Design | CSD

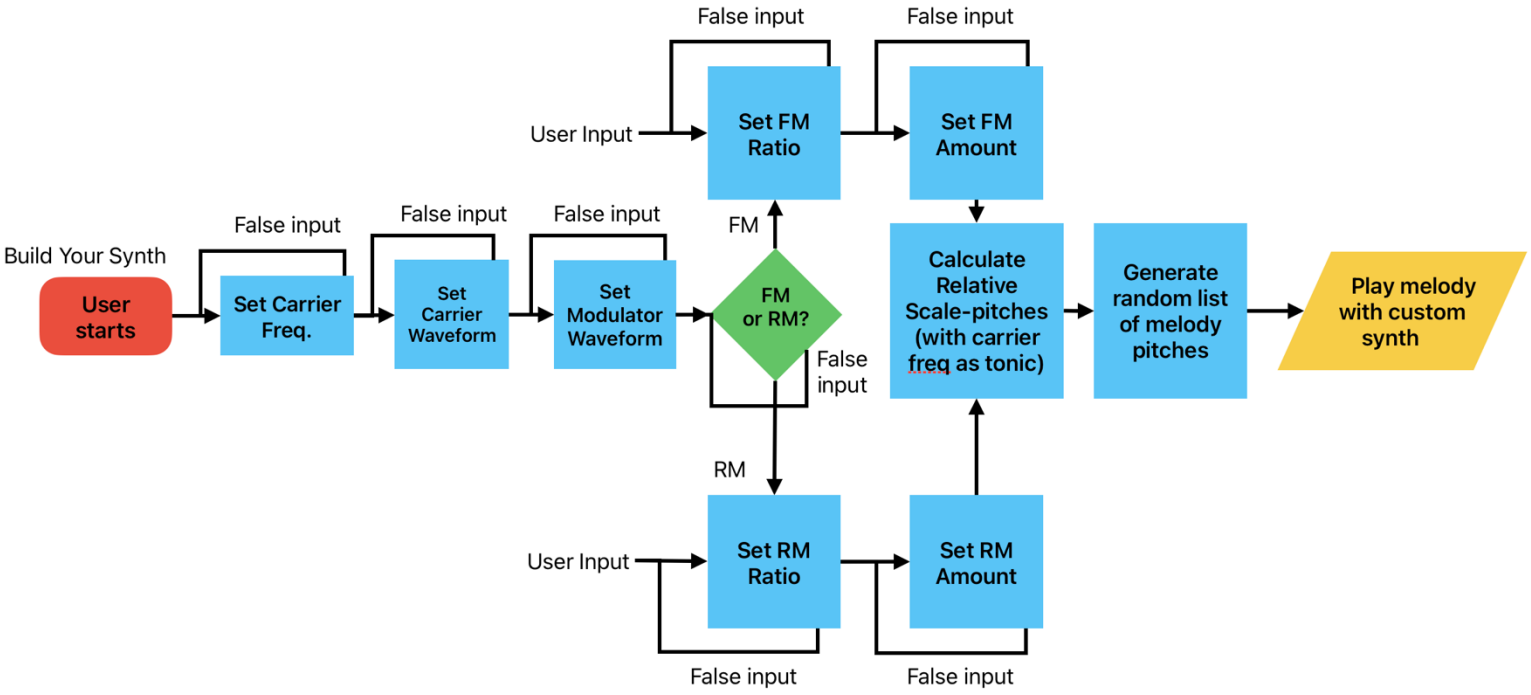
Ik ga voor de keuze SYNTH.

Onderbouwing keuze

Hoewel ik geen KO heb gekozen, vind ik synthesizers enorm interessant. Graag wil ik meer weten hoe deze in elkaar zitten op technisch gebied en hoe ik dit kan toepassen. Ik ga voor FM en RM als synthesizers. Het lijkt mij leuk om deze twee tegen over elkaar te zetten om te zien hoe deze verschillend in elkaar zitten.

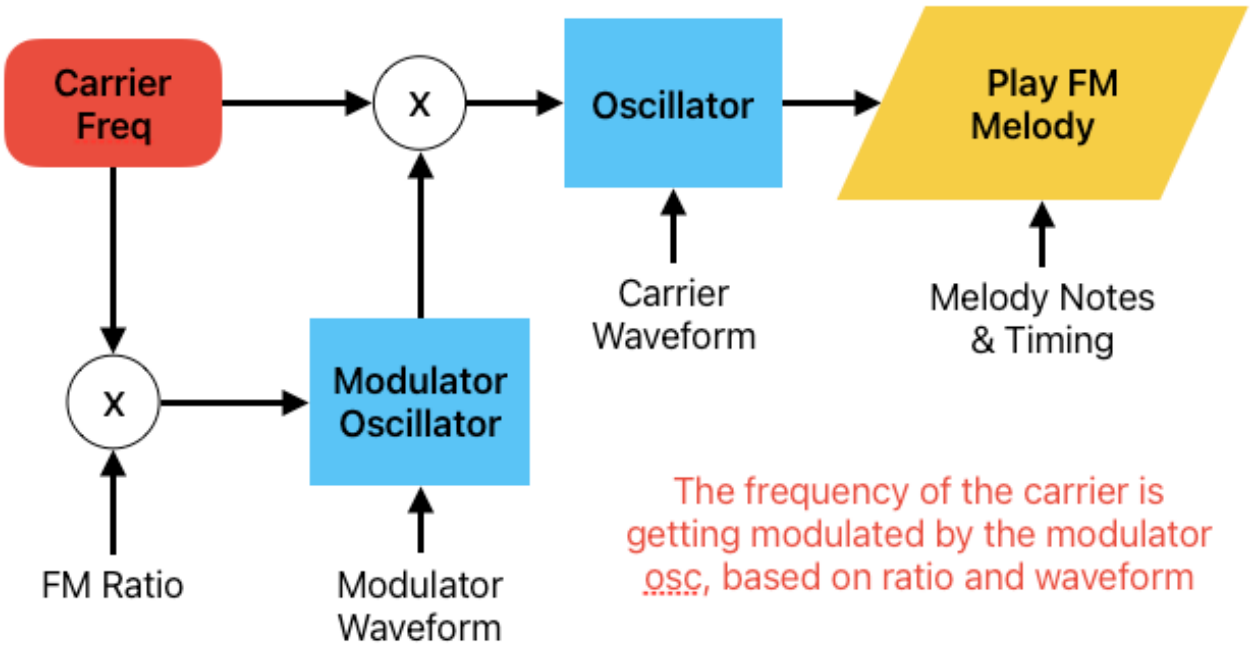
User Flow

Om aan de slag te gaan met deze twee synthesizers maak ik eerst een overkoepelend ontwerp van hoe de UI-flow eruit gaat zien. Mijn doel is dat de user intuïtief een melodie kan genereren met een synthesizer-setting naar zijn keuze. Die keuzes heb ik overzichtelijk gemaakt in de volgende flowchart:



FM

Voor de FM-synthesizer is een carrier- en modulator-oscillator nodig. Hierbij stuurt de modulator de frequentie van de carrier aan met een bepaalde freq-ratio. Hierbij een flowchart voor hoe dit in zijn werking gaat:

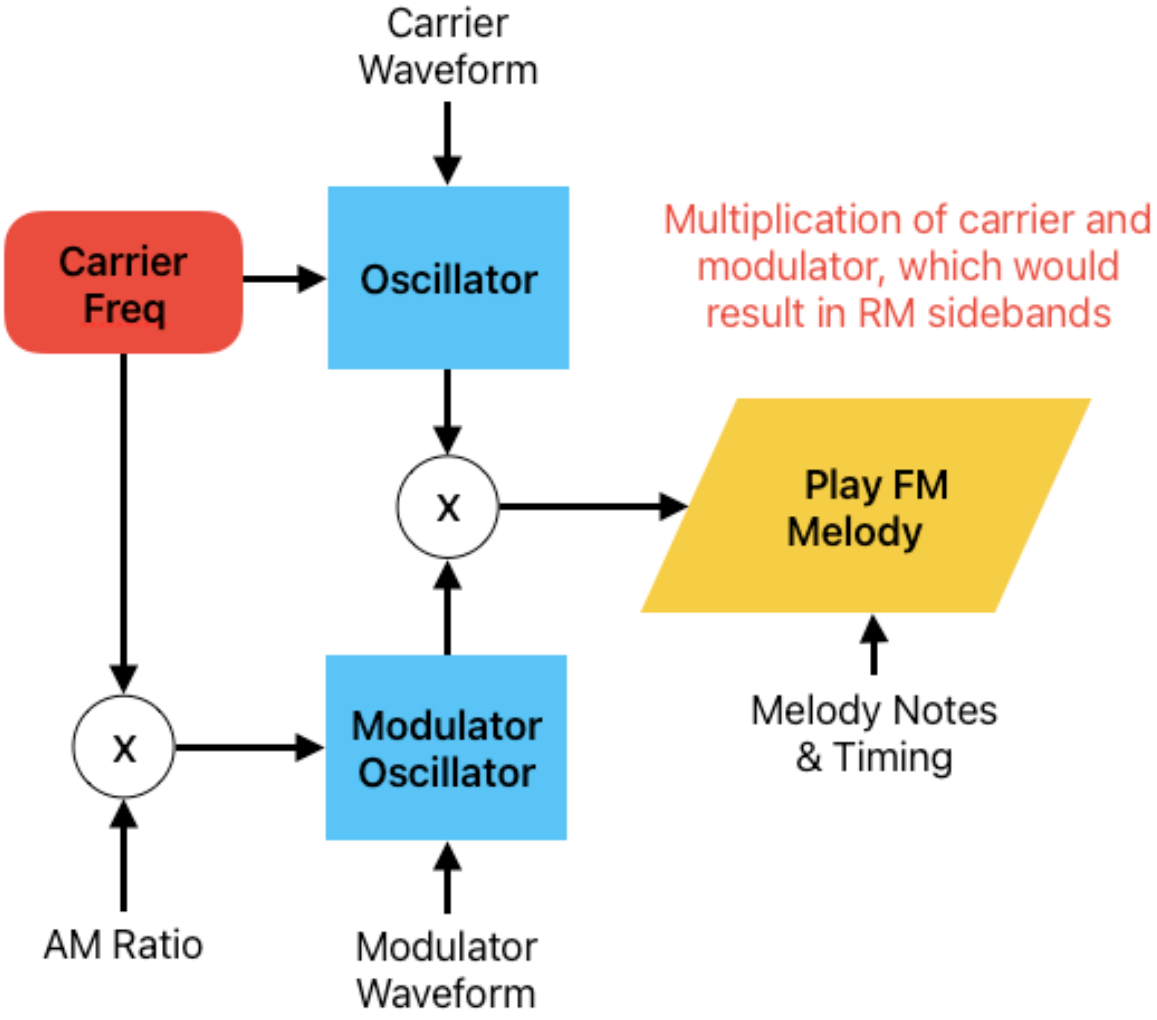


En daarvoor de pseudo code:

```
// PSEUDO CODE RM

//Variables
float ratio = "ratio between carrier and modulator oscillating frequency"
float depth = "how much RM is applied to the carrier"
float melFreq = "frequency played by melody"
float carFreq = "frequency played by carrier"
float modFreq = "frequency played by modulator"
float carAmp = "amplitude of carrier"
float modAmp = "amplitude of modulator"
float soundoutput = "sound being played"
float outputSignal = "signal of carrier and modulator combined with RM"

//Class
class Oscillator() {
  float freq
  float amp
  //Incorporates chosen waveform as well
  //outputs sample signal playable by playSound function
}

//Initialisation
melFreq = 440.0
ratio = 2.5
modFreq = melFreq * ratio
depth = 1.0
carAmp = 1.0
modAmp = depth * ratio * melFreq

//Frequency modulated by modulator
modulator = new Oscillator(modFreq, modAmp)
carrier = new Oscillator(carFreq, carAmp)
outputSignal = carrier.output * modulator.output

//Functions
function getSample() { //gets the sample of an oscillator }
function playSound() {
  //play signal of thisPhase
  soundoutput = outputSignal
}
```

RM

Voor de RM-synthesizer is ook een carrier- en modulator-oscillator nodig. Ring Modulation berekent het verschil tussen de carrier en modulator, en geeft als output de sidebands van deze frequentie (carrier-verschil en carrier+verschil). Dit gebeurt met een bepaalde ratio. Hierbij een flowchart voor hoe dit in zijn werking gaat:



En daarvoor de pseudo code:

```
// PSEUDO CODE FM

//Variables
float ratio = "ratio between carrier and modulator oscillating frequency"
float depth = "how much FM is applied to the carrier"
float melFreq = "frequency played by melody"
float carFreq = "frequency played by carrier"
float modFreq = "frequency played by modulator"
float carAmp = "amplitude of carrier"
float modAmp = "amplitude of modulator"
float soundoutput = "sound being played"

//Class
class Oscillator() {
  float freq
  float amp
  //Incorporates chosen waveform as well
  //outputs sample signal playable by playSound function
}

//Initialisation
melFreq = 440.0
ratio = 2.5
modFreq = melFreq * ratio
depth = 1.0
carAmp = 1.0
modAmp = depth * ratio * melFreq

//Frequency modulated by modulator
modulator = new Oscillator(modFreq, modAmp)
carFreq = melFreq + modulator.output
carrier = new Oscillator(carFreq, carAmp)

//Functions
function getSample() {//gets the sample of an oscillator}
function playSound() {
  //play signal of thisPhase
  soundoutput = getSample(carrier)
}
```