ECE45: Project Guidelines and Instructions for Fall 2021

Prof. Massimo Franceschetti
Department of Electrical and Computer Engineering
University of California San Diego, CA 92093
Email: massimo@ece.ucsd.edu

***Abstract*— In this project, you will be building an audio synthesizer using some of the concepts learnt in class. The goal is to give you some hands-on experience to familiarize you with a small subset of the numerous applications of signal processing and systems theory.**

I. INTRODUCTION

The word *synthesizer* refers to any device or instrument that produces audio signals. Examples include battery alarm software used in computers, noise-generators used in children's toys, and musical instruments usually controlled by keyboards. Synthesizers convert mechanical or electrical energy into sound energy and often use controls that enable the user to adjust the properties of the generated sound.

II. EXAMPLES OF SYNTHESIZER CONTROLS

The controls provided by a synthesizer depend on the kind of applications the synthesizer is designed for. While some synthesizer controls are used adjust the most basic physical properties of the generated sound output, namely, the *amplitude* (which determines the loudness or intensity) and the *pitch* (the audio frequency, which determines how shrill the generated sound is), others serve more sophisticated purposes such as changing the resonant frequency of the low-pass filter through which the input signal is passed. We provide two examples of synthesizer controls below.

1) The simplest example is the volume slider that appears on your computer screen. It is a one-dimensional synthesizer control.
2) The amplitude-pitch control [1] is a 2-dimensional control in which the x and the y axes plot the pitch and the amplitude, respectively.

An important class of controls, called *envelopes*, is discussed next.

III. ENVELOPES

[2] defines envelopes as "a table of data points that is output over a specified period of time". One may think of an envelope as an x-y plot that shows how any given attribute of the output signal (such as pitch) should evolve over time. Envelopes enable a synthesizer to change sound characteristics automatically over time, unlike sliders that need to be varied manually.

Typically, the timeline of an envelope can be partitioned into four stages: *attack*, *decay*, *sustain*, and *release*.

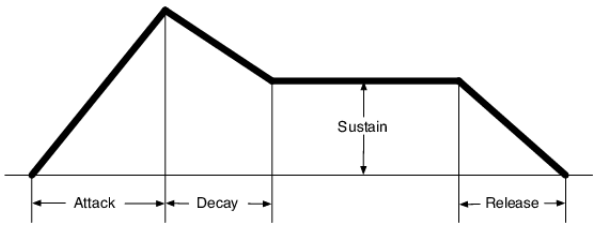

Fig. 1: Stages of an ADSR Envelope [3]

1) In the attack stage, the synthesizer tunes up the desired sound attribute (the quantity plotted along the y-axis, say amplitude) to a peak value.
2) In the decay stage, the quantity is made to decrease until it hits a certain user-defined value.
3) The sustain stage simply sustains the amplitude at the user-defined value resulting from the decay stage.
4) The release stage lets the quantity fade out until its value reaches 0.

These stages are illustrated in Fig. 1. Note that the plot (i.e., the envelope) need not be piece-wise linear in general. Envelopes that exhibit these stages are called *ADSR envelopes*.

IV. AUDIO OSCILLATORS

Audio oscillators are instruments that produce mechanical oscillations to generate all of the sound that a synthesizer subsequently processes and outputs. Most audio oscillators generate signals that are either sine waveforms, saw-tooth waveforms, or square waveforms. Certain synthesizer provide controls to vary the *pulse-widths* of these waveforms.

V. INSTRUCTIONS

Pick a programming language of your choice and write a code to design your own synthesizer. You are encouraged to collaborate with other students, because synthesizers with more functions and features will lead to better grades. Therefore, you are encouraged to merge the features that you developed with the ones of other groups to enhance the capabilities of your synthesizer. One way to start is for example to pick an oscillator waveform and add to it a linear filter with adjustable parameters to obtain different sounds. You can also add noise at different frequencies, have controls to change the pitch, add different waveforms, and process them in many ways. You can also use different periodic

signals to modulate the parameters of your synthesizer's features using so-called LFO's (see below). The resources indicated below will give you useful guidelines and examples for reference. You should also work on the presentation of your synthesizer using a simple and clear interface.

In addition, be sure to do the following while writing your project report.

1) Discuss the potential applications of your synthesizer. This is important because an ambulance siren, for example, might need a different range of frequencies when compared to a fire alarm system.
2) Describe in detail all of the synthesizer controls and what they mean. Label all the axes and mark all the value ranges (if any) appropriately.
3) In the process of designing your synthesizer, you must apply at least one of the concepts learned in class. Clearly explain how you applied the chosen concepts in your report.
4) Cite all of the codes and functions (or predefined modules) that you borrow from other programmers across the web.

VI. Additional Resources

Here are some additional resources to help you get started. However, being free resources, they are not guaranteed to be error-free.

1) The website https://learningsynths.ableton.com provides an excellent interactive introduction to synthesizers and explains concepts that have not been discussed here. If you want to skip this tutorial and find all of the synthesizer controls in one place, visit https://learningsynths.ableton.com/en/playground.
2) https://blog.demofox.org/diy-synthesizer/ provides a coding tutorial to build a do-it-yourself (DIY) synthesizer.
3) Musicdsp.org is a "collection of algorithms, thoughts and snippets, gathered for the music dsp community."
4) http://portaudio.com/ is a free and open-source audio library that provides signals you can experiment with.
5) The Synthesis Toolkit (STK, available at https://ccrma.stanford.edu/software/stk/) is a repository of open-source C++ programs used for processing audio signals and developing algorithms for synthesizing music pieces. The repository is also available on GitHub.
6) For a more advanced development you can check out the free version of the graphical interface MAX to develop software here: https://cycling74.com/products/max, which is based on the pure data real-time music and multimedia environment MSP developed at UCSD: http://msp.ucsd.edu/software.html

This is just the tip of the iceberg; there are a lot more resources available online.

References

[1] https://learningsynths.ableton.com/en/making-changes/play-with-amplitude-and-pitch.

[2] https://theproaudiofiles.com/synthesis-101-envelope-parameters-uses.

[3] R. Rowe. Instruments and the electronic age: Toward a terminology for a unified description of playing technique, 1990.