Synthotron

Live Music Performance with Android

18-551 Spring 2014 Capstone Project
Michael Nye (mnye), Michael Ryan (mer1)

1 The Problem

Music performance applications on Android are virtually non-existent. Part of this problem is that historically, Android has not had good audio or touch latency, and music applications require very low latency to feel responsive. Modern Android versions have latency on the order of 100-150 ms. While this isn't yet low enough for a traditional keyboard-like interface, it is low enough to allow for performance aspects with an appropriate interface, with CD-quality audio (44.1kHz, 16-bit samples) in real-time.

2 Algorithms

2.1 Oscillators

The most basic element of synthesis is the oscillator. An oscillator is a device that simply emits a certain waveform at an adjustable frequency. Our oscillators are designed to emit the set of fundamental waveforms: sine waves, saw waves, square waves, triangle waves, and pulse waves. These waveforms are desirable for music applications because they have rich upper harmonics, which gives lots of flexibility to carve different tonalities out of them.

Implementing these waveforms in the naive way (i.e. with sharp edges at transition points) implicitly decimates a waveform with arbitrarily high frequency content. The result is a waveform that has large amounts of aliasing; this is especially evident when playing high frequency notes.

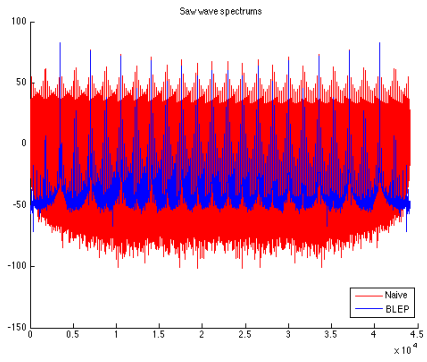

Figure 1: Frequency content of saw wave

In order to avoid this problem, our oscillators use a method called **Polynomial Bandlimited Steps**, or PolyBLEP[6]. PolyBLEP works by rounding the edges around a sharp transition point using a polynomial. If `t` represents the phase of our waveform normalized between 0 and 1, and `dt` represents the change in our phase each sample, then the PolyBLEP term can be computed as follows:

```
if (t < dt) {
    t /= dt;
    return t+t - t*t - 1.0;
} else if (t > 1.0 - dt) {
```

```
    t = (t - 1.0) / dt;
    return t*t + t+t + 1.0;
} else {
    return 0.0;
}
```

This method is incredibly effective, and can pull down the peak sidelobe aliasing my 50 to 100 decibels for a saw wave. See Figure 1 for a comparison of the frequency content of a naive saw wave, and a PolyBLEP saw wave.

2.2 Low Frequency Oscillators

A typical oscillator is used to generate an audible waveform. Low Frequency Oscillators, or LFOs, are oscillators at sub-audible frequencies that are instead used to modify parameters of filters or other oscillators. Our application uses LFOs to create two different effects: vibrato, and tremolo.

Vibrato uses an LFO to modify the frequency a waveform oscillates at. For example, we can use the following equation to compute the instantaneous frequency of an oscillator from its stationary frequency as $f_i = f_s \cdot 2^{LFO(t) \cdot I}$, where $LFO(t)$ represents the LFO waveform, and I controls the depth of the vibrato, where 0 represents no vibrato, and 1 represents an octave.

Tremolo uses an LFO to modify the gain of an input signal to create a wavering sound. The tremolo gain can be computed as $10^{LFO(t) \cdot I}$, where $LFO(t)$ represents the LFO waveform, and I controls the depth of the tremolo.

2.3 ADSR Envelopes

ADSR Envelopes (short for Attack, Decay, Sustain, Release) are used to shape a note when it is trigged. A note passes through four states: the attack, decay, sustain, and release state. Attack, decay and release occur for a fixed increment of time, while the sustain state can last indefinitely.

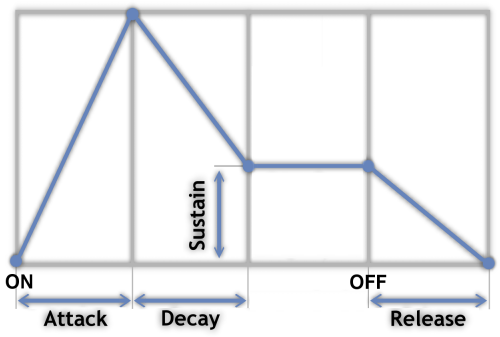

Figure 2: ADSR Envelopes

In the attack state, the gain of the note scales from 0 to 1; this gives the note a fade in. In the decay state, the gain scales back from 1 to the sustain level; this backs the note off and makes the attack sound sharper. During the sustain state, the gain remains constant until the note is told to stop. It finally enters the release state, where the gain scales from the sustain level back to zero.

2.4 Compressors and Limiters

Compressors and limiters are a class of nonlinear processors that respond to the signal level of its inputs. The devices first rely on an envelope estimation, that determines the amplitude of the signal.

In order to estimate the envelope of a signal, we rely on leaky integration of our signal. This takes the from $env(t) = \alpha \cdot env(t-1) + (1-\alpha) \cdot x(t)$. However, in order to rise quickly and fall slowly, we use to different values of α. A small value of α is used if $x(t) > env(t-1)$, and a larger value is used if $x(t) < env(t-1)$. A sample envelope estimation is show in Figure 3.

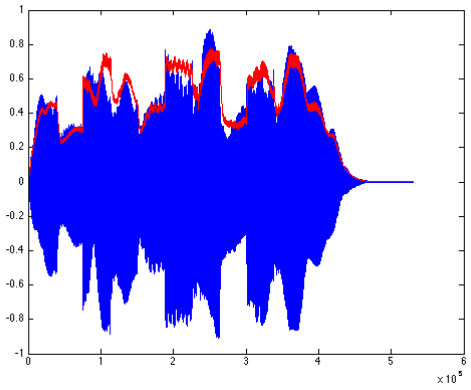

Figure 3: Envelopes Fitting

Once we have an envelope function, a limiter and compressor can use to calculate a new gain. Both devices have a set threshold; in Figure 4, the threshold is set to 0.5. A limiter scales back a signal so that its output level never exceeds the threshold. A compressor scales back a signal by a certain ratio. In Figure 4, this ratio is set to 0.5, so it reduces the signal by half the value it exceeds the threshold.

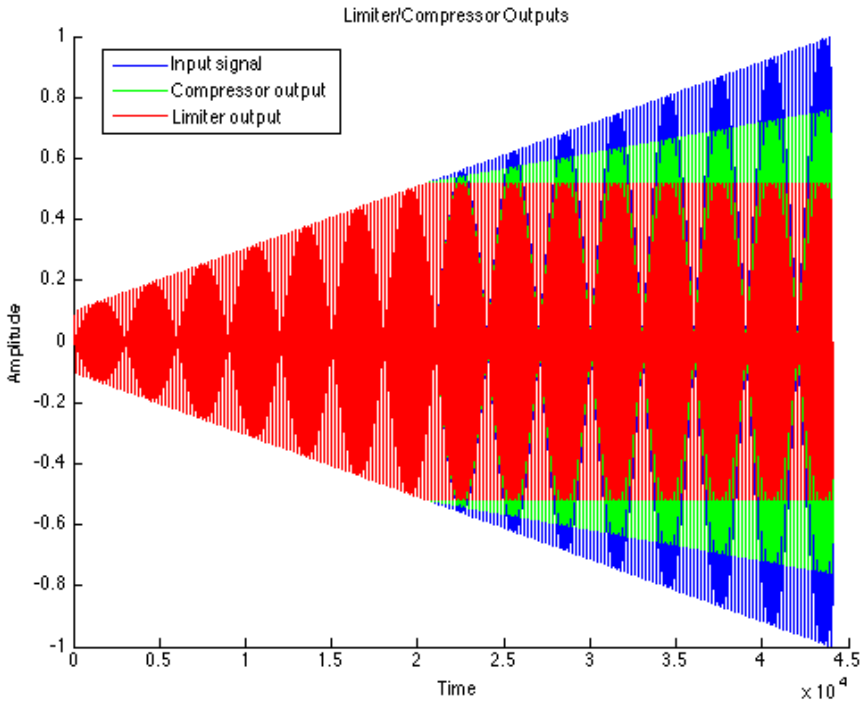

Figure 4: Compressor and limiter outputs

2.5 Reverberator

A reverberator is used to simulate the echoing of a room, in effect placing a synthesized signal in a real space. How to simulate this effect is still a widely discussed topic. For this project, we chose to implement a seminal method described by JA Moorer in the late 1970s[4]. The Moorer reverberator takes on two stages in series.

The first stage is a tapped-delay line. This attempts to simulate the first few arrivals of a multipath propagation. The individual delays should be relatively prime, in order to accidentally create strange frequency responses. Our first stage has 17 taps, the last of which arrives about 80 milliseconds after the input signal.

The second stage is a set of parallel comb filters. The comb filter attempts to simulate the random arrival of a large number of faint multipaths, in order to make the reverberation sound more full. The gains of the comb filter can be set to change the decay time of the filter. Our second stage uses 6 comb filters with delays ranging from 50 to 80 milliseconds.

2.6 Filters

An important part of synthesis is filters that allow carving of the frequency spectrum. While in many applications, sharp filters with low sidebands are more desired, musical applications actually prefer less extreme filters. To do so, we implement first and second order filters, as described in Reference [3].

First order filters support low-pass and high-pass filters, with 12dB/octave roll off. Second order filters support low-pass and high-pass filters with 24dB/octave roll off, as well as shelfing and peaking filters.

A shelfing filter allows you to specify a gain below or above a certain cutoff, while maintaining 0dB gain outside the passband. A peak filter allows you to specify a gain at a certain frequency, with a roll-off set by an adjustable Q-factor, again with 0dB gain outside the passband.

2.7 Ring Modulator

A ring modulator is a very simple device. It simply multiples a signal by a fixed frequency sine wave. The resulting signal output resembles the input, but shifted in frequency space. It can be used to create interesting pitch shifts and aliasing effects, especially on instruments that aren't inherently pitched such as drums.

Final Work Breakdown

Week 1 (Feb 16 - Feb 22)	Initial proposal and presentation Feb 18 Experiment with Android audio latency Feb 22
Week 2 (Feb 23 - Mar 2)	Nye - Begin porting ClickTrack to Android Ryan - Develop barebones sequencer, piano roll, or keyboard interface on Android
Week 3 (Mar 9 - Mar 15)	Spring break!
Week 4 (Mar 16 - Mar 22)	Nye - ClickTrack functionality on Android Ryan - Combine interface with ported audio production
Week 5 (Mar 23 - Mar 29)	Nye - Finish port of ClickTrack with Java interface Ryan - Refactor and make robust interface
Week 6 (Mar 30 - Apr 5)	**Updates** Begin expanding functionality for additional instruments and effects
Week 7 (Apr 6 - Apr 12)	Continue expanding functionality
Week 8 (Apr 13 - Apr 19)	Continue expanding functionality. Spectrum and envelope visualization tools
Week 9 (Apr 20 - Apr 26)	Nye - Additional effects and cleanup of audio engine Ryan - Additional effects and cleanup of interface
Week 10 (Apr 27 - May 3)	Completed project and working demo
Week 11 (May 4 - May 11)	Final report

References

[1] https://play.google.com/store/apps/details?id=com.singlecellsoftware.caustic

[2] https://github.com/thenyeguy/ClickTrack

[3] lzer, Udo. "DAFX digital audio effects." 2nd ed. Chichester, West Sussex, England: Wiley, 2011. Print.

[4] Moorer, James A. "About this reverberation business." Computer music journal (1979): 13-28.

[5] http://www.khronos.org/opensles/

[6] http://www.kvraudio.com/forum/viewtopic.php?t=375517