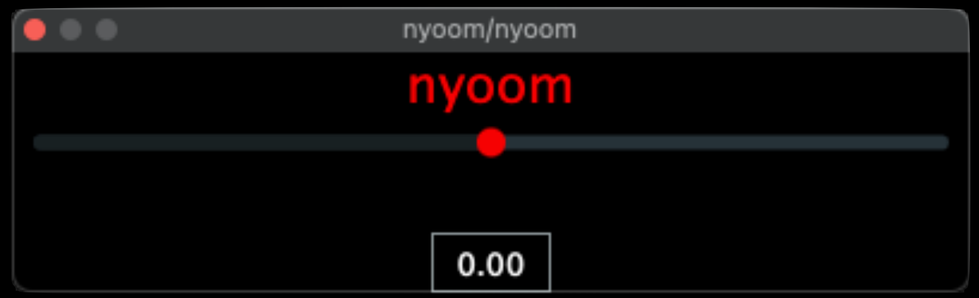
PRESENTATIE CSD2D
SEMUEL LEIJTEN

~~fake doppler~~

~~dopplinator~~

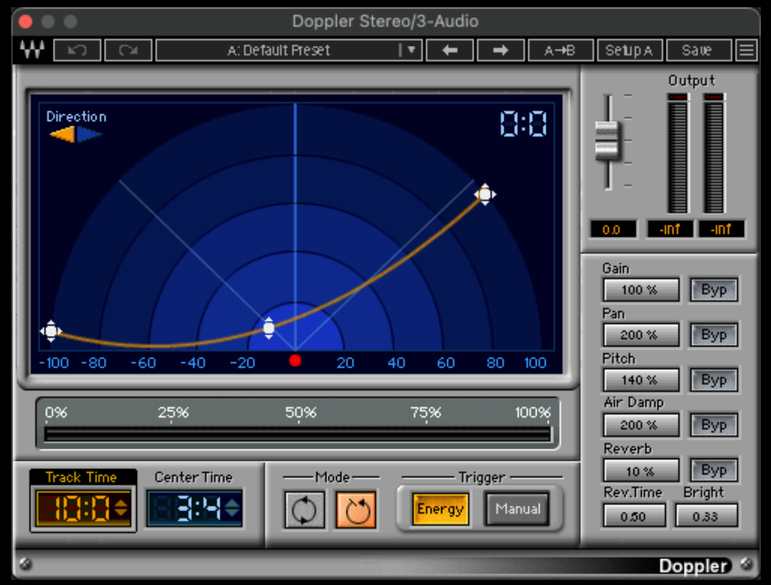
nyoom

a single parameter audio effect plugin
that can make any sound go nyoom



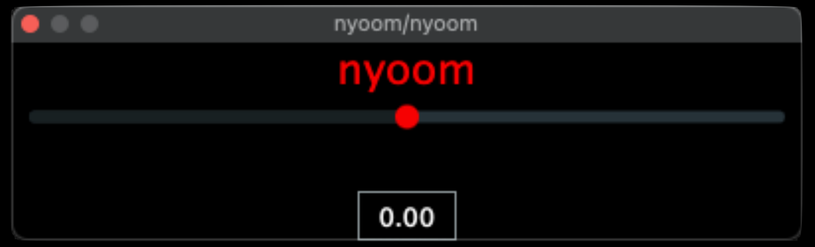
made for lazy sound designers who want to a simple no-frills
Doppler effect that’s convincing enough and easy to use

other Doppler effect plugins



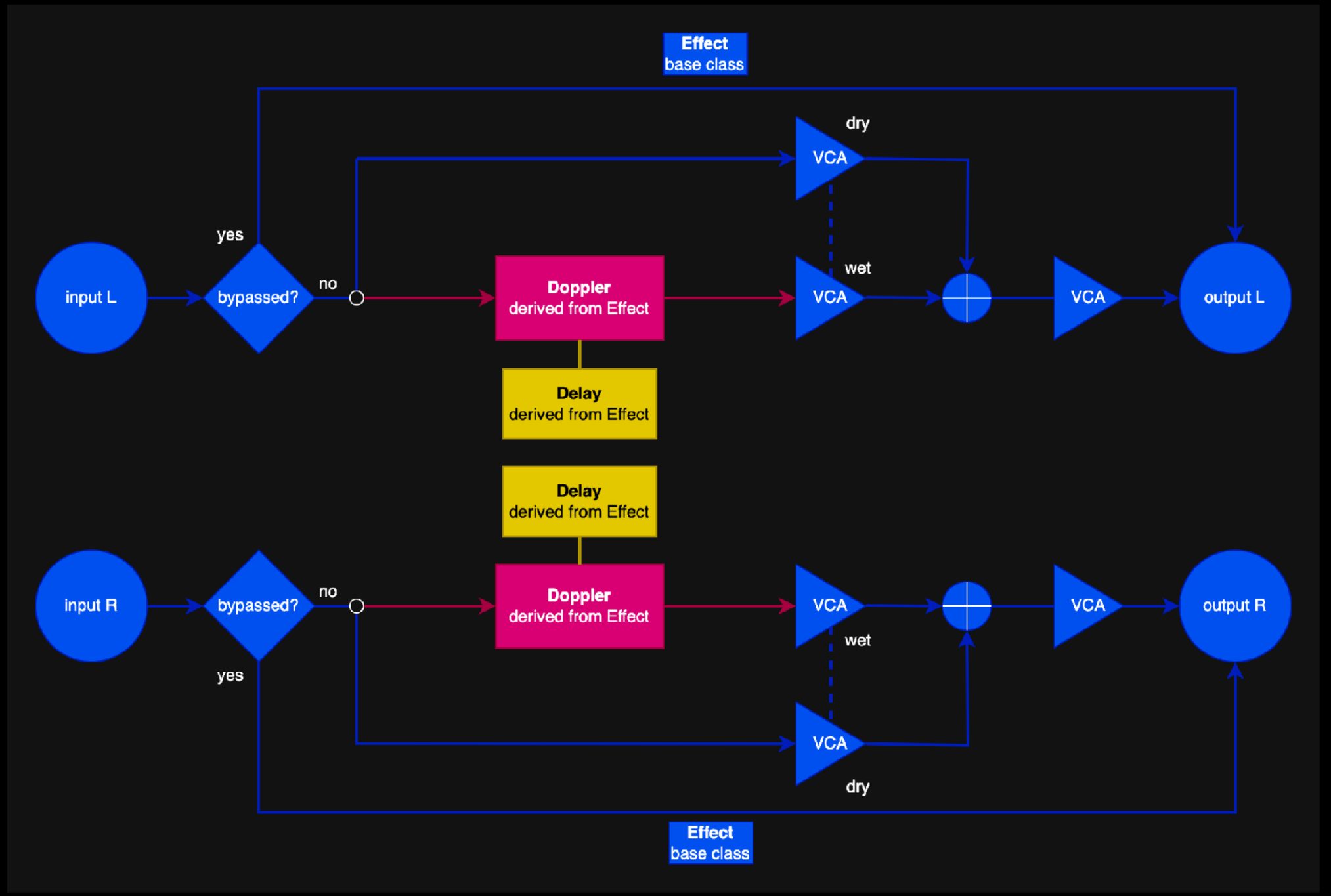
- overly complicated
- like a million parameters that just make you feel insecure
- can’t even control the position in the trajectory
- need a PhD to understand

nyoom



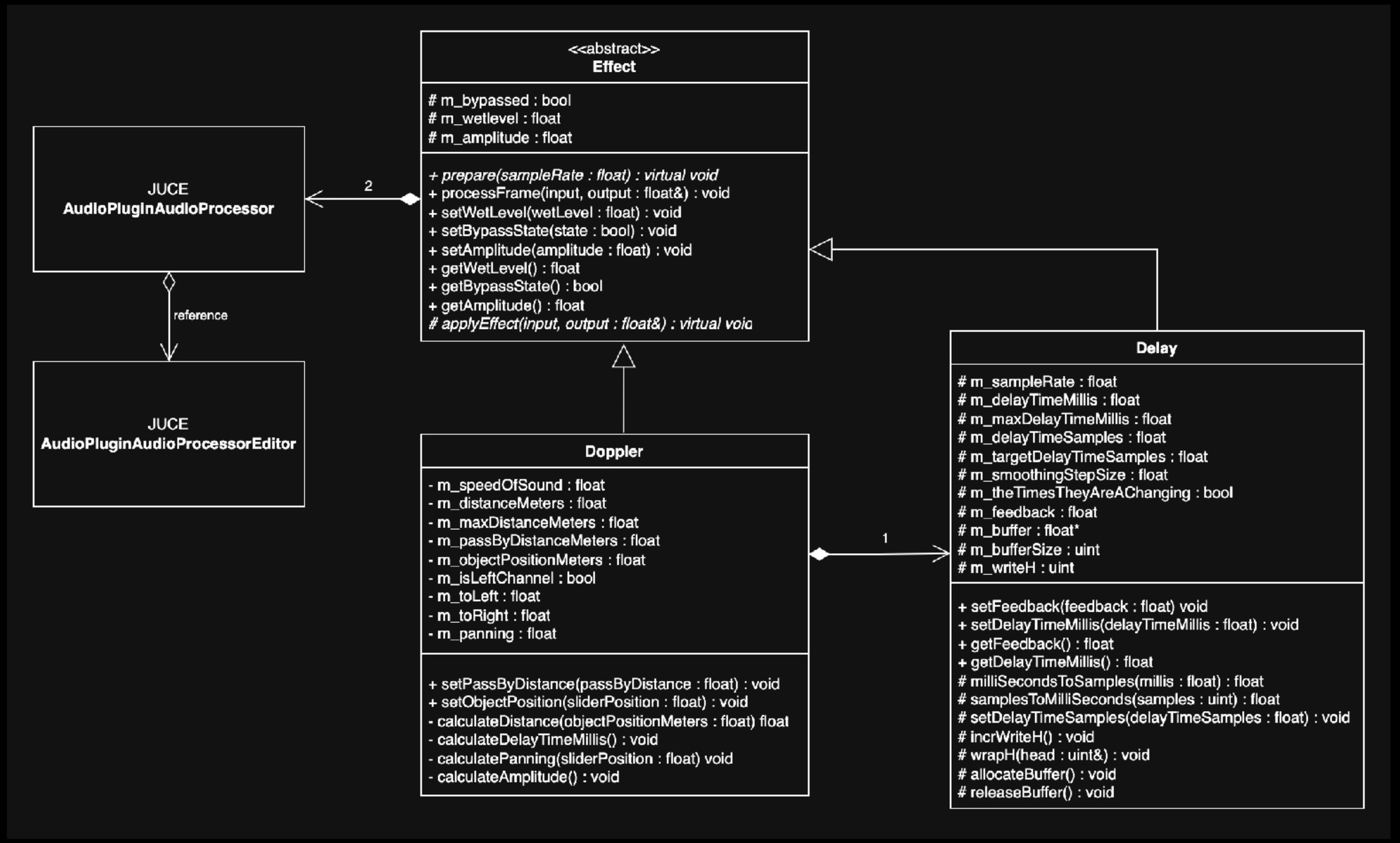
- simple, easy to use
- literally one parameter
- control the position in the trajectory and speed
- takes one second to understand

AUDIO FLOW DIAGRAM



SEMUEL LEIJTEN

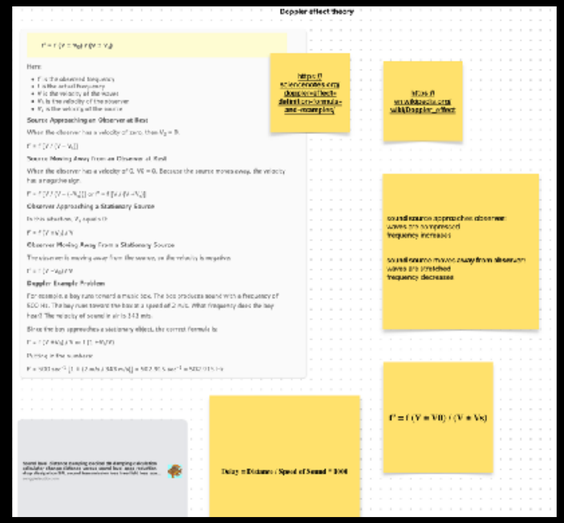
CLASS DIAGRAM

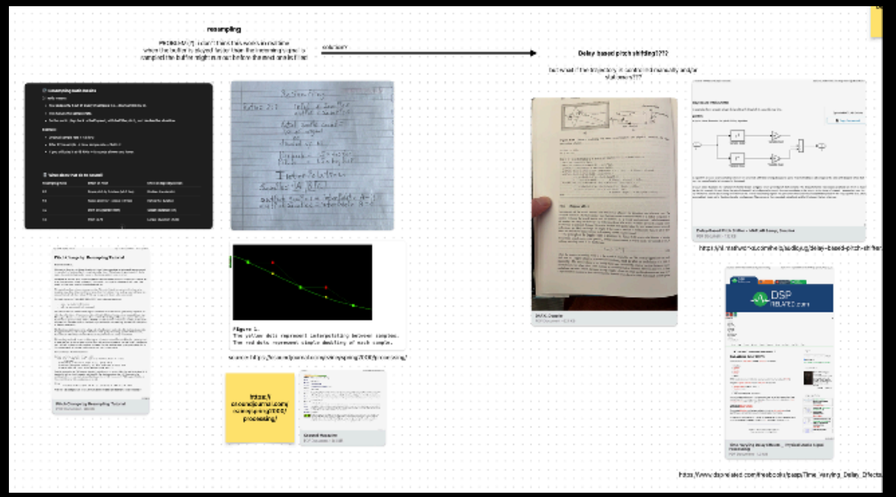


SEMUEL LEIJTEN

LEERDOELEN + REFLECTIE

1. ik kan zelfstandig onderzoek doen naar onderwerpen die ik nog niet begrijp





- hoe werkt het Doppler effect?
  - formule, wat zijn variabelen?

- pitch shifting: verschillende methodes onderzocht

- uiteindelijk op delay uitgekomen
  - doet pitch shifting zelf
  - hoe van afstand naar delay-tijd?

LEERDOELEN + REFLECTIE

2. ik begrijp in de basis de wiskunde achter het Doppler effect en kan dit toepassen in C++

- had de formule die ik als eerst vond uiteindelijk niet nodig

- stelling van Pythagoras om afstand te berekenen

- van afstand naar delay-tijd

$$f' = f\,(V \pm V_0) \,/\, (V \pm V_s)$$

Here:

- f' is the observed frequency
- f is the actual frequency
- V is the velocity of the waves
- V_0 is the velocity of the observer
- V_s is the velocity of the source

```
73
74  float Doppler::calculateDistance(float objectPositionMeters) {
75      /* Pythagoras go brrrrrrrrr */
76      const float a = m_passByDistanceMeters;
77      const float b = objectPositionMeters;
78      const float c = std::sqrt( lcpp_x: a*a + b*b);
79
80      return c;
81  }
```

```
82
83  void Doppler::calculateDelayTimeMillis() {
84      /*  delay = distance / speed of sound * 1000  */
85      const float delayTimeMillis = m_distanceMeters/m_speedOfSound * 1000.0f;
86      m_delay.setDelayTimeMillis(delayTimeMillis);
87  }
```

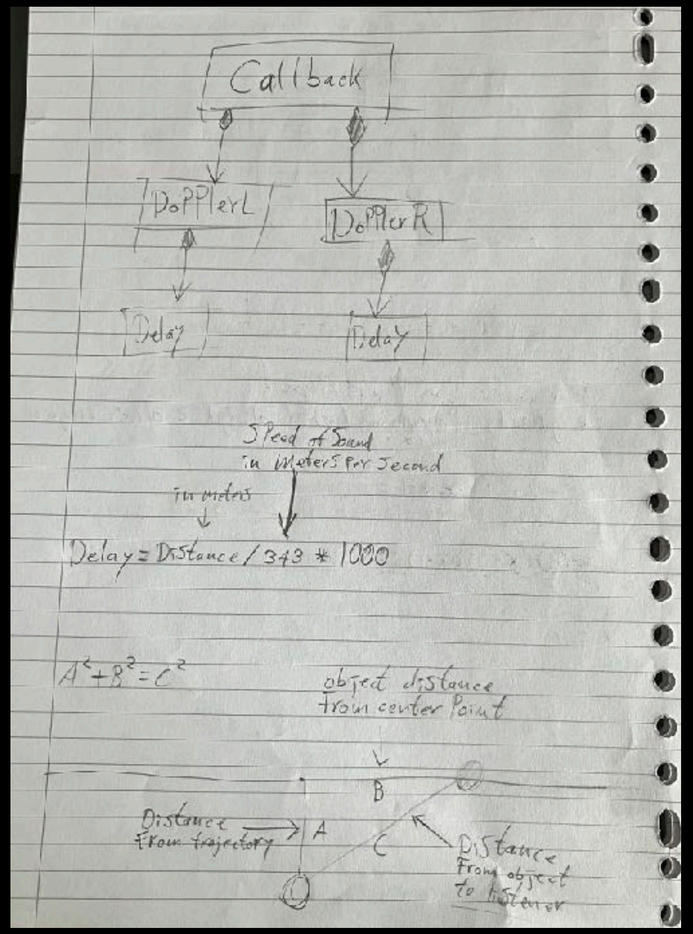
LEERDOELEN + REFLECTIE

3. ik kan een concept uitwerken tot een artefact met eigen gestelde kaders en planning

- kaders
  - 1 slider voor user, mogelijk enkele andere variabelen
  - pitch shifting
  - panning
  - fade-in/out

- delay duurde langer dan verwacht
  - raakte achter op planning
  - veel psuedocode geschreven
  - begrijp interpolatie nu beter

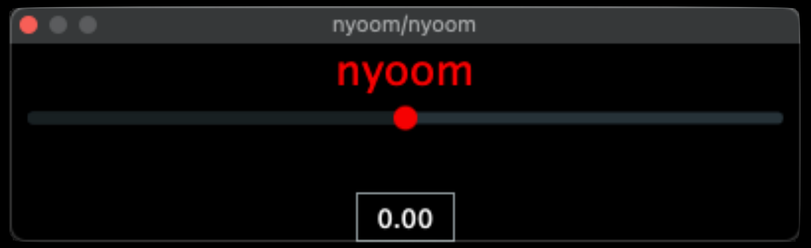
- de rest ging sneller
  - toch werkend prototype aangeleverd



LEERDOELEN + REFLECTIE

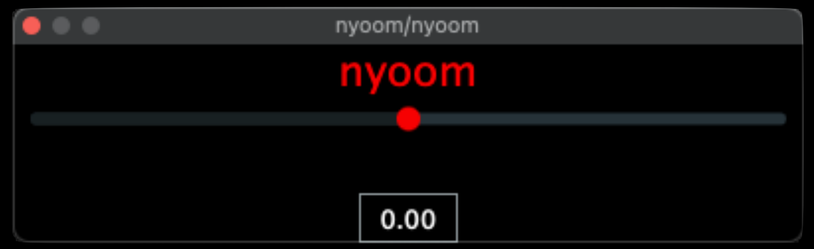
4. ik kan met JUCE plugins met een simpele GUI maken

- is ook gelukt 👍
- begrijp nu in de basis hoe dit werkt
- zie voor me hoe ik kennis kan verbreden



EINDOPLEVERING EN VERVOLGSTAPPEN

- tevreden met prototype
  - gekozen om bepaalde dingen achterwege te laten ivm tijd
  - eerst meest simpele versie maken en daarna verbeteren werkt goed voor mij



vervolgstappen:

- panning: lineair -> equal power panning
- fade-in/out: lineair -> volume berekenen aan de hand van afstand
- filter
- reverb (misschien)
- pass by distance variabel (als dat iets toevoegt)

URENGEWERKT
(buiten les om)

ongeveer 40 uur

SEMUEL LEIJTEN

vragen?