Design

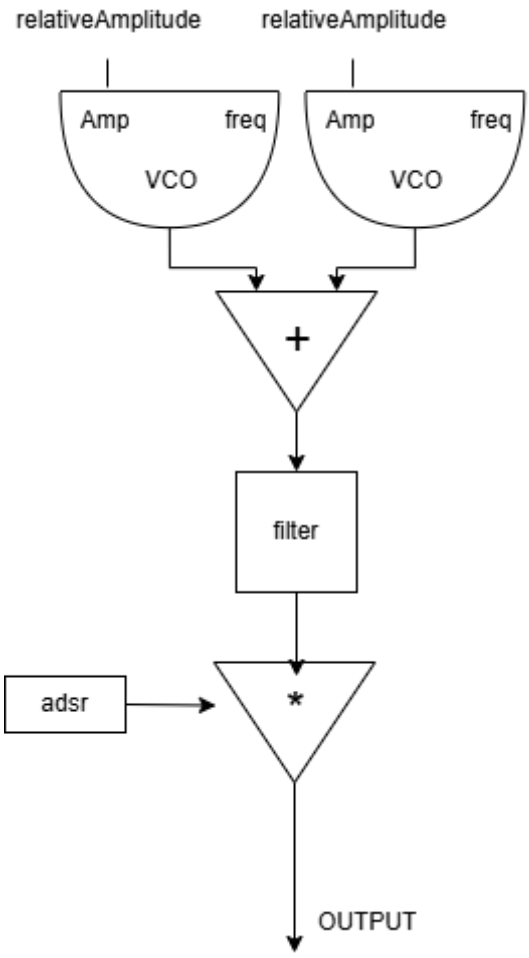
Sytnh Focus

Voor dit project heb gekozen voor synth focus omdat we met andere vakken al veel sequencing en melodie generatie hebben gedaan. Ook wilde ik graag kijken hoe ik de dingen die we bij synth basis en anologe studio in code werken. Voor mijn UI heb ik besloten dat ik van de lange lijst van vragen af wil en deze wil vervangen door een commando systeem. Bij het starten van de code kan je selecteren welke synth je wil gebruiken om deze vervolgens met een lijkt waan commandos aan te sturen.

Subsynth

Als eerste synthesizer ben ik gegaan vaan een subtractieve synth omdat veel (semi-)analoge synths hier gebruik van maken en ik dat graag wilde reproduceren. De subsynth zoals ik plan werkt door verschillende golven bij elkaar op te tellen waardoor en een complex spectrum aan boventonen ontstaat. Door dit te filteren met in dit geval een low-pass filter produceer je klanken.

Audioflow diagram



Pseudocode

```
class subSynth {
bank OscillatorBank()
filter Filter()
adsr Adsr()

void setNoteFrequency(freq) {
    oscillatorBank.setFrequency(freq)
}

void setFilterFrequency(freq) {
    filter.setFrequency(freq)
}

void tick() {
    oscillatorBank.tick()
```

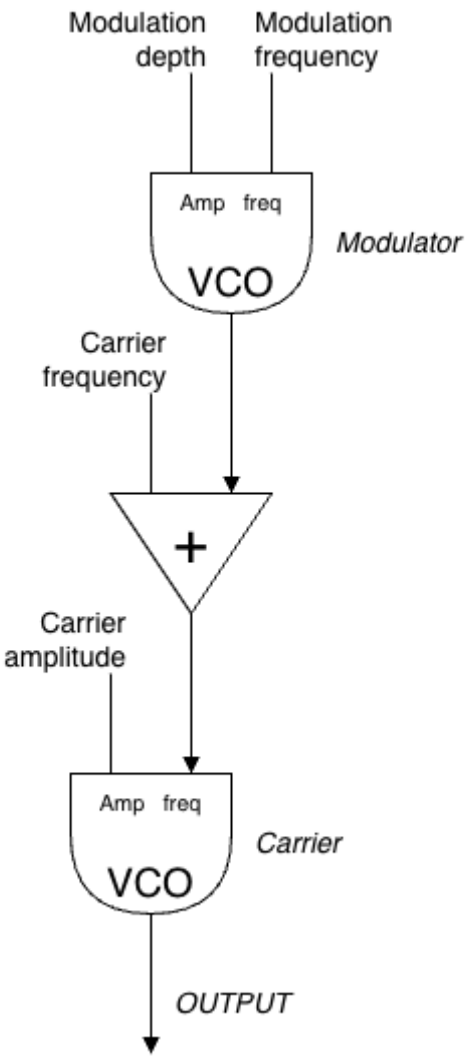
```
    adsr.tick()
}

float getSample() {
    float sample = oscillatorBank.getMixedSample()
    sample = filter.processSample(sample)
    sample = sample * adsr.getCurrentAmplitude()
    return sample
}
}
```

FMSynth

Mijn tweede synth wordt een FM synth. Ik heb FM8 een aantal keer gebruikt en het leek me interessant omdat het met vrije simpele wiskunde interresante klanken genereed. FM werkt door met een carrier wave die gemoduleerd wordt door een modulator wave. Voor nu hou ik het bij één carrier en één modulator om het simpel te houden, het zou wel eventueel mogelijk zijn om andere golven dan de gebruikelijke sinus toe te passen om ingewikkeldere waveforms te produceren.

Audioflow diagram



Pseudocode

```
class FMSynth(float carrierFreq, float modulatorFreq, float modIndex, float
samplerate) {
    carrier = new Oscillator(carrierFreq)
    modulator = new Oscillator(modulatorFreq)
    moulationIndex = modIndex
    baseCarrierFreq = carrierFreq

    float getSample() {
        float instantFreq = baseCarrierFreq + modulationIndex * modValue *
baseCarrierFreq;

        carrier.setFrequency(instantFreq)
```

```
        return carrier.getSample()
    }

    void tick() {
        modulator.tick()
        carrier.tick()
    }
}
```