Reflectie **Eindopdracht CSD2B**

Semuel Leijten

Mijn insteek voor deze opdracht was om het relatief simpel te houden en me vooral bezig te houden met het schrijven van een efficiënt en overzichtelijk programma. Mijn programma heeft een synth met en super saw en een synth met een square bass, beide hebben ook de optie om gebruik te maken van bit reduction als extra toevoeging.

De super saw synth heeft een vector die in de constructor gevuld wordt met 5 saws. Iedere saw heeft een hard coded detune value die vermenigvuldigd wordt met een getal dat door de gebruiker gekozen wordt. Hierdoor kan de gebruiker de mate van ‘detuneness’ kiezen, het nadeel is dat dit nu alleen kan tijdens het aanmaken van de synth en daarna staat het vast.
Ik heb met behulp van ChatGPT geprobeerd de meest efficiënte manier om de vector te vullen te vinden. Voordat ik de reserve en emplace_back functies gebruikte viel het me op dat de vector gelijk gevuld werd met saw die vervolgens meteen destructed worden en weer opnieuw constructed maar dan met de juiste detune value. Dit leek me niet efficiënt en zo ben ik hierop uitgekomen. Hierbij gaf ChatGPT ook de suggestie van de for loop die itereert aan de hand van de array ‘detuneValues’, bijna Python zo.

De square bass synth is wat simpeler en heeft een square waarvan de frequentie door 4 gedeeld word om de melodie zo 2 octaven lager af te spelen en een sine waarvan de frequentie door 8 gedeeld wordt zodat die dus 3 octaven lager is en als sub functioneert.

De bitcrusher (of eigenlijk bit reductor) is een member van de synth base class, hierdoor kunnen beide synths er gebruik van maken. Ik heb hiervoor wat voorbeelden op internet bekeken, maar vond deze al complexer dan wat ik in gedachten had en ze leken moeilijk te implementeren. Hierop heb ik ChatGPT gevraagd om een voorbeeld van een zo simpel mogelijke bitcrusher te geven en die heb ik geïmplementeerd.
De bitcrusher kan ook alleen ingesteld worden bij het aanmaken van de synth. Om te checken of deze bypassed is of niet wordt in de getSamples() functies van beide synths een functie aangeroepen die de huidige sample door de bitcrusher halen als deze niet bypassed is. Dit is een enigszins pragmatische oplossing aangezien de condition nu iedere sample gecheckt wordt ondanks dat deze terwijl het programma draait nooit zal veranderen. Voor nu werkt dit, maar een efficiëntere oplossing was waarschijnlijk mogelijk geweest.

Ik heb vanaf het begin geprobeerd om modulair te werken, ik merkte dat dit er ook aan bijdroeg om het overzicht te bewaren in een uitgebreid programma. Het is fijn om delen van mijn programma die goed werken te kunnen afsluiten en me te focussen op andere dingen.
Afgezien van de hierboven genoemde voorbeelden en enkele anderen heb ik voor deze opdracht eigenlijk minder gebruik gemaakt van ChatGPT, ik merkte dat ik vaak erg complexe antwoorden kreeg waar eigenlijk een vrij simpele oplossing mogelijk was. Ook kon ik dit blok steeds meer aan één stuk door programmeren zonder hulp te hoeven inschakelen. Ik heb me tegen het einde ook bezig gehouden met het wegwerken van warnings, ik kreeg bijvoorbeeld warnings over het feit dat ik signed integers gebruikte in for loops. Hierop kreeg ik van Daan de oplossing om unsigned integers te gebruiken. Waarschijnlijk nog niet nodig voor dit blok, maar ik vind het leuk om dingen zo efficient mogelijk te maken, daarom zitten er hier en daar een aantal dingen zoals dit die ik zelf heb uitgezocht.

Urenoverzicht
(ongeveer)

Gemiddelde uren zelfstudie per week: 8
Totaal dit blok (schatting): 46
Sinds moment dat ik werkende oscillatoren had (schatting): 25