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

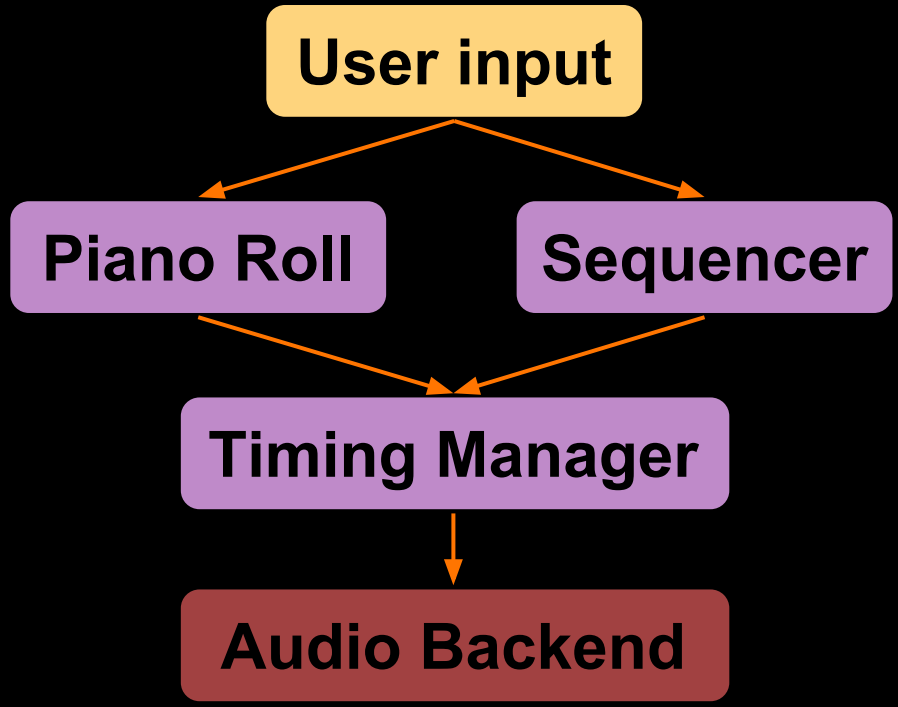
- A few projects dealt with audio processing
  - None tackle the problem of a self-contained synthesis package
- No good Android musical instruments
  - No application attempts to overcome the input latency issue

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

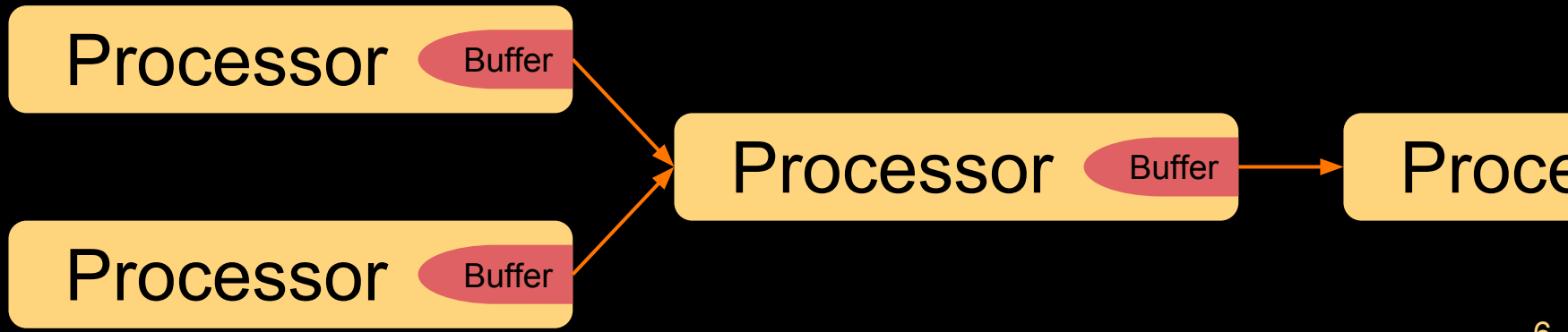
Data Flow

- 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

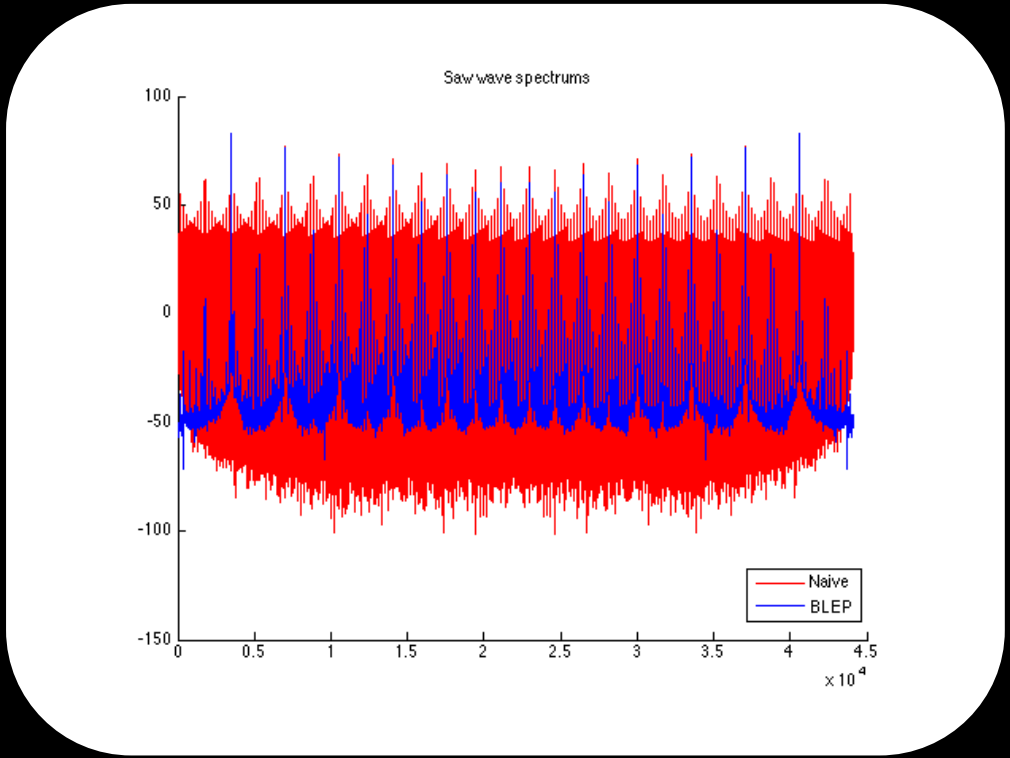
- Subtractive Synthesizer, Frequency Modulation Synthesizer
  - Oscillators with multiple waveforms
  - LFOs for tremolo and vibrato
  - ADSR envelopes
  - Adjustable filter
- Drum Machine
  - Ring modulator
  - Compressor
- Mastering Channel
  - 3-point equalizer
  - Reverb
  - Limiter

Oscillators

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

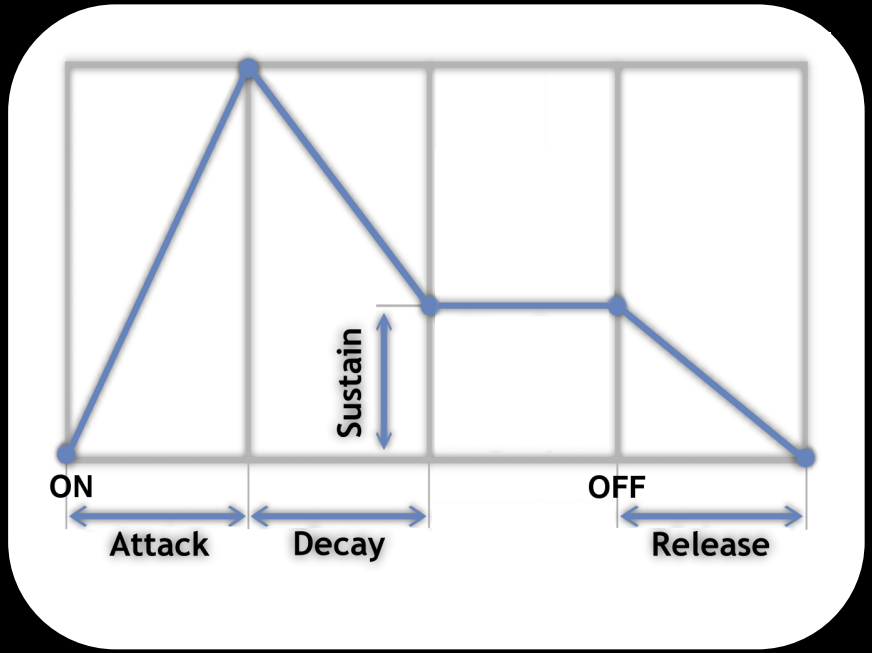
```
float Oscillator::polyBlepOffset(float t)
{
    float dt = phase_inc / (2*M_PI);
    if (t < dt) {
        t /= dt;
        return t+t - t*t - 1.0;
    } else if (t > 1.0 - dt) {
        t = (t - 1.0) / dt;
        return t*t + t+t + 1.0;
    } else {
        return 0.0;
    }
}
```

Oscillators



ADSR Envelopes

- Shapes the note being played



http://upload.wikimedia.org/wikipedia/commons/5/58/ADSR_envelope.png

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/Fs * LFO(t))`
- Tremolo modulates the signal amplitude
  - For example, `out(t) = in(t) * pow(10, LFO(t))`

Limiters and Compressors

- First, estimate an envelope of the amplitude of your signal
  - Leaky integration, `e(t) = α*e(t-1) + (1-α)*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

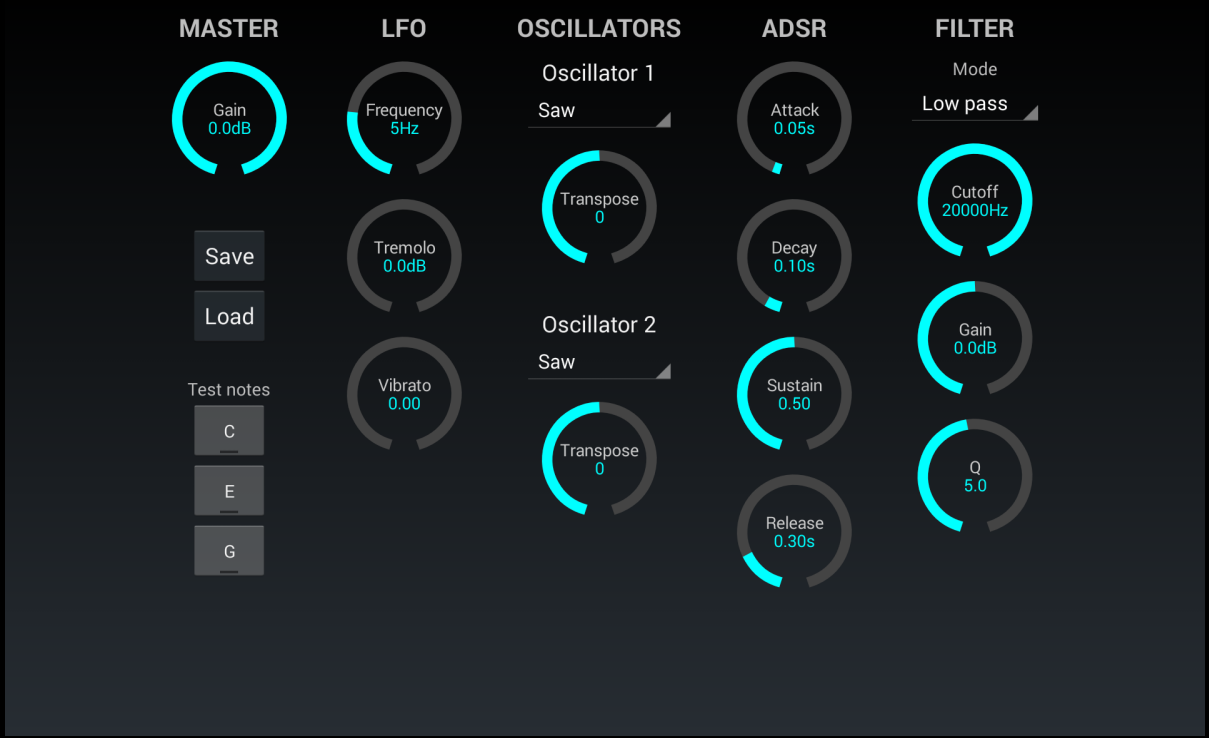
- Creates echos to create room sounds

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

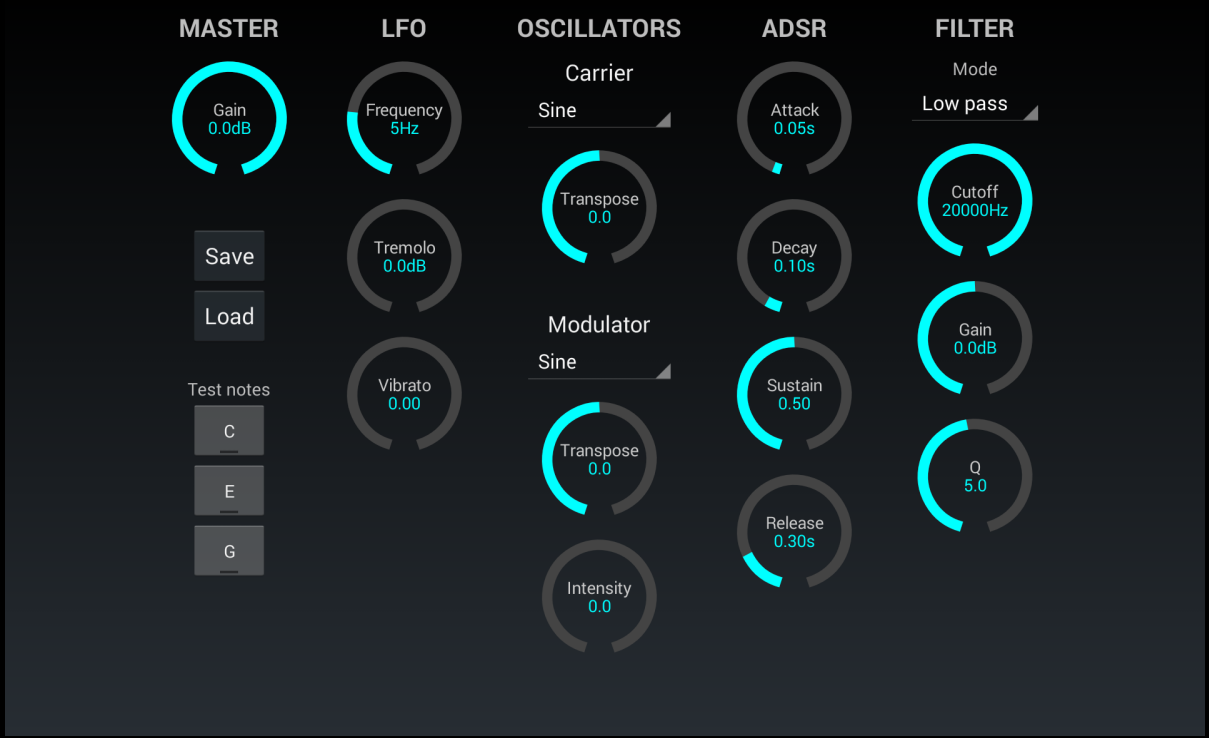
Other Elements

- Filters for lowpass/highpass/bandpeak
  - 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

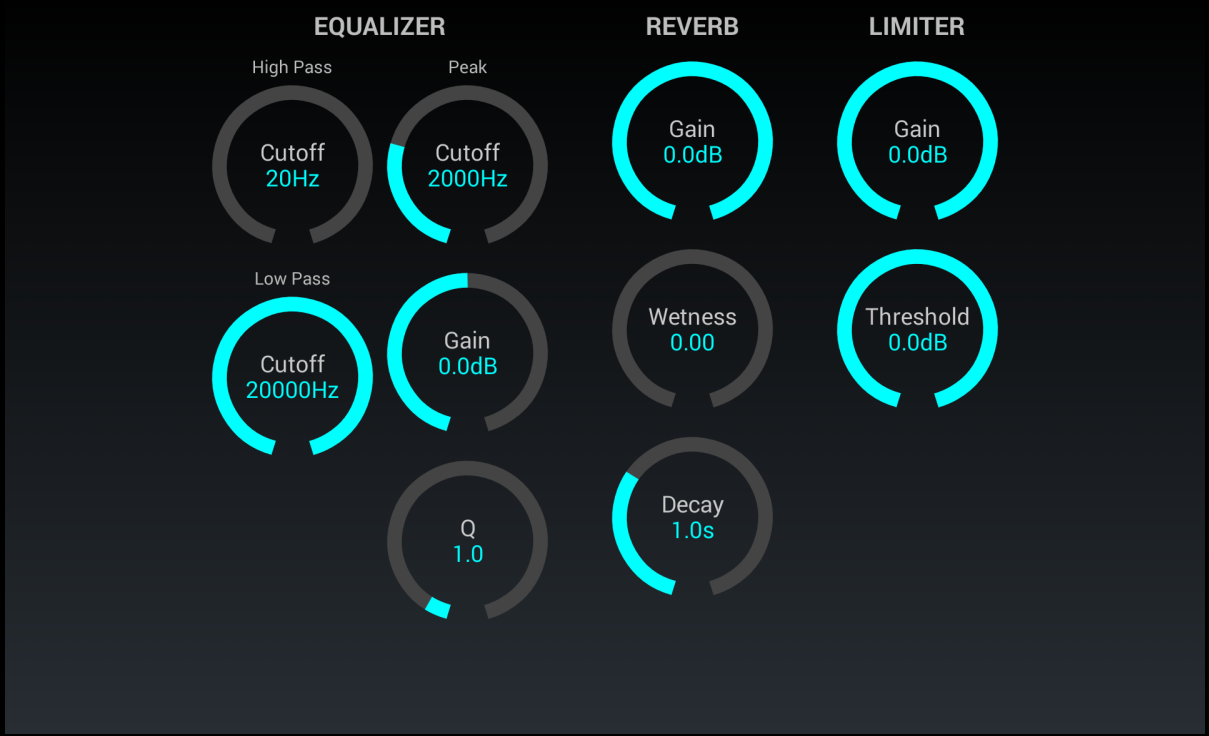


Frequency Modulation Synthesizer



Drum Machine

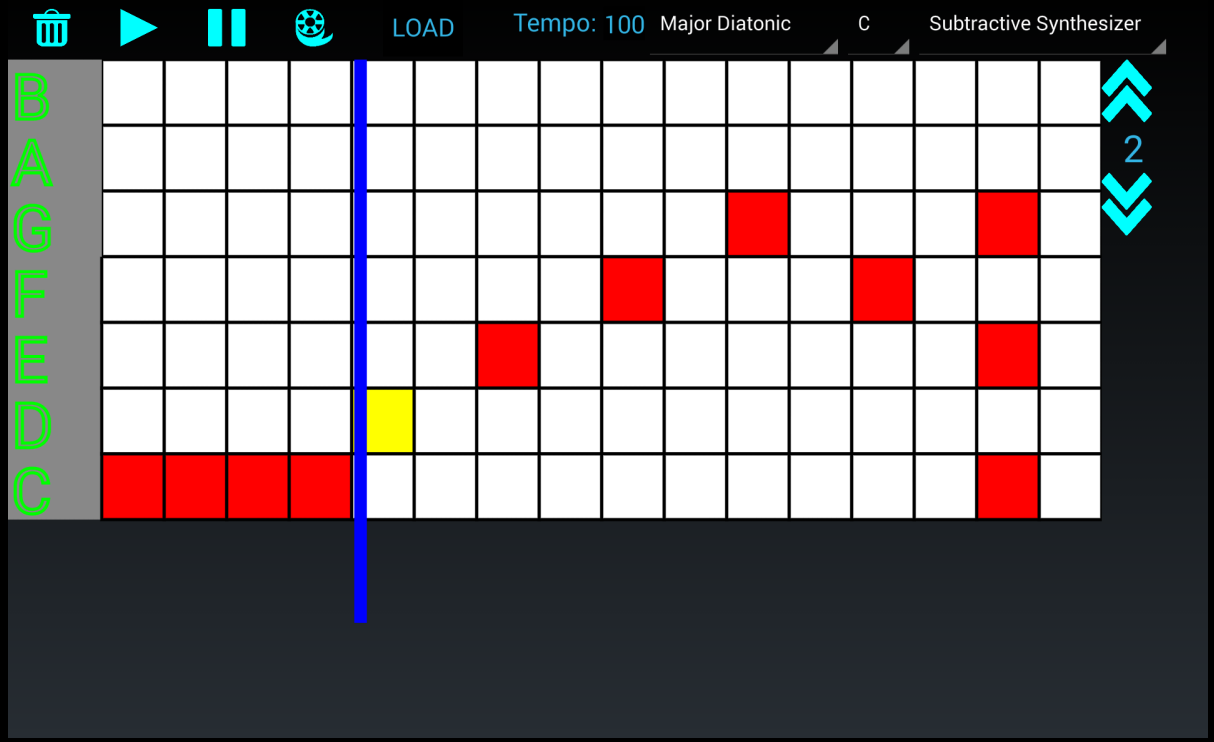
Mastering Channel



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

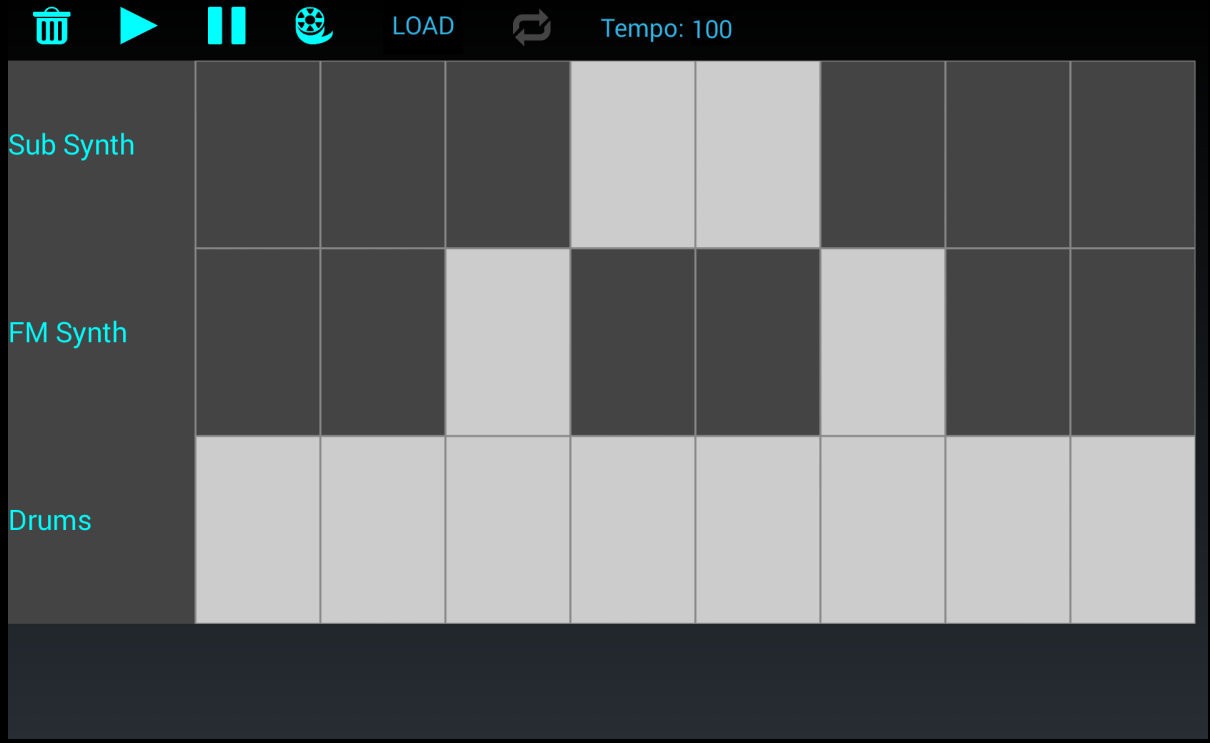
Piano Roll


LOAD
Tempo: 100
Major Diatonic
C
Subtractive Synthesizer
B
A
G
F
E
D
C
2

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

- 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?