Digital lyd: Afsluttende opgave

Skillevej: En algoritmisk komposition

Af Sebastian Rostved
20/12-2017

34300 anslag / 14 normalsider

Table of Contents

Baggrund

I den afsluttende opgave i Digital Lyd, har jeg produceret et generativt musikalsk værk i det objekt orienterede programmeringssoftware Max 7. Ud fra et selvvalgt billede, som jeg beskriver senere, har jeg udarbejdet et koncept, samt fået musikalsk inspiration. I Max har jeg arbejdet med algoritmiske principper, som i sidste ende har resulteret i et stykke generativt musik. Lyden er hovedsageligt genereret ved hjælp af subtraktiv syntese, men undervejs har jeg også arbejdet med granulær syntese, additiv syntese og bearbejdning af samplet lyd.

Formålet med denne opgave er at beskrive min arbejdsproces, samt de overvejelser og valg som jeg har truffet undervejs, i udarbejdelsen af min Max-patch. Undervejs vil jeg bruge begreberne algoritmisk musik og generativ musik i flæng, da begge begreber beskriver alle systematisk regelbaserede fremgangsmåder til komposition af musik (Roads, 2015, s. 339).

Som start vil jeg beskrive algoritmisk/generativ musik teknisk og historisk. Herefter vil jeg forklare mit valg af billede og beskrive billedet, og forklare hvordan billedet relaterer sig til mit koncept, som jeg også vil gå i dybden med.

De forskellige algoritmiske og musikalske dele af mit patch vil jeg beskrive kort, hvorefter jeg vil gennemgå dem teknisk og æstetisk. Visse dele kræver, at man forklarer de teknikker der er brugt inden, og derfor vil jeg kort gennemgå algoritmisk musik, subtraktiv syntese, additiv syntese og granular syntese. I de tekniske afsnit, forekommer der ofte ord med firkantede klammer udenom. Klammerne bruges til at vise, at der er tale om et objekt i Max 7, som ordet indeni henviser til. For eksempel henviser [metro 4000] til metro-objektet i Max, med en tidsparameter sat til 4000 millisekunder.

Efter at have gennemgået de algoritmiske og musikalske dele i min patch, samler jeg op på de forskellige dele og har et afsnit om det samlede udtryk, og hvordan de forskellige dele hænger sammen. Dette leder til en generel æstetisk og historisk perspektivering, som ender i en konklusion.

Om algoritmisk musik og Max
I det følgende afsnit vil jeg forklare hvad begrebet algoritmisk musik dækker over, og kort beskrive udvalgte historiske eksempler på brug af algoritmer til komposition af musik. Algoritme er oprindeligt et matematisk begreb, som dækker over en proces eller et sæt af regler, som bruges i udregninger og problemløsninger. I forhold til musik og komposition, bruger man begrebet algoritme til at beskrive regelbaserede processer og modeller, som kan være mere eller mindre automatiserede. Disse modeller bruges til at skabe musik og styre forskellige musikalske parametre som for eksempel tonehøjde, amplitude og klangfarve.

Algoritmisk musik kan spores tilbage til oldtiden, hvor regler om tonalitet og kanon blev nedskrevet. I Barokmusik og første wienerskole har mange komponister arbejdet med algoritmiske metoder, blandt andet J. S. Bach med *Goldbergvariationerne* og J. A. Mozart med *Musikalisches Würfelspiel* (Essl, 2007, s. 110). Oversat fra Tysk, betyder det musikalsk terningespil. Kompositionen var delt op i brudstykker, som alle vilkårligt kunne sættes sammen til et sammenhængende stykke. Rækkefølgen blev afgjort ved tilfældighedsprincipperne i terningekast med to terninger.

I starten af 1920'erne udarbejdede komponisten Arnold Schoenberg en metode til komposition, hvor alle tolv toner var ligevægtige. Dette markerede starten på anden wienerskole. I 1940'erne videreudviklede den østrigske komponist Josef Matthias Hauer Schoenbergs tolvtone-teknik til en algoritme, som dannede grundlag for en harmonisk model til komposition. Hauer dannede, ud fra sin algoritme, vilkårligt tolv firklangsakkorder, som han byggede melodiske figurer og arpeggios oven på (Essl, 2007, s. 110).

Efter 2. verdenskrig begyndte flere komponister, heriblandt Anton Webern, at arbejde med serialisme, som tog udgangspunkt i Arnold Schoenbergs metoder, men udvidede dem til at gælde andre parametre som nodelængde, dynamik o.a. Denne form for algoritme kaldes *deterministisk*, da den for eksempel involverer et fastlagt sæt af toner, som komponeres efter et antal bestemte metoder eller regler, som ikke involverer tilfældighed. Et tidligere eksempel på determinisme er J. S. Bachs koraler, hvor der var et fastsat sæt regeler for melodi og harmoni (Roads, 2016, s. 346).

Ligesom den første wienerskole, som arbejdede med terningekast i deres kompositioner, begyndte blandt andet Iannis Xenakis i 1950'erne at arbejde med tilfældighedsalgoritmer i sine kompositioner (Essl, 2007, s. 115). Denne type algoritme til komposition, kaldes *stokastisk*. Stokastiske processor kan analyseres statistisk og er styret af sandsynlighedsprincipper, men resultatet eller udfaldet kan aldrig udregnes præcist. Xenakis var også med til at programmere stokastiske computerprogrammer, til komposition af algoritmisk musik.

Senere har Karlheinz Stockhausen og andre, arbejdet med at kombinere stokastiske- og deterministiske principper og algoritmer i deres værker, John Cage med kinesiske stokastiske algoritmer fra oldtiden, og Brian Eno med at afspille båndloops, forskudt af hinanden, med forskellig længde. (Essl, 2007, s. 120-121)

I 1960'erne byggede Institute for Sonology Utrecht en modular synthesizer, som ved hjælp af analoge synthesizermoduler, kunne bruges til at komponere generativ musik (Essl, 2007, s. 122). Modulerne blev koblet sammen og påvirkede hinanden igennem strømførende kabler. Nogen moduler genererede toner, andre modulerede lyden og nogen moduler styrede parametre på de andre moduler. Computerprogrammet Max, som mit stykke generative musik er komponeret i, minder meget om denne tilgang til lyd- og musikproduktion, hvor objekterne i max kan sammenlignes med synthesizermoduler, og begge kobles sammen med ledninger (Breinbjerg, 2006, s. 32).

Billede og konceptualisering: Skillevej

Billedet som mit værk relaterer sig til, forestiller en vej. Det er nat, og vejen er belyst af gadelamper, som ikke er synlige på billedet. Man kan dog ane en pæl. Omgivelserne til vejen er meget svære at tyde, da billedet er uskarpt, grynet og i sort/hvid. Umiddelbart kan man se at vejen ligger på et lille dige, men om omgivelserne er sne eller græs, er ikke til at se. På grund af lyset, er det også svært at se om vejen deler sig i to i forgrunden, hvor den laver et knæk, men det går jeg ud fra i mit koncept.

Mit koncept tager udgangspunkt i den formodning om at vejen deler sig i to, og at den dermed er en skillevej. I Den Danske Ordbog bliver ordet *skillevej* i overført betydning beskrevet således:

> *situation hvor man står over for et afgørende valg, eller hvor udviklingen tager en ny retning*[1]

En *skillevej* er derfor et sted, hvor der bliver foretaget et valg, og hvor at der sker noget nyt. Hvis man skal overføre det til generativ musik og programmering er der en klar parallel til begrebet og teknikken *selektion*, som Morten Breinbjerg beskriver i Musikkens Interfaces (2006, s. 35):

> *Selektion betyder at programafviklingen kan følge flere veje. Selektionen åbner for forgrening og dermed for forskellige udfald.*

Her kan man se parallelen imellem de to udtryk. Begge beskriver et udgangspunkt, hvorfra der er to eller flere veje som man kan ”gå” ned ad, og hvert valg vil have forskellige konsekvenser eller udfald. I softwareprogrammering vil dette ske når en forudbestemt betingelse er opfyldt, hvorefter koden vil angive hvad der videre skal ske. Dette kan for eksempel være med en if-sætning, som jeg vil beskrive senere. Derfor besluttede jeg at mit projekt skulle have elementer, som bliver styret af selektive processer, herunder if-sætninger og dets lige. Selve kompositionen og forløbet skal derfor kunne ændre sig markant, hvis visse betingelser er opfyldt, eller hvis der bliver foretaget en bestemt handling. Konceptet skal også indeholde musikalske elementer og repræsentere dele af billedet lydligt og kompositorisk. For eksempel gadelamperne, billedets kvalitet o.l.

Algoritmerne

I Composing Electronic Music af Curtis Roads (2015, s. 338), skriver Roads i indledningen:

> *For centuries, composers organized music compositions according to an evolving set of rules governing harmony and counterpoint.*

Curtis Roads mener, at man kan sammenligne generativ og algoritmisk musik med de regler om tonalitet, som komponister og musikere har brugt til at skrive musik med, i de sidste mange hundrede år. Algoritmisk musik kan spores helt tilbage til den tidlige middelalder, hvor instruktionerne til kanon blev nedskrevet. Den samme kanon, som senere blev videreudviklet af Johann Sebastian Bach (Essl, 2007, s. 109). Derfor synes jeg, at det kunne være interessant at undersøge, hvilke muligheder der er, for at lave en algoritme som selv styrer kandence, tonalitet og modulation, imellem toneskalaer i et musikalsk stykke. Ved hjælp af selektion og andre teknikker, vil man kunne lave en form for digital akkordmaskine i Max ud fra et sæt givne musikteoretiske regler. På denne måde, håber jeg, at man kan koble noget af det mest basale inden for vestlig musik, som samtidig har sit eget algoritmiske islæt, sammen med moderne programmering med selektionsalgoritmer i Max, for at skabe en sammensmeltning eller symbiose af algoritmiske traditioner.

1 http://ordnet.dk/ddo/ordbog?query=skillevej

Akkordalgoritmen

Kiefer og Riehl (2016), som matematisk har analyseret mange hundrede værkers akkordprogression fra Palestrina, Bach, Mozart og Beethoven, er kommet frem til følgende markovkæder, som beskriver Bach og Mozarts akkordprogressioner i sandsynligheder.

Bach Minor	I	ii	iii	IV	V	vi	vii°
I	0	.18	.01	.20	.41	.09	.12
ii	.01	0	.03	0	.89	0	.07
iii	.06	.06	0	.25	.19	.31	.13
IV	.22	.14	0	0	.48	0	.15
V	.80	0	.02	.06	0	.10	.02
vi	.03	.54	.03	.14	.19	0	.08
vii°	.81	0	.01	.03	.15	0	0

Bach Major	I	ii	iii	IV	V	vi	vii°
I	0	.15	.01	.28	.41	.09	.06
ii	.01	0	0	0	.71	.01	.25
iii	.03	.03	0	.52	.06	.32	.03
IV	.22	.13	0	0	.39	.02	.23
V	.82	.01	0	.07	0	.09	0
vi	.15	.29	.05	.11	.32	0	.09
vii°	.91	0	.01	.02	.04	.03	0

Mozart Minor	I	ii	iii	IV	V	vi	vii°
I	0	.08	0	.07	.68	.06	.11
ii	.37	0	0	0	.46	0	.17
iii	0	0	0	1	0	0	0
IV	.42	.10	0	0	.39	0	.09
V	.82	0	0	.05	0	.07	.05
vi	.14	.51	0	.16	.05	0	.14
vii°	.76	.01	0	0	.23	0	0

Mozart Major	I	ii	iii	IV	V	vi	vii°
I	0	.13	0	.15	.62	.05	.05
ii	.49	0	.01	0	.40	.01	.09
iii	.67	0	0	0	0	.33	0
IV	.64	.14	0	0	.15	0	.07
V	.94	0	0	.01	0	.04	.01
vi	.11	.51	0	.14	.20	0	.04
vii°	.82	0	.01	.01	.16	0	0

En markovkæde, opkaldt efter den russiske matematiker Andey Markov, er en stokastisk matematisk metode til at udregne sandsynligheden for, at noget går fra et stadie til et andet. I Kiefer og Riehls (2016) diagrammer, er akkordtrinene på y-aksen det indledende stadie og akkordtrinene på x-aksen det næste mulige stadie. Hvis man for eksempel tager udgangspunkt i første trin i tabellen (angivet med I), som viser sandsynlighederne i Bachs mol skalaer, kan man se at der er 41% sandsynlighed for at det næste er femte (V) trin. Hvis man er på femte trin er der 80% sandsynlighed for at det næste er første trin, og så fremdeles. Den samme fremgangsmåde gælder også for de tre andre tabeller. Jeg valgte at tage udgangspunkt i disse tabeller og bruge dem til min max-patch, for at udregne sandsynligheden for akkordprogressionerne.

Akkordalgoritmen består af to dele, og er en selektions-algoritme, som kører i ring. Den første del danner treklangsakkorder i en a-mol toneskala. Der er i alt syv trin i a-mol toneskalaen, og hvert eneste trin danner en akkord. Skift imellem trinene er udregnet med den tidligere nævnte markovkæde. Når en forudbestemt betingelse er opfyldt (sjette trin i a-mol skalaen bliver spillet), stopper den første del af akkordmaskinen, og der skiftes til den anden del, som er magen til den første, men i g-dur. Musikteoretisk er der tale om en modulation. Der er også mulighed for at algoritmen skifter/modulerer tilbage til a-mol, hvis den samme betingelse som før er opfyldt. A-mol og g-dur skalaerne er arrangeret på følgende måde:

A-mol

Trin	I	ii	iii	IV	V	Vi (Mod. til g-dur)	vii^{o}
Akkord	Am	Bm	C	Dm	Em	F	G^{o}

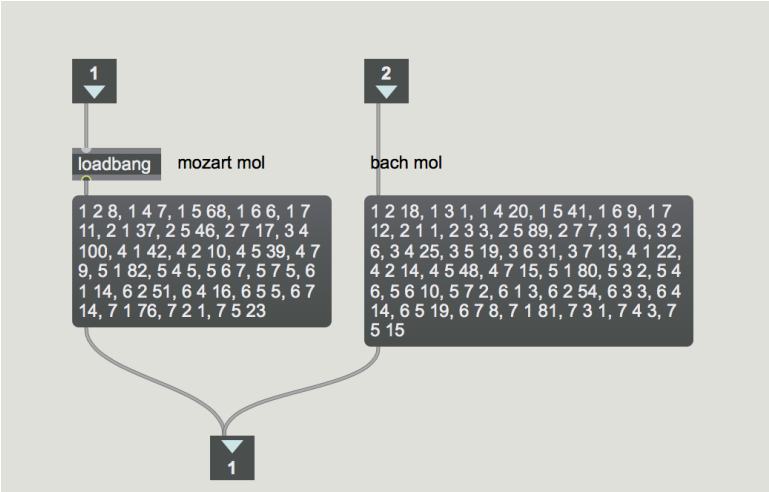
G-dur

Trin	I	ii	iii	IV	V	Vi (Mod. til a-mol)	vii^{o}
Akkord	G	Am	B	C	D	Em	F#^{o}

Teknisk
I det følgende afsnit, vil jeg beskrive den tekniske del af akkordmaskinen, og gennemgå de vigtigste dele. I Max ligger akkordmaskinen i et [patcher] eller subpatcher objekt, som er en underpatcher til forsiden på hele Max-patchen. Subpatcheren hedder [p akkordmaskine]. En patch i Max, er et dokument som består af et kanvas, hvorpå man kan ligge sine objekter og forbinde dem med cords (da. Ledninger). [p akkordmaskine] objektet er opbygget af to dele. Den første del vil jeg starte med at beskrive her:

Akkordmaskinen tændes med et toggle objekt på forsidepatcheren. Toggleobjektet er forbundet til et input på [p akkordmaskine] og signalet sætter gang i et [metro] objekt, som sender et bang ud hver fjerde sekund (4000 ms.). Det ene objekt som bliver påvirket af [metro], er objektet [prob]. [prob] er et objekt, som udregner sit output ud fra parametre, som man giver den. I dette tilfælde, er det de førnævnte markovkæder, som ligger gemt i en subpatcher som hedder [p markov_amol].

Markovkæderne som den første del styres af, er dem som er baseret på data fra Mozart og Bach. I subpatcheren ser det således ud, når informationerne er plottet ind i en message box.



Markovkæderne giver mulighed for [prob] at skifte imellem syv forskellige akkorder, som er de syv trin i en a a-mol-skala. [prob] sender for hvert bang den modtager fra [metro] et tal ud, som er udregnet fra markovkæden. Udover, at tallet bliver sendt videre ud i systemet, bliver det også sendt videre til et [switch] objekt. [switch] har syv forskellige indgange, og åbner én indgang ad gangen, som er den indgang, der svarer til det nummer som [switch] modtager. Så hvis [switch] modtager et femtal, åbner den indgang fem, indtil den modtager et nyt tal. Inputtet fra den åbne indgang, sendes videre ud af den ene udgang på [switch]. Hver eneste indgang på [switch] er koblet til en udgang på en et subpatcher object som hedder [p amol].

[p amol] modtager bangs fra det førnævnte [metro] objekt, som aktiverer 21 [random 5] objekter i [p amol]. Hvert [random 5] objekt udsender et vilkårligt tal imellem 0 og 4 for hvert bang, som igen aktiverer et tal fra en liste med midi node numre. De 21 lister er delt op i grupper på tre, som indeholder midinodenumre til treklange i a-mol. Treklangene kan bestå af fem oktaver, så der kan forekomme forskellige omvendinger i akkorderne. Når en treklang er aktiveret og sendt videre fra listerne, pakkes deres midi-data ind, som igen bliver sendt videre i systemet.

Tallet fra [prob], som angiver hvilket trin som maskinen er på i a-mol, bliver også sendt videre til et objekt med en *if*-sætning ([if $i1 == 6 then bang]), som gør at der bliver udsendt et bang, når [prob] sender tallet 6 ud, eller sagt med andre ord: Når akkordalgoritmen er nået til det sjette trin i a-mol-skalaen, sender den et bang ud, som aktiverer den anden del i akkordmaskinen. Et [delay 4000] objekt, gør at sjette trin spilles færdig, før at der skiftes. Den anden del fungerer ligesom den første del, men indeholder mididata til en g-mol skala i stedet, så når den første del af a-mol skalaen når til sjette trin (f-dur), modulerer hele maskinen til en g-dur skala. Når den anden del når til det sjette trin, skifter den også, og modulerer tilbage til a-mol og den første del af akkordmaskinen.

Æstetisk
Da akkordrækkefølgen er baseret på data fra enten J. S. Bach og W. A. Mozart, får tonaliteten og akkorderne et meget ”klassisk” udtryk, forstået på den måde, at første (tonika) og femte (dominanten) trin dominerer i deres musik (Kiefer & Riehl, 2016, s. 21).

> *Our results for Bach and Mozart agree with the Baroque and Classical eras, respectively, because they show the dominance of the I and V chord - where the I chord is most probable and the lack of the iii chord.*

Dog skaber markovkæderne en hvis uforudsigelighed, da akkordprogressionerne er baseret på sandsynlighed. Modulationerne skifter også uforudsigeligt over tid, men da der kun er to skalaer, giver det tilnærmelsesvis et cyklisk udtryk frem og tilbage imellem a-mol skalaen og g-dur skalaen. Som det kan ses i Markovkæderne for henholdsvis Bach og Mozart, er der forskel på deres brug af akkorder. Bach tillader flere forskellige skift imellem de forskellige akkordtrin end Mozart, som man måske vil kunne høre i mit endelige generative værk.

Som beskrevet i mit koncept, anvender jeg selektionsalgoritmer flere gange i akkordalgoritmen, med de mest tydelige i form af if-sætninger til styring af modulation. Switchen kan også tilnærmelsesvis betegnes som en selektionsalgoritme, da den udfører en bestemt handling, hvis dens tilsvarende betingelse er opfyldt. Et eksempel kan være: *Hvis input er lig med 6 så før input 6 til output ellers ignorer input 6*. Akkordalgoritmen er grundet brugen af markovkæder og tilfældighedsgeneratorer udelukkende en stokastisk algoritme.

Nodealgoritme 1 og 2

Nodealgoritmen er en algoritme i mit projekt, hvor flere parametre er styret af akkordalgoritmen. I modsætning til akkordalgoritmen, sender den kun ét midinode nummer ud ad gangen. Formålet med nodealgoritmen er at have en monofonisk (en enkelt stemme) til at supplere akkorderne fra

akkordalgoritmen, for at skabe flerstemmighed med forskelligt tempi, rytme og mulighed for andre klangfarve- og amplitudeindstillinger.

Det tekniske
Nodealgoritmen fungerer på stort set samme måde som akkordalgoritmen. Et metroobjekt med en prædefineret værdi på 500 ms. aktiverer [prob] med et bang hver halve sekund. [prob] har ved opstart af patcheren, modtaget statistiske informationer om sandsynlighed, fra et messageobjekt i form af en markovkæde. Markovkæden for nodealgoritme 1 ser således ud:

Input	Output			
		1	2	3
	1	0	100	0
	2	20	0	80
	3	100	0	0

Outputtet er en opadgående tretoners arpeggio, som i 20% af tilfældene går tilbage til første trin. Dette skaber en smule variation i melodien. Værdierne fra 1-3 bliver herefter sendt ned til en subpatcher [melo-index (1 og 2)], som indeholder indexfiler med midinodenumre for treklange over de syv akkorder i en toneskala. Der er en, til når a-mol er aktiveret, og en, til når g-dur er aktiveret. [switch 7] får information fra akkordalgoritmen om hvilken akkord som bliver spillet, og åbner op for den tilsvarende treklangs data til sit output. To gates sørger herefter for at der kun sendes data jfr. den rigtige toneskala. Dette bliver sendt videre til en synthesizer. Nodealgoritme 2 fungerer på samme måde, men har en nedadgående arpeggio i stedet.

Input	Output			
		1	2	3
	1	0	0	100
	2	100	0	0
	3	0	100	0

Æstetisk
Den første nodealgoritme er udelukkende stokastiske metoder, da dens arpeggio bliver udregnet af en markovkæde. Den anden bruger også en markovkæde, men denne er fastlagt med 100% sandsynlighed, og derfor er den ikke stokastisk. Der bliver også brugt selektion i algoritmen, i form af [switch] og [gate]. Tilsammen giver den opadgående- og nedadgående arpeggio Treklangstonalitet og toneskala er styret fra akkordalgoritmen.

Pads (Gadelamper)

Pads tonalitet er styret fra akkordalgoritmen, og er en polyfonisk (flerstemmig) subtraktiv synthesizer. Pads repræsenterer gadebelysningen i mit koncept, som jeg vil komme ind på i det æstetiske afsnit.

En subtraktiv synthesizer er en synthesizer, med en lydkilde, som er meget rig på overtoner. Dele af lydkilden bliver herefter trukket fra (subtraktiv), og klangfarve og/eller amplituden ændres. Som regel består en subtraktiv synthesizer af en eller flere oscillatorer, hvis lydlige produkt ledes ind i et filter, som kan dæmpe eller forstærke udvalgte frekvensspektra. Man kan kun forstærke eller dæmpe frekvenser som allerede findes i lydkilden (Cipriani & Giri, 2016, s. 309). Til sidst er der som regel en forstærker som kan moduleres fra forskellige kilder. Det kan blandt andet være en funktionsgenerator eller en lavfrekvensoscillator (LFO).

Teknisk
I [p gadebelysning] er lydkilderne seks oscillatorer. Oscillatorerne er delt op i tre stemmer, med to oscillatorer i hver stemme. I hver stemme er der en savtandsbølgeform oscillator (osc 1), og en oscillator, hvor der kan skiftes imellem savtand- og rektangelbølgeformer (osc 2). [saw~] og [rect~] oscillatorerne i Max, er henholdsvis en savtand oscillator og en rektangel oscillator. Bølgeformerne har forskelligt harmonisk frekvensindhold, og de har derfor forskellige klangfarver.

Alle oscillatorer får input om hvilken frekvens de skal spille fra akkordalgoritmen. [mtof] sørger for at konvertere midi-information, til information om frekvens. Osc 1s inputfrekvens bliver lagt til outputtet af et [random 10] objekt, som skaleres til 0 til 3 Hz, og som i sidste ende kan detune signalet på osc 1. Osc 2s bølgeform bliver valgt af en tilfældighedsgenerator ([random 2]) i [p waveform_random]. Dvs. at den skifer tilfældigt imellem en rektangel- og en savtand bølgeform. Signalet fra de tre oscillatorer bliver filtreret i [svf~], som er et multimode filter. I dette tilfælde bliver lowpass-filtret brugt. Et lowpass-filter dæmper frekvenserne gradvist over en bestemt frekvens, kaldet cutoff-frekvensen. Den gradvise dæmpning kan grafisk ligne en bakke, og derfor kaldes den en slope. På en hvis type filtre, heriblandt [lores~], [biquad~] og [svf~], kan man også styre et parameter kaldet Q (Quality), hvor en højere værdi gør slopen stejlere og danner resonans omkring cutoff-frekvensen, hvor en del af de eksisterende frekvenser bliver forstærket (Cipriani & Giri, 2016, s. 314-324).

Lowpass filtrets cutoff-frekvens, bliver moduleret af en simpel attack-decay (AD) funktion, repræsenteret af [function], hvor det andet punkt i funktionen bliver styret af et drunk objekt på x-aksen og et random objekt på y-aksen. [metro 4000] aktiverer de to stokastiske objekter, så de udsender en ny værdi, hver gang der bliver spillet en ny akkord. [drunk 500 100] har en maks-værdi på 500 og den tager et positivt eller negativt trin på 100 hver gang den bliver aktiveret. Signalet bliver herefter skaleret til at ligge imellem 500 og 3500 ms., som gør at attack (1. til 2. punkt) og decay (2. til 3. punkt) ikke bliver for kort. [random 10]s 10 mulige trin, bliver skaleret op til at ligge imellem 500 Hz og 2000 Hz, så cutoff-frekvensens bevægelser kommer til at ligge inden for det hørbare frekvensområde (20-20k Hz)

Efter filtret, bliver signalet delt op i to. På begge signaler bliver amplituden styret af en AD funktioner, hvor det andet punkt er lidt forskudt af hinanden. Da det ene signal bliver ledt ud i

venstre stereokanal, og det andet signal i den højre stereokanal, får lyden en glidende bevægelse fra venstre mod højre. Til sidst sendes hver kanal ned i et rumklangsobjekt, med en kort rumklang (<100 ms.), som har en høj feedbackværdi og mixindstilling, så rumklangen er meget tydelig og relativt lang ift. maximumlængden.

Æstetisk
De subtraktive synthesizerpads, repræsenterer gadebelysningen i mit koncept. Envelopegeneratorer/funktionsgeneratorer ændrer dynamik og filtrering på lyden over tid, med langsomt attack og decay, som kan sidestilles med gadelygterne på en vej, når man kører ned af den, men filtrene når dog aldrig at åbne helt op, og lyden forbliver meget mørk, på trods af enkelte høje toner, da de højere frekvenser altid er filtreret væk. Klangfarverne ændres løbende igennem stokastiske ændringer af dynamiske parametre (Breinbjerg, 2007, s. 31). I patchen ændres parametre som detuning af oscillatorer, ændring af bølgeformer på oscillatorerne og ændringer på funktionernes form i filterets envelopegenerator. Rumklang på lyden gør den mere fjern.

Synth lead (omgivelser)

[p omgivelser] indeholder en meget simpel monofonisk subtraktiv synthesizer, som agerer leadinstrument i den generative patch. Nodeværdier bliver modtaget fra nodealgoritme 1, som er styret af akkorderne fra akkordalgoritmen. Synth lead spiller en opadgående treklangsarpeggio, som i 20% af tilfældene kun spiller to af trinnene i treklangen.

Teknisk
Frekvensinputtet fra nodealgoritme1 bliver inden det når savtandsoscillatoren ganget op med værdi imellem 1 og 3, som betyder at signalet bliver transponeret. [urn 3] genererer en tilfældig værdi imellem 1 og 3, hvor et foregående værdi ikke kan bruges igen. Når alle tre værdier er brugt op, starter den forfra. Derfor transponeres arpeggioen op eller ned hver 2. sekund.

Oscillatorens output bliver ledt igennem et filter, som moduleres af en AD-funktion. Funktionens attacktid forbliver den samme, men filterets cutoff frekvens og længden på decayet bliver moduleret af to [drunk] objekter. De bliver opdateret hvert sekund, så filterenvelopen (funktionens påvirkning af filtret), er aldrig den samme, men hopper lidt rundt.

Amplituden på signalet styres også af en AD-funktion, hvis decay påvirkes af et [random 3] objekt, som skiftevis ændrer funktionens decay til enten 250 ms., 500 ms. eller 1000 ms. Dette sker hvert 1000 ms. Et ekstra [metro 1500] objekt, sørger for at funktionen bliver triggeret en ekstra gang hver 1,5 sekund.

Til sidst bliver signalet ledt igennem [p delay], hvor et comb-filter agerer delay-effekt, med triol intervaller. På mix-knappen kan man indstille mængden af delay på signalet. Efter delay-subpatcheren, bliver signalet ledt ind i et rumklangsobjekt, hvor feedbackværdien, bliver styret af en random-generator.

Æstetisk
Omgivelserne styres og forandres igennem akkordalgoritmen, som leder vejen. Foranderligheden bliver forstærket igennem store ændringer i lydens toneleje igennem deterministisk oktavtransponering. Deterministisk, da man har 3 oktaver som bliver gennemgået serielt, ligesom Schoenbergs algoritme til udvælgelse af toner. Små stokastiske ændringer i de dynamiske parametre bidrager også til foranderligheden, som drunk- og randomobjekterne, der styrer funktionerne i envelopegeneratorerne og feedbackparametret i rumklangen. Over tid, kan drunkobjekterne også ændre lyden markant, i takt med at drunkobjektets værdi kommer længere væk fra udgangspunktet. En delay og et ekstra metroobjekt som styrer amplitudefunktionen, gør rytmen mindre pulserende.

Additiv synthesizer (Nedbør)

[p nedbør] indeholder en additiv synthesizer. Additiv syntese er en synteseteknik, som mixer (addere) sinustoner sammen. Alle lyde i verden er opbygget af sinustoner med forskellige frekvenser, fase og amplitude. Derfor bidrager hver eneste sinustone til klangfarven på den endelige lyd (Cipriani & Giri, 2016, s. 201). På samme måde, kan man opbygge de tidligere omtalte bølgeformer savtand og rektangel med sinustoner. For at en lyd kan være harmonisk og have en tone, skal den have en grundtone eller et frekvensspektrum som er opbygget i oktaver.

Teknisk
Den additive synthesizer modtager information om tonehøjde fra nodealgoritme2. Undervejs bliver signalet ledt ud til otte sinustonegeneratorer (oscillatorer), og signalet bliver ganget op inden hver eneste generator, så man ender ud med toner over otte oktaver. Sinustonernes amplitude, bliver styret af otte funktionsgeneratorer, med forskelligt attack og decay. Funktionerne bliver aktiveret af to forskellige random-sequencere, som igen bliver triggeret af et [metro 500] objekt. Grundtonen bliver aktiveret af et bang hvert halve sekund, i modsætning til de andre toner som bliver aktiveret tilfældigt på grund af random-sequenceren. Hele lyden bliver mixet ind og ud, med jævne mellemrum, i intervaller af otte sekunder, af en tilfældighedsgenerator.

Æstetisk
Sinustonerne fra den additive synthesizer, opstår tilfældigt i frekvensspektrummet, og falder dermed ligesom regn eller sne, billedligt talt. Tonalt spiller dens arpeggio modsat [omgivelser], og danner en sammenhæng med denne, men i modsætning til omgivelserne, kommer og går [nedbør], hvis amplitude bliver styret af en stokastisk tilfældighedsgenerator.

Granular 1 og 2 (Noise & Grit)

Noise og Grit er to granulare synthesizere. Granular syntese er en synteseteknik, hvor en lyd deles op i ”grains” eller korn imellem 10 ms. og 100 ms. I hvert korn bliver dynamikken og amplituden formet af en envelope, som kan have forskellige former (Roads, 1996, s. 168-9). Oprindeligt blev sofistikerede ombyggede båndmaskiner brugt til granular syntese. I dag kan man gøre det digitalt. Typen af granular synthesizer, som bliver brugt i Noise og Grit, er samplebaseret.

Teknisk
Den granulare synthesizer i Max, fungerer ved at man udvælger et område i en lydfil, hvor der afspilles grains. Otte identiske [grain] objekter, giver bredde til den syntetiserede lyd. I [Buffer envelope2], bliver der lagret en meget kort envelope, som bruges til at forme dynamikken i det enkelte grain. I Noise og Grits tilfælde er der brugt en halv sinustone. Stokastiske elementer styrer afspilningen af grains i lydfilerne, og panorerer, starter og stopper afspilningen tilfældigt.

Æstetisk
Noise og Grit giver støj og uharmoniske lyde til kompositionen, ligesom billedet af skillevejen er grynet, uden farver og i dårlig kvalitet. I Noise er der valgt en lydfil med af et brændende bål, og i Grit er lydfilen et trommeloop. Tilfældighedsgeneratorer styrer udvælgelsesprocessen i afspilning af lydfiler, i et prædefineret tempo, som stemmer overens med tempoet for hele kompositionen. Dette giver et rytmisk lag. Tilfældigt styret panorering, får lydene til at bevæge sig frem og tilbage imellem højre og venstre kanal.

Historisk og æstetisk perspektivering

Skillevej er en algoritmisk komposition, som er lavet i Max, men hvilke historiske teknikker kan mine sammenlignes med? For at finde ud af dette, må vi tage afsæt i det første afsnit i denne tekst kaldet: ”Om algoritmisk musik og Max” på side 3, som beskriver nogle af de vigtigste historiske personer og teknikker, brugt i algoritmisk musik. Som tidligere beskrevet, er det statistiske afsæt i min akkordalgoritme, baseret på matematiske analyser af Bach og Mozart, foretaget af Kiefer & Riehl (2016). Bach og Mozart har brugt både stokastiske og deterministiske algoritmiske metoder til kompositionen i deres musik, men tonalitet og kadencen, som er blevet analyseret af Kiefer & Riehl, er ikke nødvendigvis algoritmisk, selvom man kan se mønstre og regler, som det er tilfældet med Bach og Mozarts stykker. Dermed har jeg gjort det, som normalt er styret af komponistens følelser, intuition, musikalske viden og ører, og gjort det algoritmisk. Roads (2015) skriver i Composing Electronic Music: A New Aesthetic (s. 367):

> *We all know that machines can easily outdo us in certain tasks. If composition was merely a game of logic, then the technical virtuosity of the machine would already have relegated human efforts to a sideshow. … On a fundamental level, music is not a formal system of logical communication. It is a sensory experience involving acoustics, psychoacoustics, and subjective aesthetic response.*

Ifølge Roads (2015), har flere komponister, som har benyttet sig af algoritmer, igennem tiden forsøgt at genskabe musik fra komponister som Bach og Mozart. Ifølge ham ligger problemet ofte i valg af algoritmemodellen, som skal ”udregne” kompositionen (s. 345).

> *A human composer is listening as she goes and remembers what has been composed. In constrast, it seems that few generative have any awareness of what they have done beyond a few stages in a Markov chain.*
> (Roads, 2015, s. 366)

For at imødekomme dette problem, kan man arbejde videre med at gøre algoritmen mere kompleks. Man kan blandt andet variere længden på akkorderne, gøre melodierne og de melodiske rytmer

mere komplekse og tilføje flere tilfældige elementer, for at gøre kompositionen mere organisk eller menneskelig. Et eksempel på komponister som ændrede deres musikalske algoritmer var Gottfried Michael Koenig og andre serielle komponister, som i 1950'erne begyndte at blande deterministiske med stokastiske metoder, for at få mere kontrol over den musikalske struktur. Metoden som blev udviklet, hedder aleatorisk komposition (Essl, 2007, s. 117). Iannis Xenakis kritiserede også den serielle deterministiske model. Xenakis tyede til brug af stokastiske algoritmer i sine kompositioner, heriblandt også kompositioner baseret på grafiske tegninger (Essl, 2007, s. 115-6)

Josef Matthias Hauer (se s. 3), benyttede en metode til algoritmisk komposition, som har visse ligheder, med den metode som jeg har brugt. Ud fra en deterministisk model, komponerede han sin musik ud fra 12 akkorder (akkordalgoritmen/gadelamper), som han lagde barok-inspirerede arpeggioer oven på (nodealgoritmen/omgivelser). En væsentlig forskel imellem hans algoritme og min, er at hans var deterministisk og min er stokastisk, dog er dele af de kompositoriske elementer meget ens. Han så ikke sig selv som en komponist, som udtrykkede følelser igennem sin musik, men snarere som et medium, som omdannede kosmiske vibrationer til lyd (Essl, 2007, s. 110).

Tonaliteten i lydlandskabet *Skilleveje*, er styret af stokastiske algoritmer, og det musikalske lydlige udtryk er styret af forskellige former for synthesizere. Synthesizerlydene repræsenterer forskellige genstande, kvaliteter og tilstande, men skal ikke være en direkte lydlig repræsentation. Om dette har Karlheinz Stockhausen skrevet:

> *It is clear that a composer of electronic music should not try to imitate timbres of the traditional instrumentarium or familiar sounds and noises. If, exceptionally, a sound of this kind is required, it would be unfunctional to generate it synthetically...*
> (Cox & Warner, 2004, s. 374)

Den subtraktivt syntetiserede lyd, *Omgivelser*, kan engang imellem lyde som et syntetiseret klaver, men straks ændres lydens amplitude og klangfarve, i takt med at lydens dynamiske parametre ændres af de stokastiske tilfældighedsgeneratorer.

Iannis Xenakis påpeger, at de stokastiske, statistiske og ubestemte principper, findes i naturen (Essl, 2007 s. 116). Den additivt syntetiserede lyd *Nedbør*, gør brug af dette, i form af sin stokastiske sequencer, som afspiller sinustoner i forskellige oktaver tilfældigt.

Konklusion

Algoritmisk musik er ikke en ny opfindelse, da det har eksisteret siden oldtiden. Det samme kan man sige om de stokastiske og deterministiske tilgange til musikkomposition, som har været brugt i mange hundrede år, af blandt andet Bach og Mozart.

I mit algoritmisk producerede musikstykke, som er produceret i Max, har jeg taget udgangspunkt i statistiske modeller for Mozarts og Bachs brug af tonalitet, og brugt det i et stykke, som primært er instrumenteret af forskellige former for synthesizere. Algoritmen er også konceptualiseret igennem et billede som forestiller en skillevej. Skillevejen er et billede på den vej, som algoritmen tager percipienten i stykket, og stokastisk selektion vælger vejen. Vejen som algoritmen tager dig, har

indflydelse på omgivelserne og tonalitet. Klangfarverne er styret mere tilfældigt. Lydlandsskabet, som passerer undervejs, er baseret på klassisk modulær synthesizeræstetik.

Curtis Roads kristiserer denne måde, at gribe algoritmisk komposition an på, da markovkæder og andre statistiske modeller, aldrig vil kunne genskabe en klassisk komponists arbejdsmetoder. Josef Matthias Hauer, som også arbejdede med algoritmisk musik, blandet med klassiske elementer, så dog ikke sig selv som komponist, men mere som et medium.

For at gøre kompositionen mere levende, kan man måske gøre, ligesom komponisterne som startede med at bruge aleotorisk komposition, hvor man kombinerer algoritmiske og ikke algoritmiske kompositionsteknikker, for at skabe noget mere komplekst og organisk. En anden måde, kan være at gøre algoritmen mere kompleks.

Litteratur og kilder

Cipriani, A., & Giri, M. (2010). Electronic Music and Sound Design: Theory and Practice with Max 7. Volume 1. Italy: ConTempoNet s.a.s.

Essl, Karlheinz. (2007). "Algorithmic Composition." In The Cambridge Companion to Electronic Music, edited by Julio d'Escrivan Nick Collins, 107−125. New York: Cambridge University Press.

Roads, Curtis. (1996). ”The Computer Music Tutorial.” Cambridge: The MIT Press.

Roads, Curtis. (2015). ”Composing Electronic Music: A New Aesthetic.” New York: Oxford University Press

Breinbjerg, Morten. (2007). "Musikkens Interfaces." In: *Interface*, edited by Søren Pold and Lone Koefoed Hansen, 137 −69. Aarhus Universitetsforlag.

Stockhausen, Karlheinz. (2004). ”Electronic and Instrumental Music.” ”In Audio Culture: Readings in Modern Music”, edited by Cristoph Cox and Daniel Warner, 370-443. New York: Continuum

Kiefer, Peter & Riehl, Manda. (2016). ”Markov Chain of Chord Progressions.” In: Ball State Undergraduate Mathematics Exchange. Vol. 10, No. 1 (Fall 2016), 16-21. Lokaliseret d. 8/12-2017 på:
http://www.bsu.edu/libraries/beneficencepress/mathexchange/10-01/MarkovChainsChordProgressions.pdf