Laboration 4 – GenAI för ljuddesign

I denna laboration ska ni använda ljudprogrammeringsmiljön SuperCollider (SC). SC är en realtidssyntesmiljö med stora möjligheter att koda ljudsyntes och signalbehandling. SC finns för Mac, Linux och Windows och kan laddas ned här:
https://supercollider.github.io/

Uppgiften

Uppgiften går ut på att undersöka möjligheterna med och använda GenAI för att skapa grunden för ljud som sedan procedurellt kan justeras och anpassas. Ljudet/ljuden ska sedan användas i ett gränssnitt för en termometer och ska kunna representera temperaturen (mellan -20 och +40 grader), samt tre typer av väder (solsken, regn, och snöfall).

Företaget som tillverkar termometern marknadsför stenhårt att de är så moderna att de använder AI för att skapa ljuden. Men tyvärr har inte termometern processorkraft nog för att använda AI i realtid för att skapa alla varianterna av ljud, utan i stället används procedurella metoder för att förändra ljudet som AIn skapat efteråt.

Er uppgift är att använda AI för att skapa ljud som sedan procedurellt kan förändras till termometern.

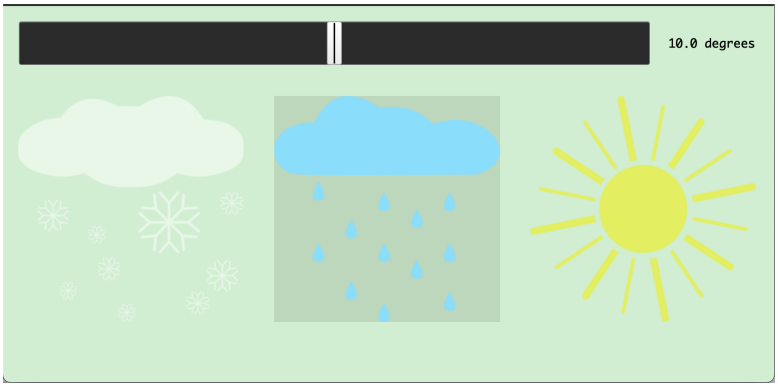
Funktionen i ljuden ska vara:

1. Temperaturen ska sonifieras av ljudet som ni skapat grunden till med AI.
2. De tre väderförhållandena ska ljudsättas, även här grundat i det/de ljud som skapats av AI.
3. Ljuden, och de procedurella ljudförändringarna, för temperaturen och väderförhållandena ska fungera tillsammans och skapa en helhet, ett soundscape, som ger en användare tillräcklig information för att förstå klimatet utomhus just nu.

Inför laborationen

Förberedelse – 1. Tänk på vilket ljud som ska användas

Fundera på funktionaliteten i gränssnittet och vad som ska kopplas till ljud. Bestäm er också för vilken typ av ljud och vilken typ av ljudförändringar som ni tycker passar för ändamålet.



Sedan, sök efter möjliga GenAI-verktyg för ljud. De flesta AI-verktyg för ljud fokuserar på text-till-tal eller tal-till-text, eller olika typer av ljud- och hörbarhetsförbättringar. En stor andel av AI-verktygen riktar sig också mot musikindustrin, där musikinstrument i en mix isoleras, eller sångröster görs om efter andra sångare. Det finns dock några som kan skapa ljud från en textprompt, exempelvis:

Följande är några tjänster som kan skapa ljudeffekter från en textprompt:
https://poppop.ai/ai-sound-effect-generator
https://ai.vadoo.tv/ai-tools/ai-sound-effect
https://www.aisoundfx.com/generate
https://elevenlabs.io/sound-effects
https://www.clipmove.com/ai-sound-effect-generator
Följande verkar dessvärre vara spärrad av LiU IT:
https://myedit.online/en/audio-editor/ai-sound-effect-generator

GenAI works ser ut att kunna bli ett intressant verktyg:
https://genai.works/app/audiogen/

ElevenLabs är ett annat verktyg:
https://elevenlabs.io/

Följande är intressant, men verkar mest skapa låtar:
http://suno.com/
https://www.udio.com/

Meta AudioGen, har en intressant GenAI, men verkar vara svår att skriva promptar till:
https://audiocraft.metademolab.com/audiogen.html
https://about.fb.com/news/2023/08/audiocraft-generative-ai-for-music-and-audio/

Förberedelse – 2. Bekanta er med koden som ges i zip-filen

Synthdefinitioner

Koden har en kort/liten synthdefinition (`temperatureSynth`) som från början innehåller en sinusoscillator (`SinOsc.ar`) i 220Hz och med en ljudvolymsförändring (`level`). Ljudvolymen är kopplad till att musknappen är nedtryckt på slidern, och tonhöjden är kopplad till var på slidern man är, från 110Hz till 880Hz. Ljudvolymen i sin tur ändras med hjälp av `lag(0.5)`, för att få en lite mjukare övergång mellan ljud och inget ljud. Den här synthdefinitionen är bara som en enkel demo, ni kommer att använda ljudsamplingar från Ain.

När ni löser uppgiften så skapar ni fler synthdefinitioner. Hur många beror på hur ni väljer att lösa uppgiften.

Klientsideskript

På klientsidan har jag försökt sortera koden, men de viktigaste delarna för er kommer en bit ner från rad 130.

Här kommer först två funktioner för när musknappen klickar (och sen sluter vara nedtryckt) på slidern, `~slider.mouseDownAction` och `~slider.mouseUpAction`. Det enda som görs här i grundkoden är att sätta `level` på `temperatureSynth`.

Därefter följer tre funktioner som ska användas för att justera ljudet och återspegla väderförhållandena, `~sunAction`, `~rainAction` och `~snowAction`. Endast ett av väderförhållandena kan gälla åt gången, men alla tre kan vara avstängda samtidigt. Just nu skrivs bara läget på väderförhållandena ut i `Post window`, men dessa utskrifter kan tas bort sen.

Dessa följs sedan av en funktion som justerar synthdefinitionerna (i början bara `temperatureSynth`) i relation till sliderns position. Först deklareras två variabler som sneare används för att justera GUIt, sedan deklareras och mappas variabeln `temperature`. Den variabeln används sedan och mappas för att justera tonhöjden på `temperatureSynth`. Därefter justeras GUIts bakgrundsfärg och texten som visar temperaturen uppdateras.

Skapa kopplingar mellan data och ljud

När ni har valt en GenAI, behöver ni utforska hur ni bäst formulerar en prompt som ger de ljudresultat som ni önskar. Orden är viktiga, och de måste reflektera den ljudidé som ni har. Ni kommer säkert behöva flera försök och flera olika formuleringar innan ni hittar ett ljud ni är nöjda med. Tänk på att det kan behövas flera ljud som ni mixar ihop efteråt för att skapa en helhet. För att mixa ljudet kan ni använda SuperCollider, men det går också att använda andra ljudredigeringsprogram som (Adobe Audition eller Audacity, eller för all del Matlab). Fundera bara innan ljuden mixas ihop så att de inte behöver vara åtkomliga enskilt i SuperCollider för att skapa mappningen mot datan.

De GenAI-genererade ljuden kan/bör/ska sedan procedurellt anpassas och förändras i SuperCollider. I den följande texten finns en del tips och idéer, men kolla tillbaka till laborationstexterna till de tidigare laborationerna, och använd dessa som stöd och underlag för att förändra ljuden. Det går utmärkt att kombinera olika GenAI-genererade ljud för att skapa rätt intryck, men det kan också behövas mixas in en del procedurellt genererade ljud i olika vågformer eller i form av brus.

Modulationer

Amlitudmodulation (kallas också tremolo) kan vara användbart och både tempot i modulationen och modulationsdjupet kan vara intressanta att utforska. Exempelvis följande kod

```
var temperatureSound;
var modulationSpeed = 1;
var modulationDepth = 0.75;
var amplitudeModulation = LFPulse.kr(modulationSpeed).range(modulationDepth, 1);
var output = temperatureSound * amplitudeModulation;
```

I denna kod kan hastigheten i amplitudmodulationen styras med `modulationSpeed`, och det kan vara intressant att undersöka 0.1Hz till 10Hz eller mer. Modulationsdjupet styrs i sin tur av `modulationDepth` och kan varieras mellan 0, och då är ljudet helt tyst när vågformen är låg, upp till 0.95, då ljudet är nästan lika starkt hela tiden.

Det går också att använda andra vågformer än fyrkantsvåg för att skapa andra intryck av modulationen.

Frekvensmodulation (eller vibrato) är också intressant att använda, och även i detta fall kan man manipulera modulationsfrekvensen och modulationsdjupet. Exempelvis:

```
var modulationSpeed = 1;
var modulationDepth = 2;
var frequencyModulation = SinOsc.kr(modulationSpeed).range((-1 *
modulationDepth), modulationDepth);
var output = SinOsc.ar(freq: frequency * frequencyModulation);
```

I denna kod modifieras frekvensen (`frequency`) av en sinuston med `frequencyModulation`. Modulationsdjupet sätts med `modulationDepth` (som är en variation över en oktav upp och en oktav ner när den sätts till 2). Även i detta fall kan man använda andra vågformer för att skapa andra ljudintryck.

Harmoni

Harmoni kan också vara intressant att experimentera med. Om skapas helt procedurellt är det enkelt att göra harmoniska förändringar. Exempelvis

```
var detuneAmount = 0.01;
var detune = frequency * detuneAmount;
var sine1 = SinOsc.ar(freq: frequency + detune);
var sine2 = SinOsc.ar(freq: frequency - detune);
var output = sine1 + sine2 / 2;
```

I denna kod styr `detuneAmount` hur mycket detuning (hur falskt det ska bli) och variabeln `detune` ändrar detta till frekvens (Hz). Därefter stäms två sinusoscillatorer åt två håll med `+` och `-`, därefter mixas båda ljuden ihop och ljudnivån halveras för att undvika distorsion.

Detta går naturligtvis lika bra att göra med andra vågformer också. När två ljud nästan stämmer i tonhöjd uppstår en beating. Denna beating, rytm eller puls i amplitud, går långsammare desto bättre de två tonerna stämmer med varandra medan rytmer ökar i tempo ju falskare tonerna blir tills det till slut tydligt hörs att det är två olika toner.

Det går också att göra en liknande ansats för två samplade ljud. Då är det inte frekvensen i sig, utan `rate` som justeras på motsvarande sätt. Det innebär att det måste skapas två `PlayBuf.ar` med olika variation av rate. Exempelvis:

```
var detune = 0.01;
var sound1 = PlayBuf.ar(
   numChannels: 2,
   bufnum: bufnum,
   rate: BufRateScale.kr(bufnum) * (1 + detune),
);
var sound2 = PlayBuf.ar(
   numChannels: 2,
   bufnum: bufnum,
   rate: BufRateScale.kr(bufnum) * (1 - detune),
);
var output = (sound1 + sound2) / 2;
```

I denna kod motsvarar `detune` procenten (`%`) av originalfrekvensen (tonhöjden).

Frekvensinnehåll och filtrering

Glöm inte bort möjligheterna till att förändra frekvensinnehållet i ljudet med olika typer av filter. Testa att använda lågpass- och högpassfilter (`LPF.ar` och `HPF.ar`), eller bandpass-

eller bandstoppfilter (`BPF.ar` och `BRF.ar`). Utforska resonansen i filtren (`rq` eller `reciprocal of Q`) som alltid finns tillgängligt i bandpass och bandstop, men som också finns i de resonanta lågpass- och högpassfiltren (`RLPF.ar` och `RHPF.ar`).

Mappning

Kom ihåg att utforska mappningarna som finns inbyggda i SuperCollider, som linjär till linjär skalning (`linlin`) och linjär till exponentiell skalning (`linexp`). Men det går förstås utmärkt att skriva egna mappningar.

If-satser är bra att använda, men i SuperCollider finns inte else if. I stället för ni nästla if-satser.

```
if (x >= 10) {
   // do something
} {
   // else do someting
};
```

En nästlad if-sats med en motsvarande else if skulle då se ut:

```
if (x >= 10) {
   // do something
} {
   if (x < 10) {
      // do something
   } {
      // else do something
   }
};
```

Det finns olika sätt att lösa nästlade if-satser som `switch` och `case`. Se mer här: https://depts.washington.edu/dxscdoc/Help/Reference/Control-Structures.html

Lag är ett utmärkt sätt att lägga fördröjningar eller långsammare förändringar av mappningar i synthdefinitionen (`amplitude.lag(1)` till exempel, vilket ger en förändring av värdet i variabeln `amplitude` en långsam övergång på en sekund).

Mer ljuddesignmöjligheter

Titta i extramaterialet till laboration 2 (la2_extra.pdf) för fler möjligheter för att modulera och designa ljudet, och förstås också laborationshandledningarna till laboration 1, 2 och 3.

Och för olika typer av brus, läs mer här:
https://en.wikipedia.org/wiki/Colors_of_noise

SuperCollider har möjligheten att skapa vitt, rosa, brunt, och grått brus. Men, det finns också olika varianter av brus som är möjliga att använda för att skapa intressanta ljud och kontrollmöjligheter. Sök på `noise` i SuperCollider för mer information.

Liksom bruset har SuperCollider många olika möjligheter för att skapa vågformer. Förutom sinus, triangel, fyrkant och sågtand, kan man använda formantoscillator, gaussisk funktions generator, impulsoscillator, eller klang som är en sinusoscillatorbank. Sök på `oscillator` i SuperCollider för mer information.

Det finns också olika typer av envelopegeneratorer att använda. `Perc` och `ADSR` är två vanliga, men det går också att använda `linen` (som är en linjär envelop), `triangle` och `ASR` (`Attack`, `Sustain`, `Release`) för att nämna några. Sök på `Env` i SuperCollider för mer information.