Virtual Synthesizer Development in C++

Will Sieber

Contents

1 Introduction

1.1 Project Goals

2 Synthesizer Basics

2.1 Components

2.1.1 Oscillators

Oscillators can be generally broken into two categories: tone generators and controllers. Tone generators repeat periodic waves in order to generate sound. Simple tone generators can be expressed as mathematical expressions, such as $f(x) = sin(x)$, while complex tone generators allow for any waveform to be repeated, regardless if they can be expressed as a single expression. Complex oscillators generally allow for a number of different waveforms to be represented in a single table, allowing for a user to select any number of waveforms in real time. Controller oscillators, on the other hand, are much lower frequency than tone oscillators, thus have the name *Low-Frequency Oscillators* (LFOs). LFOs are generally used as an input to automate other parameters of a synthesizer, allowing for a greater range of timbres. For the purposes of this project, simple oscillators are used as tone generators.

2.1.2 Envelope Generators

In order to provide articulation, envelope generators describe how the amplitude of a particular sound should change over time. Several parameters are used to accomplish this goal, and those are the synthesizer's *attack*, *decay*, *sustain*, and *release*. *Attack* describes how much time it takes for the synthesizer to reach full amplitude upon receiving the signal to generate sound. A short attack will reach full amplitude quickly, such as pressing a key on a piano or organ. Longer attacks will take more time to reach full amplitude, which create a "swelling" effect that is similar to that of a violin increasing in volume. *Decay* describes how much time the synthesizer should take to reach a particular amplitude once the attack is finished. A longer decay will prolong a sound prior to the sustain, while a shorter will reach it quicker [word better]. *Sustain*, unlike attack and decay, describes the amplitude itself to which the

decay will reach upon completion. When reached, the synthesizer will continuously play at this amplitude until a signal is sent that it should stop generating sound. Upon this signal, *release* describes how long the synthesizer should take to stop generating sound by gradually decreasing the amplitude to zero. In more technical terms [].

One may notice a problem with this model - the rate at which the attack, decay, and release change amplitude itself is not described. Powerful synthesizers allow for this change to be represented as a polynomial function, but for the purposes of this project, the rate at which each parameter changes will simply be linear.

With these two components in mind, one can create a simple synthesizer in the same manner that is described in this project.