

Synthotron

Live Music Performance on Android

18-551, Spring 2014, Group 5
Michaels Nye & Ryan
mer1 & mnye

Project Overview

- Goal
 - Create an Android synthesizer that is usable for live performance
 - Provide a typical suite of effects for tone creation
 - Overcome system I/O latency by sequencing music rather than simulating a traditional instrument

Novelty

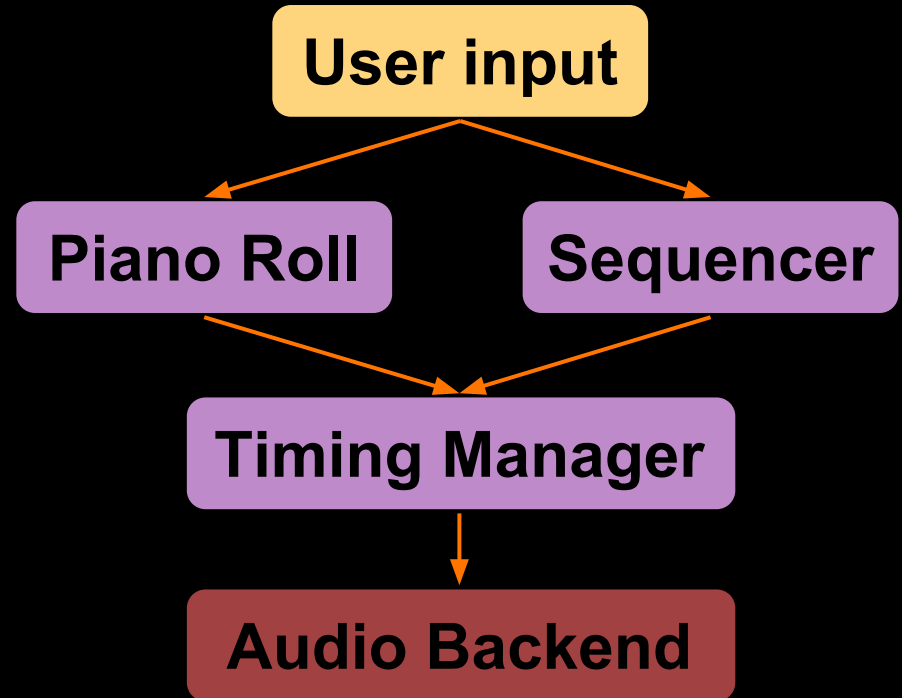
- A few projects dealt with audio processing
 - None tackle the problem of a self-contained synthesis package
- No good Android musical instruments
 - No application attempts to overcome the input latency issue

Datatypes

- Mono, CD-quality audio
 - 44.1kHz sample rate
 - 16-bit floating point samples
- Buffered audio processing
 - Buffer sizes determined by OS drivers
 - Nexus 7: ~3300 samples, 75ms
- System defined latency
 - Typical observed values: 350ms

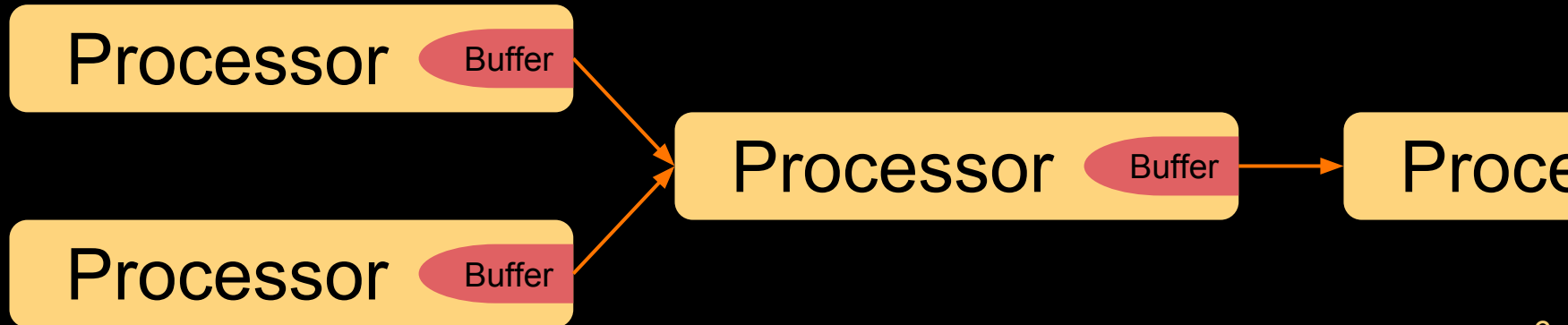
Data Flow

- User
 - Designs loops
- Timing interface
 - Java
 - Sends MIDI events
- Audio backend
 - Native C++
 - Generates audio



Backend Data Flow

- Composition of processing units
 - Connected by circular buffers
 - Each computes one full buffer at a time
 - 38 units in our backend



Code Summary

- Audio processing code
 - Written in C++, no external code
- Timing code and user interface
 - Written in Java, no external code
- External dependencies
 - Google Guava for simplified collections
 - Google Gson for serialization

Instruments and Effects

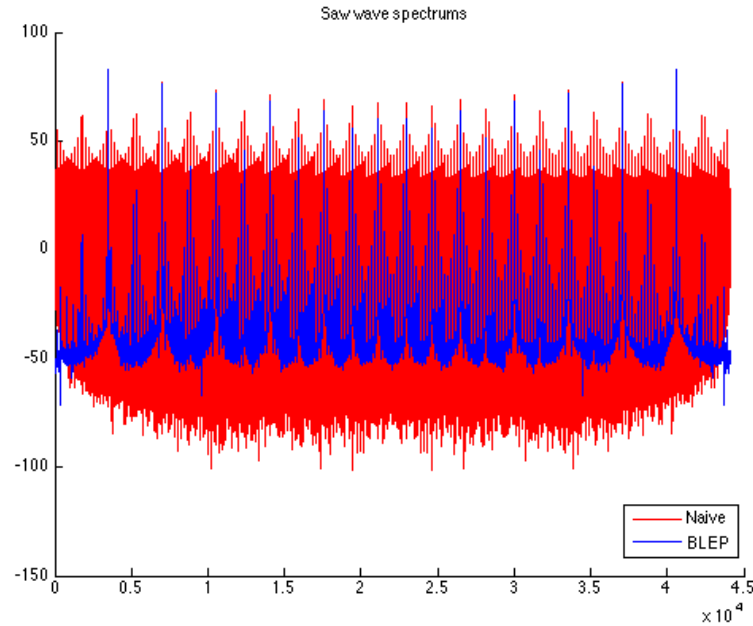
- Subtractive Synthesizer, Frequency Modulation Synthesizer
 - Oscillators with multiple waveforms
 - LFOs for tremolo and vibrato
 - ADSR envelopes
 - Adjustable filter
- Drum Machine
 - Ring modulator
 - Compressor
- Mastering Channel
 - 3-point equalizer
 - Reverb
 - Limiter

Oscillators

- Sine, square, saw, and triangle waves
- Antialiased using Polynomial Bandlimited Steps (PolyBLEP)
 - Adds smoothing to the discontinuous points

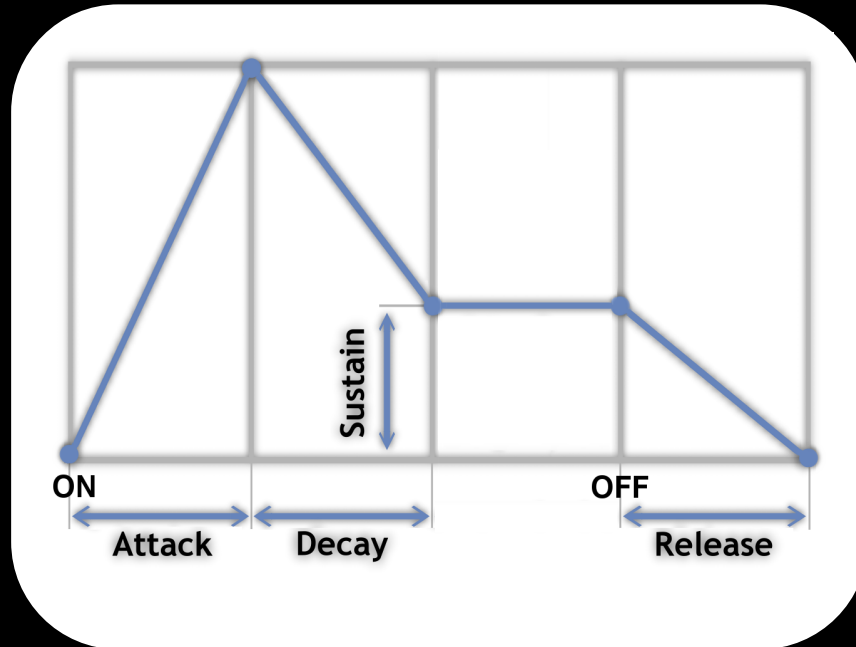
```
float Oscillator::polyBlepOffset(float t)
{
    float dt = phase_inc / (2*M_PI);
    if (t < dt) {
        t /= dt;
        return t*t - t*t - 1.0;
    } else if (t > 1.0 - dt) {
        t = (t - 1.0) / dt;
        return t*t + t*t + 1.0;
    } else {
        return 0.0;
    }
}
```

Oscillators



ADSR Envelopes

- Shapes the note being played



Low Frequency Oscillators

- Oscillator that is used to modulate other elements of a synthesizer
- Vibrato modulates an oscillator frequency
 - For example, $f(t) = \sin(2\pi t / F_s * LF0(t))$
- Tremolo modulates the signal amplitude
 - For example, $out(t) = in(t) * \text{pow}(10, LF0(t))$

Limiters and Compressors

- First, estimate an envelope of the amplitude of your signal
 - Leaky integration, $e(t) = \alpha * e(t-1) + (1-\alpha) * in(t)$
- Then, use the envelope to calculate a gain
 - A limiter makes sure the envelope *never* exceeds a threshold
 - A compressor allows the envelope to exceed the threshold, but reduces the level by a certain ratio

Reverberator

- Creates echos to create room sounds
- Two stages:
 1. Tapped delay line, with random coprime delays
 - Creates initial echos of shortest paths
 2. Comb filter
 - IIR filter that creates decay

Other Elements

- Filters for lowpass/highpass/bandpass
 - Uses second order IIR filters
- Ring modulator
 - Multiplies a signal by a high frequency sine wave to shift the signal in the frequency domain

Subtractive Synthesizer

The synthesizer interface is organized into five main sections, each with its own set of controls:

- MASTER**:
 - Gain: 0.0dB (represented by a knob)
 - Buttons: Save, Load
 - Test notes: C, E, G (represented by buttons)
- LFO**:
 - Frequency: 5Hz (represented by a knob)
 - Tremolo: 0.0dB (represented by a knob)
 - Vibrato: 0.00 (represented by a knob)
- OSCILLATORS**:
 - Oscillator 1:
 - Waveform: Saw (represented by a dropdown menu)
 - Transpose: 0 (represented by a knob)
 - Oscillator 2:
 - Waveform: Saw (represented by a dropdown menu)
 - Transpose: 0 (represented by a knob)
- ADSR**:
 - Attack: 0.05s (represented by a knob)
 - Decay: 0.10s (represented by a knob)
 - Sustain: 0.50 (represented by a knob)
 - Release: 0.30s (represented by a knob)
- FILTER**:
 - Mode: Low pass (represented by a dropdown menu)
 - Cutoff: 20000Hz (represented by a knob)
 - Gain: 0.0dB (represented by a knob)
 - Q: 5.0 (represented by a knob)

Frequency Modulation Synthesizer

The synthesizer interface is organized into five main sections, each with specific controls:

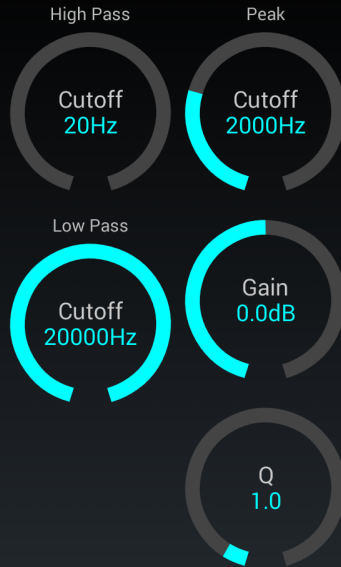
- MASTER**:
 - Gain: 0.0dB (knob)
 - Buttons: Save, Load
 - Test notes: C, E, G (buttons)
- LFO**:
 - Frequency: 5Hz (knob)
 - Tremolo: 0.0dB (knob)
 - Vibrato: 0.00 (knob)
- OSCILLATORS**:
 - Carrier**: Sine (dropdown), Transpose: 0.0 (knob)
 - Modulator**: Sine (dropdown), Transpose: 0.0 (knob), Intensity: 0.0 (knob)
- ADSR**:
 - Attack: 0.05s (knob)
 - Decay: 0.10s (knob)
 - Sustain: 0.50 (knob)
 - Release: 0.30s (knob)
- FILTER**:
 - Mode: Low pass (dropdown)
 - Cutoff: 20000Hz (knob)
 - Gain: 0.0dB (knob)
 - Q: 5.0 (knob)

Drum Machine



Mastering Channel

EQUALIZER



REVERB



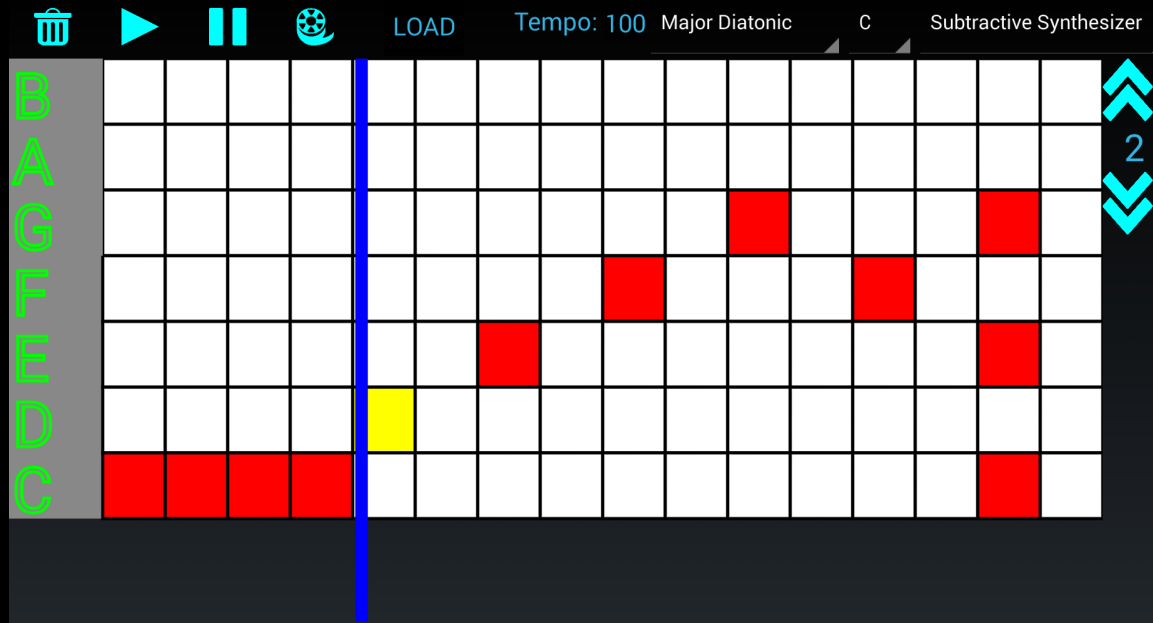
LIMITER



Timing Manager

- Keeps track of the scale being played, and notes entered for a measure
- When triggered, sends MIDI messages to audio backend on a timer
- Used by both the piano roll and sequencer

Piano Roll



Piano Roll

- The piano roll displays a grid of notes
 - One axis represents sixteenth notes
 - The other represents notes in a chosen scale
- Finger presses toggle notes on/off
 - Consecutive notes are held
- Plays a preview of the loop

Sequencer

Sequencer interface showing a 3x8 grid of notes for Sub Synth, FM Synth, and Drums. The interface includes a toolbar with icons for delete, play, pause, record, load, and repeat, along with a tempo display of 100.

Toolbar icons: Delete (trash can), Play (triangle), Pause (two vertical bars), Record (film strip), Load (LOAD text), Repeat (circular arrows). Tempo: 100

	1	2	3	4	5	6	7	8
Sub Synth								
FM Synth								
Drums								

Sequencer

- Can schedule 8 measures at a time, with loops created by the piano roll
- Can be set to loop forever
- Will also play the piano roll in time with the loops for live editing

Demo

- Unleash your creativity!
 - Design synth tones
 - Create loops
 - Play music

Work Breakdown

Week	Nye	Ryan
1-5	Basic audio backend	Piano roll interface
6	Java function calls for backend code	Clean up of piano roll, and basic timing code
7-9	Filling out effects chain	Filling out timing code, and sequencing multiple loops
10	Final integration	Final integration

Possible Future Work

- More instruments and effects
- More flexibility in loop lengths and note placement
- More consistent and friendly UI/UX
- Recording performances

Questions?

