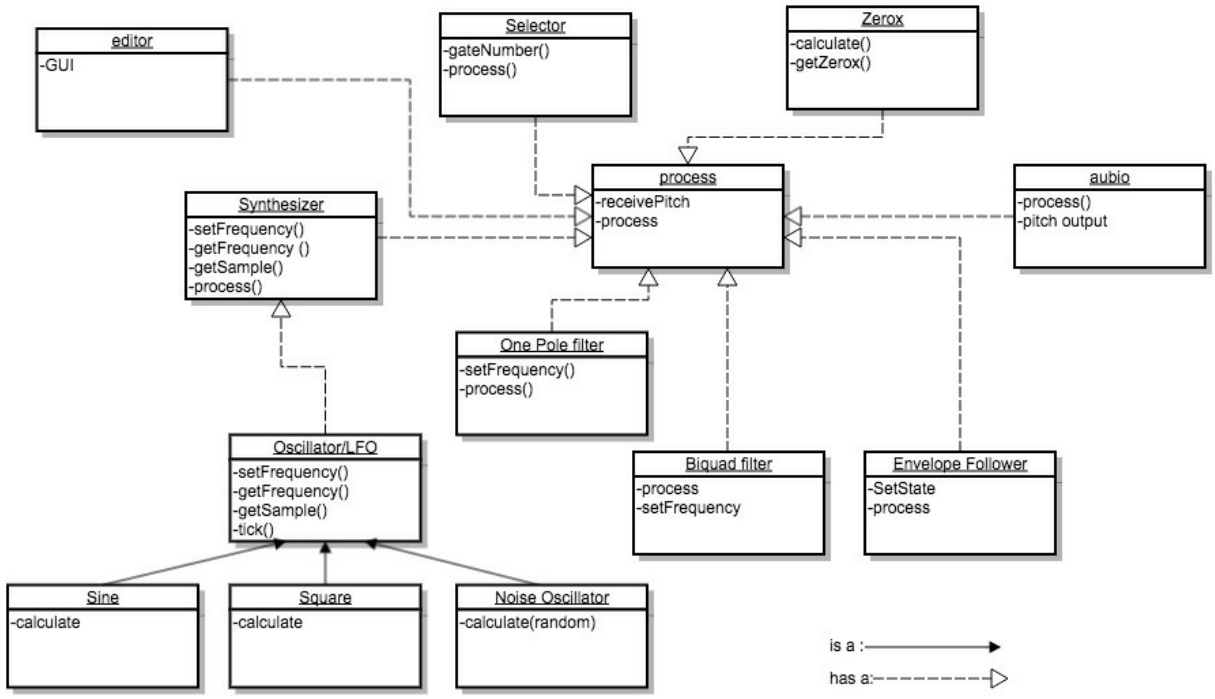
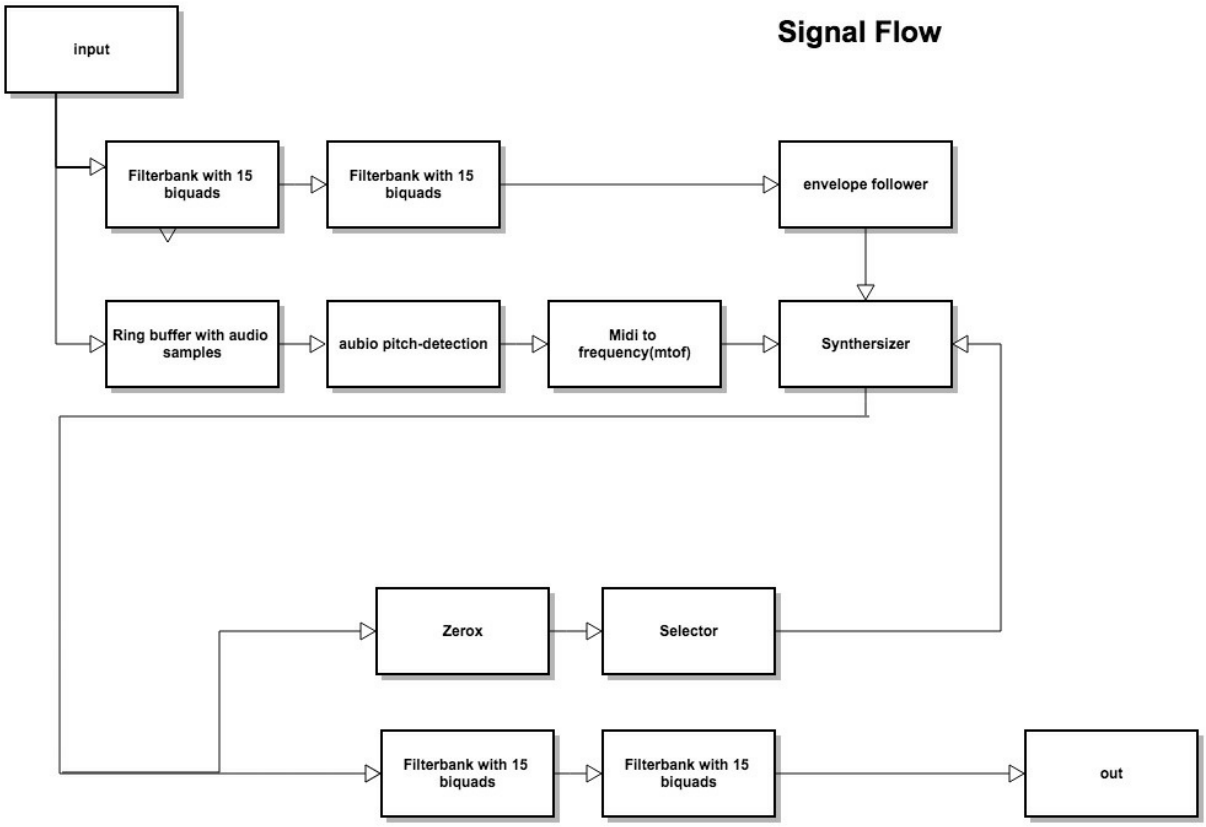
Guitar Synth Log

Blokschema:



signal flow:



classes:

- PluginProcessor
- PluginEditor
- BiquadFilter
- OnePoleFilter
- Zerox
- Selector
- Synthesizer
- oscillator
- sineWave
- SquareWave
- NoiseOscillator

GUI:

Parameters aangemaakt in for loops en in verschillende structs, hiermee word de GUI in verschillende vlakken onderverdeeld.

Filters:

Voor de filter implementatie heb ik gekozen voor een biquad filter, direct form II. Deze filter werkt met twee delaybuffers. Voor de buffers gebruik ik een aantal vectors. Ik ben begonnen

met een mono implementatie, dus waarbij je voor ieder stereo-kanaal twee buffers moest aanmaken. Nadat ik dit werkend had ben ik gaan nadenken hoe ik dit efficiënter kon maken.

Toen kwam ik op het idee om 3d vectors te gebruiken. Ik heb 1 vector aangemaakt waarin 2 vectoren zitten voor de kanalen(stereo) deze vectoren bevatten ieder weer twee vectoren voor de delayBuffers.

```
channelBuffers[channel][delayBuffer][delay];
```

De filters maken het geluid op het moment wel heel erg zacht, dus ik denk dat ik nog wat aan de gain moet draaien. Ik moet nu de envFollower versterken met 1000.

for loops:

Omdat er een behoorlijk aantal filters in een vocoder zitten, 60 stuks, heb ik mijn programma zo geschreven dat deze worden aangemaakt en ook binnen de process-functie heb ik veel for loops gebruikt om de code op deze manier flexibel en overzichtelijk te houden.

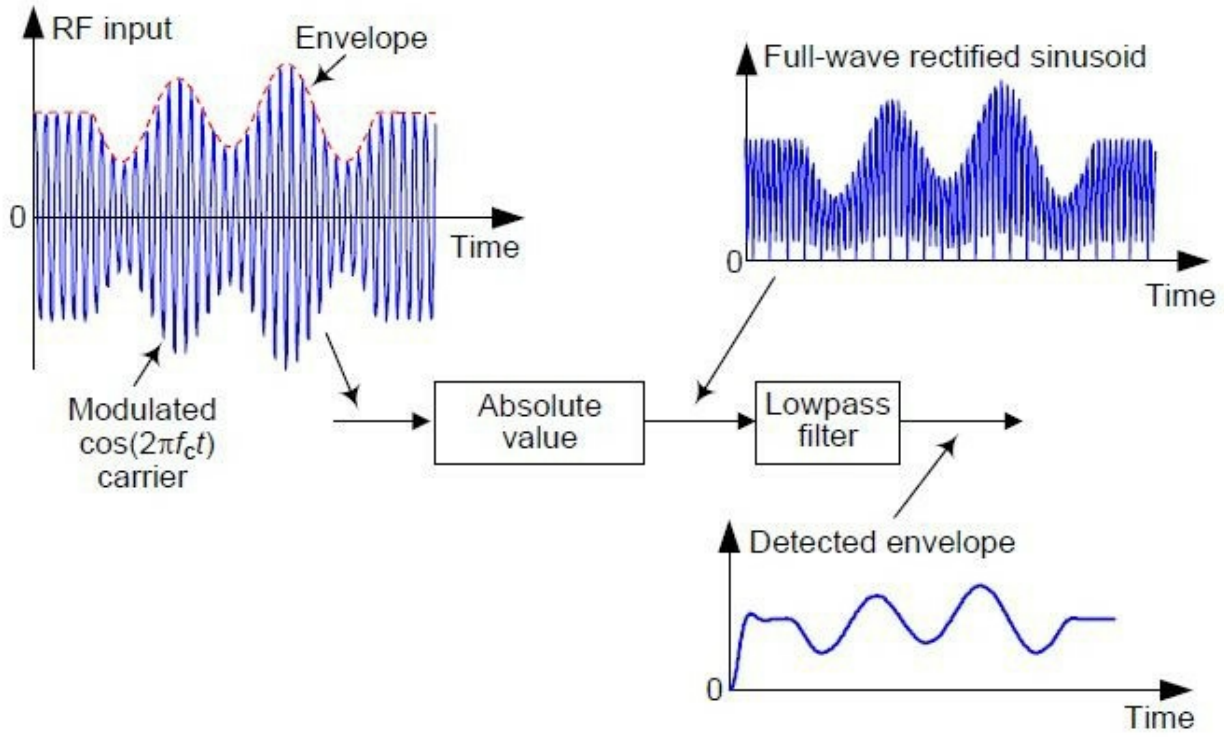
```
bandPassFilters = new Biquad*[60];


for (int i = 0; i < 60; i++){
    bandPassFilters[i]
    = new Biquad(1.0, filterFreqs[i % 15]
      / lastSampleRate, 40.0);
}
```

```
for (int filterIndex = 0; filterIndex < 15; filterIndex++){
             float filterSignal = bandPassFilters
             [filterIndex + 15]->process(channel, synthSample);
             addedfilterSignal = (filterSignal * envFollowValues[filterIndex]) + ad
         }
```

Zerox:

Ik voor mijn zero crossing class het zerox~ object van Max als referentie genomen. Ik heb gekeken naar de werking van het object en naar de output. Hierna heb ik gekeken hoe ik dit zelf zou kunnen maken en ik heb nadat ik de code had gemaakt de output van mijn Zerox class vergeleken met het zerox~ object van Max om zo mijn output te verifiëren.

Envelope follower:



Dit is het systeem dat heb gebruikt voor mijn envelope follower. Het artikel is hier te lezen. De absolute value bereken ik door de absolute waarde te pakken van het inkomende signaal over een x aantal samples. Als alle waarden bij elkaar zijn opgeteld deel het ik weer door x.
Dit gaat vervolgens door een one pole lowpass filter heen.

oscillatoren:

Naast het analyse deel en filtering heeft een vocoder ook twee oscillatoren. Als basis heb ik de classes uit blok csd2c gebruikt. Voor de noise oscillator heb ik de sinus formule vervangen met een random functie om op deze manier ruis te genereren. Op dit moment gebruik ik FM-synthese om een vollere klank te krijgen.

Aubio:

Ik vond het lastig om aubio aan de praat te krijgen binnen Xcode. Uiteindelijk bleek dat ik het framework via een ander menu moest toevoegen.

Hierna ging de rest relatief makkelijk. Ik moest wel de window en hopsize aanpassen om goede resultaten te halen.
In eerste instantie stond de window- en hopsize redelijk laag en hierdoor trackte aubio alleen de hoge noten redelijk, alle lagere frequenties niet. nadat ik de windowsize had veranderd naar 4096 en de hopsize naar 1024 werkt het goed.

Ik koppel aubio aan JUCE door de samples afkomstig uit de buffer van JUCE weg te schrijven naar een ringbuffer, deze geef ik doormiddel van een pointer door aan aubio.

Van aubio naar de synth:

Aubio pakte de frequenties redelijk goed maar er zaten wel pieken en dalen in de pitch in Hertz die aubio teruggaf.
Om de fluctuatie tegen te gaan haal ik de frequenties door een lowpass filter om de uitschieters eruit te filteren.

Hiernaast komen de frequenties die aubio geeft niet exact overeen met de frequenties van ons westers stemmings-systeem. Om dit op te lossen reken ik de frequenties van aubio eerst om naar de MIDI-nootwaarde waar de frequentie het dichst bij zit. Hierna geef ik die MIDI-nootwaarde door aan een mtof functie die het weer omzet naar de juiste frequentie.