowing users to achieve different tonal qualities.

Amplitude and Noise Control

Amplitude adjusts the overall volume, and noise control introduces noise into the audio signal. The amplitude slider scales from 0 to 1, controlling the intensity of the sound, while the noise control slider adds a controlled amount of noise to the signal, enhancing the texture of the audio.

Amplitude Envelope

The Amplitude Envelope feature shapes the volume outline of the generated sound. Users can choose from available preset envelopes or customize the envelope's parameters — attack, decay, sustain, and release, using the sliders.

- **Preset Envelopes**
  - Choose from pre-made envelope shapes, including:
    - *Default*
    - *Short*
    - *Impending Doom*
- **Envelope Parameters**
  - *Attack* (s)
    - Controls the time taken for the sound to reach its maximum volume.
  - *Decay* (s)
    - Specifies the time taken for the sound to decrease from its peak to sustain level.
  - *Sustain* (%)
    - Determines the sustained volume level as a percentage of the maximum volume.
  - *Release* (s)
    - Sets the time taken for the sound to fade out after the key is released.

Pitch Control

The Pitch tab allows users to manipulate the frequency and pitch of the audio signal using sliders.

- **Frequency**
  - Adjust the frequency of the sound wave from 100 to 1,000 Hz.
- **Pitch Semitones**
  - Alter the pitch in semitone increments from -12 to 12, enabling precise tuning and transposition of notes.

LFO (Low-Frequency Oscillator)
Pitch modulation using the LFO features.

- **LFO Semitones**
  - Adjust pitch modulation in semitone increments from -12 to 12, adding dynamic pitch variation to the sound.
- **LFO Frequency**
  - Control the frequency of the LFO from 0.1 to 300, determining the speed of the pitch modulation.
- **Amplitude LFO**
  - Adjust the amplitude of the LFO from -1 to 1, controlling the depth of pitch modulation.

Low-Pass Filter
Apply a low-pass filter to the audio signal to attenuate frequencies above a specified cutoff frequency. Adjust the maximum frequency and attenuation parameters to tailor the filter's effect.

- **Cutoff Frequency**
  - Specify the cutoff frequency of the low-pass filter. Frequencies above this value are attenuated according to the filter's characteristics.
- **Attenuation**
  - Determine the amount of reduction in amplitude for frequencies above the cutoff frequency. Higher attenuation values result in a more significant reduction of higher frequencies, effectively smoothing the audio signal.

Echo Effect

- **Basic Echo**
  - When the button is pressed, add a delay to the audio with half the amplitude, creating a simple echo effect.
- **Zarif Echo Supreme**
  - Made by our dedicated member Zarif. Adds echo to the audio with additional parameters.
    - *Length*: Adjust the length of the echo from 0 to 4 seconds (scaled from 0 to 10000).
    - *Decay*: Control the decay of the echo effect, ranging from 0 to 1.
    - *Frequency Scale*: Scale the frequency of the echo, affecting its pitch, ranging from 0 to 1.

Input Sequence and Duration
Select the input sequence type from available options, including "Ramp," "Long," "Item 3," and "Loud." Specify the duration of the input signal, choosing from 0.5s, 1.0s, 1.5s, and

2.0s. This feature adds versatility to the synthesizer as it allows users to create dynamic audio sequences with varying durations and patterns.

Tone Delay
Introduce a delay between each played tone in seconds, ranging from 0 to 5 seconds using sliders. The tone delay feature adds rhythmic variation and spatial depth to the synthesized audio.

Playback
Initiate playback of the adjusted audio signal with a “Play” button, playing the modified audio sequence a specified number of times, with each iteration at a higher pitch depending on the sequence chosen. Enable looped playback with the “Continuous Play” toggle button. When activated, it will loop the sequence continuously. Clicking “Continous Play” again will stop the looped playback. Most features update the text tone during continuous playback.

Bass Drop
Adds a bass tone to each note that plays, affected by the tone duration and base amplitude.

BUZZ DOPPLER DEMO
Demonstrates a doppler shift effect. Not affected by other settings. Requires Picture1.jpg in project folder.

CONCEPTS

Since sound waves are just signals, they can be manipulated through signal processing. MATLAB’s `audio()` function takes an array representing a digital signal — i.e., a signal composed of discrete samples (this is because, as we learned in class, computers can only process digital signals, which are then converted to analog signals to be played). Our samples are represented by our discrete time vector $t = [0, \frac{1}{\omega}, \frac{2}{\omega}, ..., d]$, where `d` is the duration of the sound.

- We used the function $x(t) = sin(2\pi ct)$ (High Frequency Oscillator) to construct the input signal. This is a digital signal since `t` is only non-zero at $\frac{n}{\omega}$.

- We constructed impulse response functions `h(t)` and convolved them with the input (i.e., HFO) signal to implement the filters. The “Zarif Echo Supreme” filter, for example, was essentially just $y(t) = x(t) * (\delta(t) + A\delta(t - t_0))$, where the impulse

response $h(t) = \delta(t) + A\delta(t - t_0)$. Of course, this is equivalent to passing the input signal `x(t)` through the LTI system `h(t)` to get output signal `y(t)`.

- We implemented a low-pass filter for our signal to cut off higher frequencies using MATLAB's `butter()` function.

- Our LFO (Low Frequency Oscillator) modulated the HFO-generated input signal using either Sine, Sawtooth, or Square waves. For the Sine LFO, for example, this meant multiplying the input signal `x(t)` by the LFO signal $o(t) = sin(2\pi\omega_0 t)$.

CONTRIBUTORS

Genaro Salazar Ruiz (A18059912) — Acted as project manager and captain in coordinating roles and positions for the group. Provided code for the UI and echo, Doppler, and oscillation functions. Adjusted code when needed for less audible distortion and more clarity. Worked with Nathan and other members on implementing all the helper functions.

Nathan Davis (A18070169) — Project founder and co-captain alongside Genaro. Coordinated project meeting times and role preferences. Worked on overall synthesizer design, GUI layout, and implementation.

Jeremy Potter (A17532711) — Revised, edited, and formatted the entire report and wrote the "Concepts" and (with Preston) the "Getting Started" and "Works Cited" sections. Worked with various members to help test the finished app.

Preston Long Le (A18100872) — Designed and edited the low pass filter and frequency and pitch amplifier. Edited and implemented various functions with Nathan and Genaro. Worked with Jeremy on the report, including the "Works Cited" section.

Eric Truong (A18104696) — Worked on the LFO, Envelope, Doppler, Alarm, and Impending Doom functions, and the options to change the frequency and amplitude, to graph and play the sound, and to switch the waveform. Worked with Genaro and Nathan on implementation and debugging.

Nick Ji (A18060321) — Provided the basic skeleton code for the audio synthesizer, including the noise generating function and the Sine, Square, Sawtooth, and Triangle oscillator waveforms. Wrote and edited the "Abstract" and "Application" sections of the report.

Pussakorn Chanpanichravee (A18081958) — Created general synthesizer 1 and 2, allowing users to create and edit sounds by adjusting waveforms to reduce or increase noise, amplitude, and frequency. Worked with Nathan to implement continuous looping of input sequences with varying pitch and tone. Worked with other team members to implement various synthesizer functions.

Mustahsin Zarif (A17499159) — Researched MATLAB functionality. Provided code and merging advice for the echo feature and convolution. Helped Nathan with scaling the GUI plots to fit in visible oscillations by using the linespace function.

Hussain Alramadhan (A18102883) — Acted as moral support and project overseer, providing feedback to members during the trial stage of their code. Worked with Genaro and Nathan to implement the GUI, Doppler function, and LFO.

WORKS CITED

Cook, Perry R. and Gary P. Scavone, *The Synthesis Toolkit in C++ (STK)*. The Stanford Center for Computer Research in Music and Acoustics, ccrma.stanford.edu/software/stk/. Accessed 24 March 2024.

Learning Synths. Ableton, learningsynths.ableton.com. Accessed 24 March 2024.

"MATLAB colon." *MATLAB & Simulink Documentation*, R2024a, MathWorks, www.mathworks.com/help/matlab/ref/colon.html. Accessed 24 March 2024.

"MATLAB `conv`." *MATLAB & Simulink Documentation*, R2024a, MathWorks, www.mathworks.com/help/fixedpoint/ref/conv.html. Accessed 24 March 2024.

Musicdsp.Org. www.musicdsp.org/en/latest/. Accessed 24 March 2024.

"Practical Introduction to Digital Filter Design." *MATLAB & Simulink Documentation*, R2024a, MathWorks, www.mathworks.com/help/signal/ug/practical-introduction-to-digital-filter-design.html. Accessed 24 March 2024.

"Signals and Systems/Filter Transforms." *Wikibooks*, en.wikibooks.org/wiki/Signals_and_Systems/Filter_Transforms. Accessed 24 March 2024.

Fig. 2: That’s all, folks. Thanks to whoever is grading this 😝