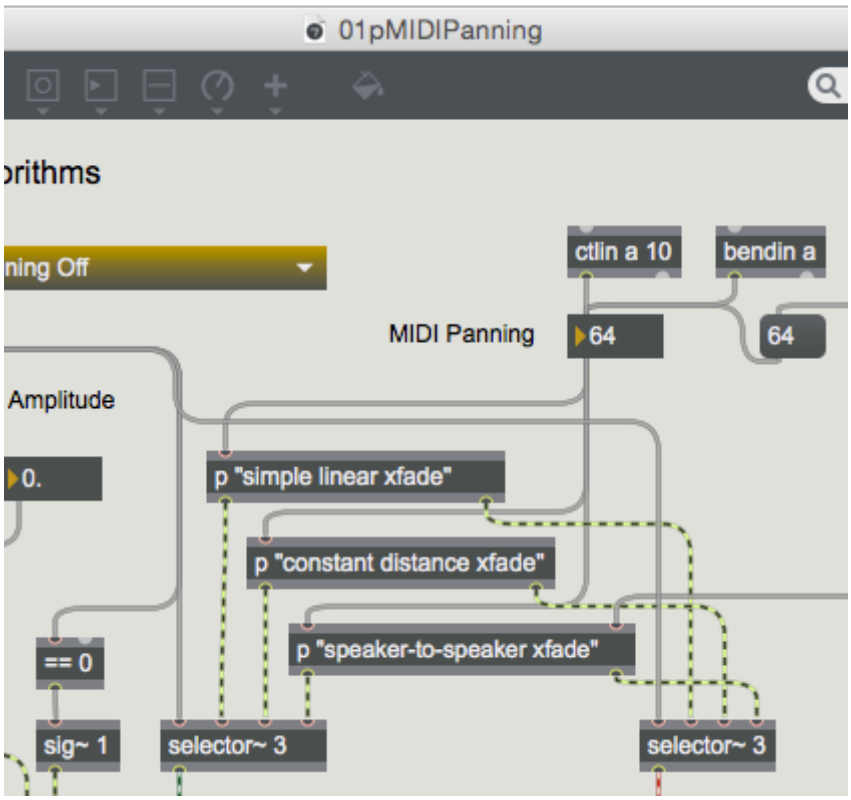
Kontrolloppgave til øvelsen *Simple Panning*

Denne øvelsen går inn på tre sentrale metoder for å simulere plassering av lyd i rom (Hammer s. 36 og s. 77 og Roads s. 449). Noen aspekter ved lydens plassering i rommet, som f.eks. amplitudenivåer, kan være vanskelig å forestille seg på et såpass abstrakt nivå. Derfor kan det lønne seg å først tegne opp rommet og lydens bevegelser mellom høyttalerne på et stykke papir.

Gå gjennom øvelsen *Simple panning* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 64

Ta utgangspunkt i øvelsen. Start med lydkilden du finner i subpatchen *«sound source»*, men modifiser den slik at den repeteres 4 ganger i sekundet med en tonehøyde som går fra 4 kHz til 1 kHz.

Panorer denne lydkilden mellom høyttalerne ved hjelp av pitchbend-hjulet på MIDI-klaviaturet. Benytt en panoreringsmetode som gir lik intensitet alle steder mellom venstre og høyre kanal. For å få til dette må man simulere lik avstand til lytteren i en jevn bue mellom høyttalerne.

Istedenfor at lyden panoreres fra venstre til høyre, så kan den panoreres fra høyre til venstre.

Nye objekter introdusert i denne øvelsen:

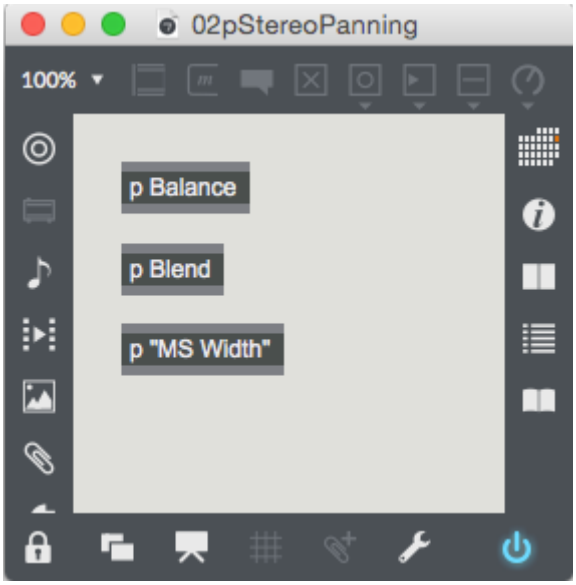
expr~

gate~

Kontrolloppgave til øvelsen *Stereo Panning*

Denne øvelsen gir oss et innblikk i ulike måter å arbeide med lyd i rom på.

Gå gjennom øvelsen *Stereo panning* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 65

65.1 Ta utgangspunkt i *Stereo panning*-øvelsen, og lag en enkel patch som panner mellom venstre og høyre kanal. La verdien for «base» være konstant på 0.75 for å gjøre patchen enklere.

65.2 La en oscillator med en frekvens på 0.125 Hz bevege lyden mellom venstre og høyre kanal. For å få til dette kan det være greit å bruke *snapshot*~-objektet for å gjøre lyd om til kontrolldata.

65.3 Ta utgangspunkt i FM-syntesen fra øvelsen *Vibrato and Frequency Modulation*. Gi den følgende parametere:

Carrier Frequency: 100 Hz
Modulator Frequency: fra 0 til 100 og tilbake til 0 på 20 sekunder.
Modulation Depth: 2000

65.4 Lag en patch der oscillatoren som du bruker i modulasjonsfrekvensen (Modulator Frequency) som går fra 0 til 100 og tilbake til 0 på 20 sekunder også styrer panoreringen. Husk å sette verdien til *snapshot*~-objektet til 1 ms samt forandre interpolasjonsverdien i *line*~-objektet til 1 ms slik at den høye modulasjonsfrekvensen til FM-syntesen også fungerer for panoreringen.

65.5 Til sist kommer en oppgave som ikke er obligatorisk. Om du ønsker en ekstra utfordring kan du gjenskape panoreringsalgoritmen («expr pow((1.-$f1),0.75)» og «expr pow($f1,0.75)») med audio-objekter. Dette løser du blant annet med objektene *pow*~, *sig*~ og !-~. Fordelen med å bare bruke audio-objekter er at alt foregår på samplenivå, noe du hører tydelig på høye modulasjonsfrekvenser.

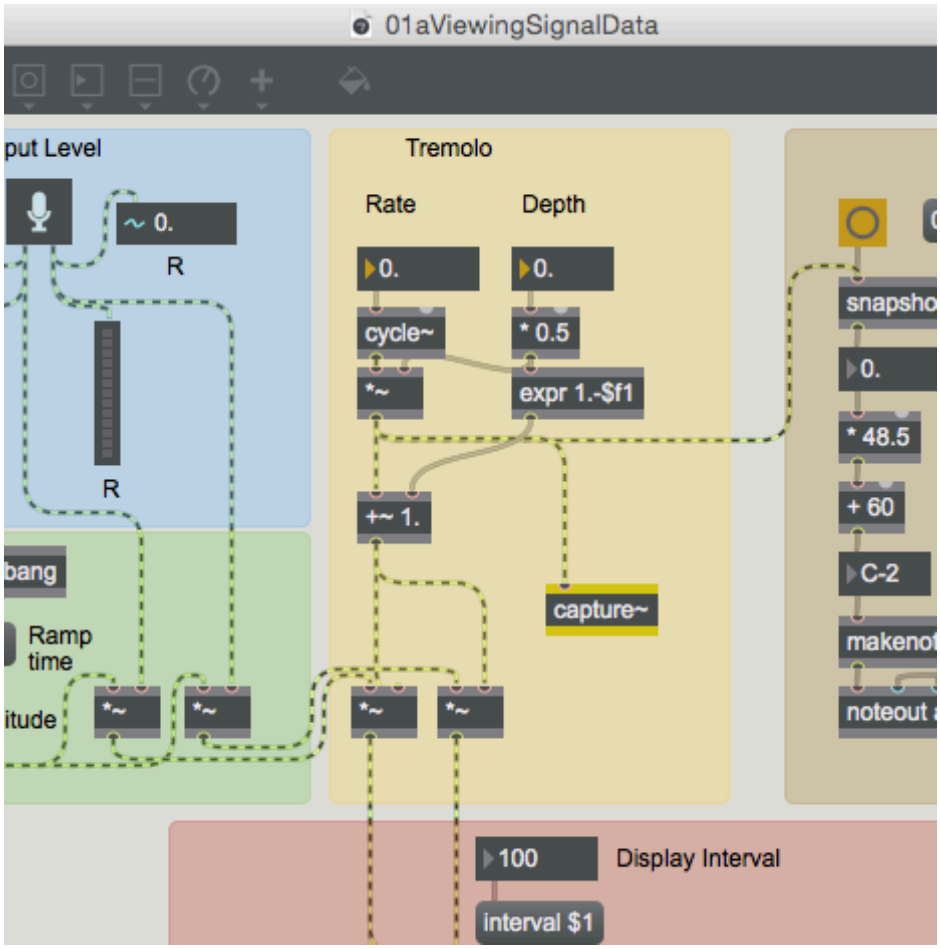
Nye objekter introdusert i denne øvelsen:

*~ +~ scale expr

Kontrolloppgave til øvelsen *Signals and Meters*

Vi vil hoppe over øvelsen *Multi-channel Panning*, siden vi trenger en flerkanals høyttaleroppsett for dette. Hvis du har tilgang til et slikt høyttaleroppsett anbefales det på det sterkeste at du jobber gjennom denne øvelsen siden flerkanals panorering er en svært interessant del av arbeidet med Max. Hvis ikke, bør du hoppe over denne øvelsen og gå tilbake til den når du har tilgang til et slik høyttaleroppsett. I stedet går vi rett til øvelsen *Signals and Meters*. Det kan være essensielt å ha visuelle tilbakemeldinger på lyden (Hammer s. 80 og Roads s. 545) for blant annet å kunne kontrollere en signalgang. F.eks. blir «*meter~*»-objektet ofte brukt mange steder i en patch for å holde oversikt på signalgangen.

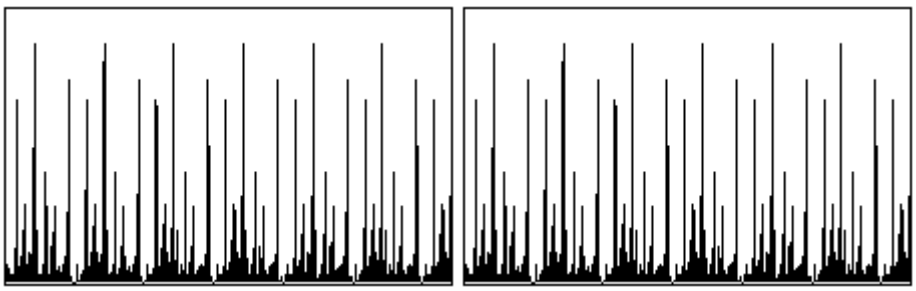
Gå gjennom øvelsen *Signals and Meters* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 66

Ved hjelp av blant annet objektene *meter~* og *multislider*, lag en patch som visualiser amplituden for venstre og høyre lydinngang på lydkortet over tid. Husk at *multislider*-objektet må stilles inn slik at det viser amplitudeforandringer grafisk over tid.



multislider-objektet skal vise amplitudeforandringer grafisk over tid

Nye objekter introdusert i denne øvelsen:

capture~

meter~

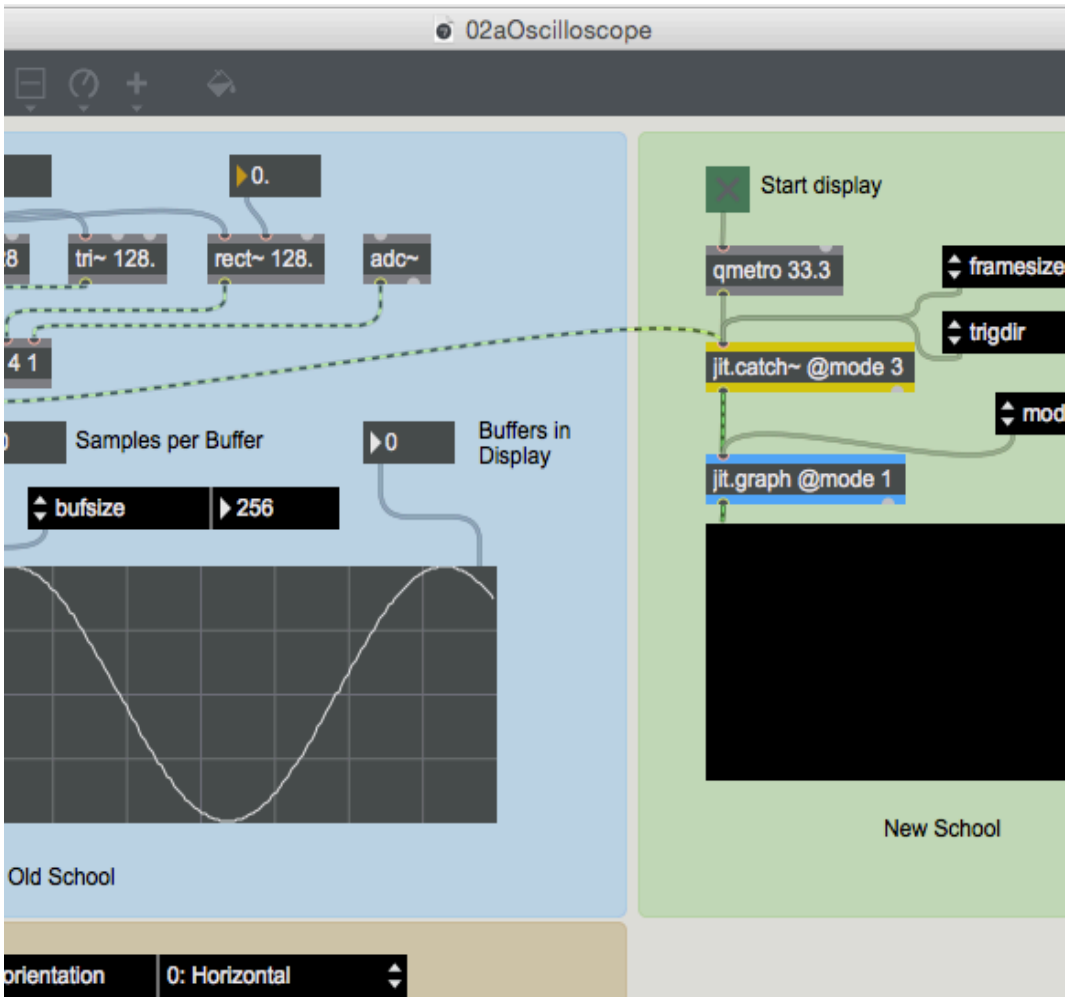
number~

snapshot~

Kontrolloppgave til øvelsen *Oscilloscopes and Spectroscopes*

I denne øvelsen ser vi på et oscilloskop og et spektogram, begge nyttige verktøy for visualisering av lyd. Oscilloskopet er mest nyttig for visualisering av tidsdomenet, mens spektogrammet er mest nyttig for visualisering av frekvensdomenet (Hammer s. 80 og Roads s. 545).

Gå gjennom øvelsen *Oscilloscopes and Spectroscopes* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 67

Gjør følgende:

Start med en sinustone på 200 Hz.

Legg til en sinustone til slik at det oppstår en rytme, en såkalt *beating* (Hammer s. 26), på ett slag i sekundet.

Vis dette i et oscilloskop (*scope*~-objektet) og vis det også i et spektrogram (*spectroscope*~-objektet).

Legg merke til forskjellen på representasjon av lyd i tidsdomenet (*scope~*) og frekvensdomenet (*spectroscope~)*.

Kopier de gradvis fallende tonehøydene som du lagde i oppgaven til øvelsen *Routing Signals* (Oppgave 37). Visualiser den med et *spectroscope~*-objekt og et *scope~*-objekt. Still inn *spectroscope~*-objektet slik at det bare viser frekvenser mellom 0 og 1000 Hz (gå til Inspector og gjør om verdiene i «Lo and Hi Domain Display Value (X-Axis)»).

Nye objekter introdusert i denne øvelsen:

scope~

jit.catch~

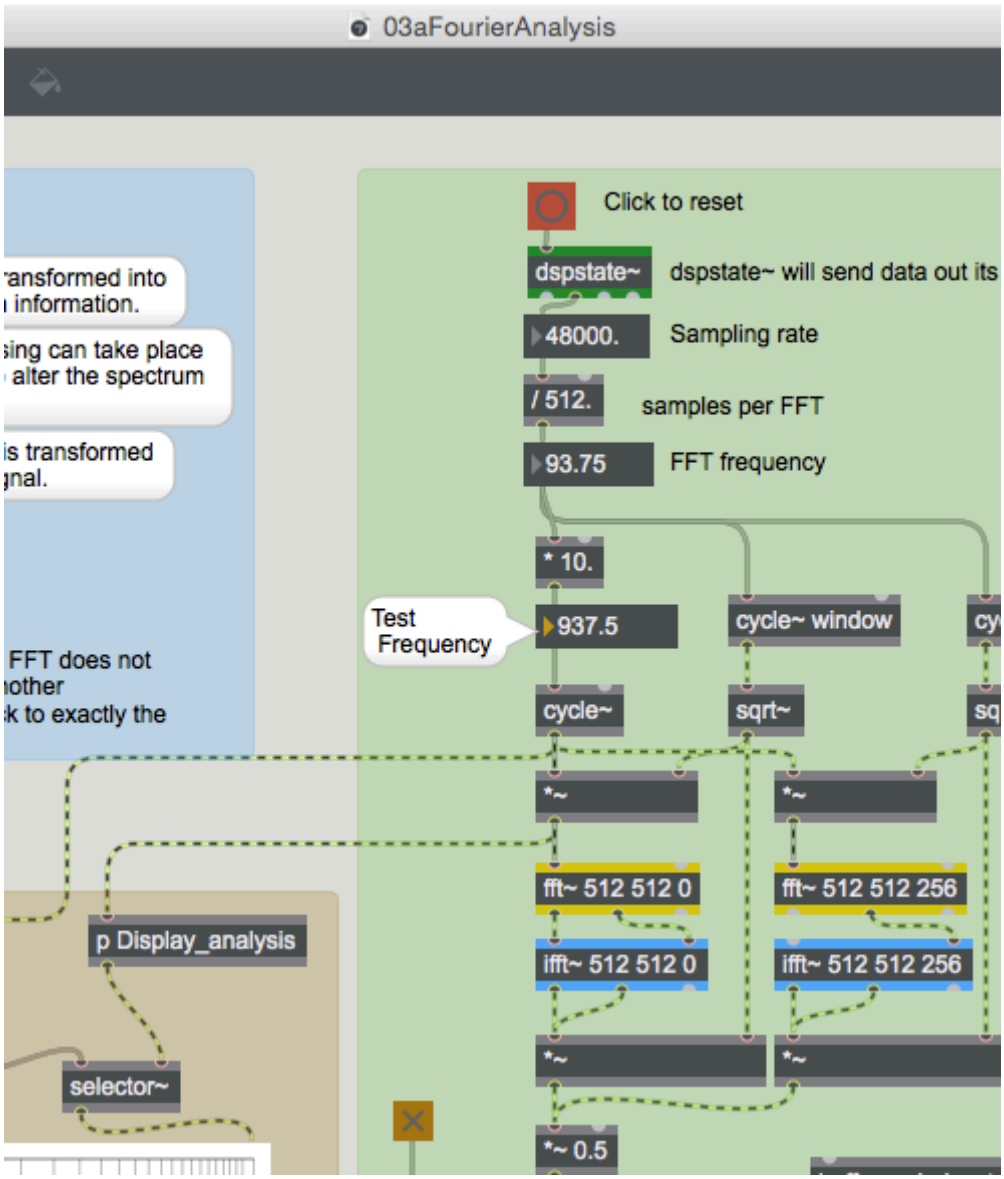
jit.graph~

spectroscope~

Kontrolloppgave til øvelsen *Simple Fourier Analysis*

Denne øvelsen går gjennom grunnleggende aspekter ved en såkalt *Fast Fourier Transform*, vanligvis forkortet til *FFT* (Hammer s. 68-73 og Roads s. 1073).

Gå gjennom øvelsen *Simple Fourier Analysis* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 68

Denne øvelsen foretar hverken en signalbehandling eller en syntese. Den går «bare» igjennom grunnleggende aspekter ved FFT. Men for at du allikevel skal være sikker på at du behersker de grunnleggende objektene så skal du lage følgende patch.
Spill av lyd og legg inn en mulighet for å skru av og på repetisjon av lyden (looping). Dette kan du gjøre med å lage denne funksjonen selv med et «sfplay~»-objekt, eller med å enkelt dra en lydfil over i Max. Når du drar en lydfil over i Max oppretter du automatisk et «playlist~»-objekt. La lyden analyseres gjennom en FFT-transformasjon. Foreta deretter en resyntese med en invers FFT

(IFFT). Så lenge vi ikke forandrer på lyddataene mellom FFTen og IFFTen skal lyden være identisk også etter resyntesen.

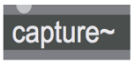
Til slutt kan du ta ut frekvensdata med *capture*~-objektet.

Nye objekter introdusert i denne øvelsen:

ifft~

fft~

capture~

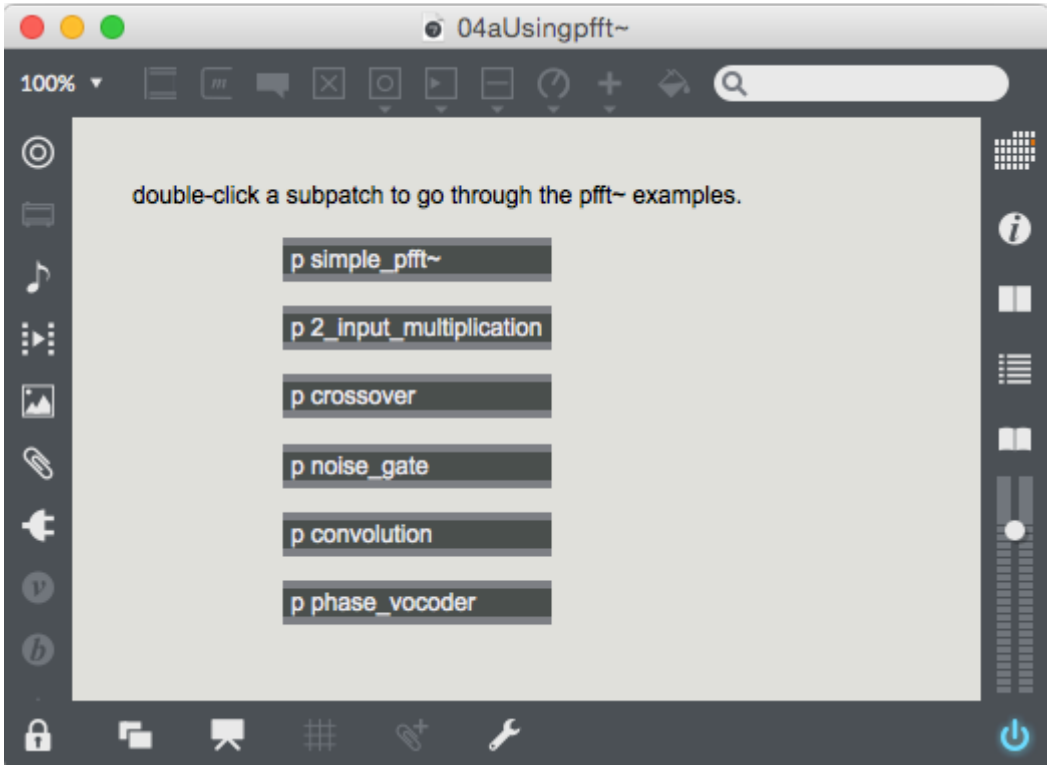


buffer~

Kontrolloppgave til øvelsen *Advanced Fourier Analysis*

Denne øvelsen er nok en av de mest komplekse av MSP-øvelsene og går inn på litt mer avanserte aspekter ved Fourier-transformasjonen. Her vil vi også, i motsetning til forrige øvelse, ha muligheten til å foreta signalbehandling på FFT-materialet. Vi skal gå gjennom «*pfft~*»-objektet som er et mer brukervennlig FFT-objekt enn det vi gikk gjennom i forrige øvelse (Hammer s. 68-73 og Roads s. 1073).

Gå gjennom øvelsen *Advanced Fourier Analysis* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 69

I subpatchen *p noise_gate* under *pfft~ ngate~ 1024 2* ligger det en patch som heter *ngate~*.

Kopier denne patchen. For å åpne denne patchen gjør du slik: Lås opp patchen. Høyreklikk på pfft~-objektet, velg «Object» i menyen, og velg «Open Original "ngate~"» Lås opp ngate~-patchen og kopier den over i en ny patch. Lagre denne nye patchen.

Gi den et nytt navn, som f.eks. *noisegate~*. Lagre den i din egen mappe og kall den opp fra *pfft~*-objektet på samme måte som i subpatchen «*p noise_gate*».

Forsikre deg om at den virker.

Vi skal nå gi *noisegate~* en ekstra inngang slik at *pfft~* får tre innganger og ikke to. Benytt objektet *in* og gi det et eget nummer, f.eks. *«in 3»*.

Den tredje inngangen skal gå til det samme stedet i *noisegate~* som inngang nr to, altså til *«>~*

0.5». Nå må du sannsynligvis lagre, lukke og åpne patchene igjen for at forandringene skal bli aktive.

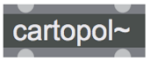
En slider kobles til den tredje inngangen i *pfft~*. Denne slideren skal gi flyttall fra 0-10 med en oppløsning på 100 steg.

Nå kan du styre grenseverdien for støyreduksjonen ved å benytte nummerobjektet som er koblet til inngang 2 og slideren som er koblet til inngang 3. (Vi kunne riktignok ha koblet begge sliderene til samme inngang for å oppnå samme resultat, men dette var bare ment som en øvelse i å bruke *pfft~* og tilhørende subpatcher).

Nye objekter introdusert i denne øvelsen:

adstatus

cartopol~



dspstate~

fftin~

fftout~

framedelta~

pfft~

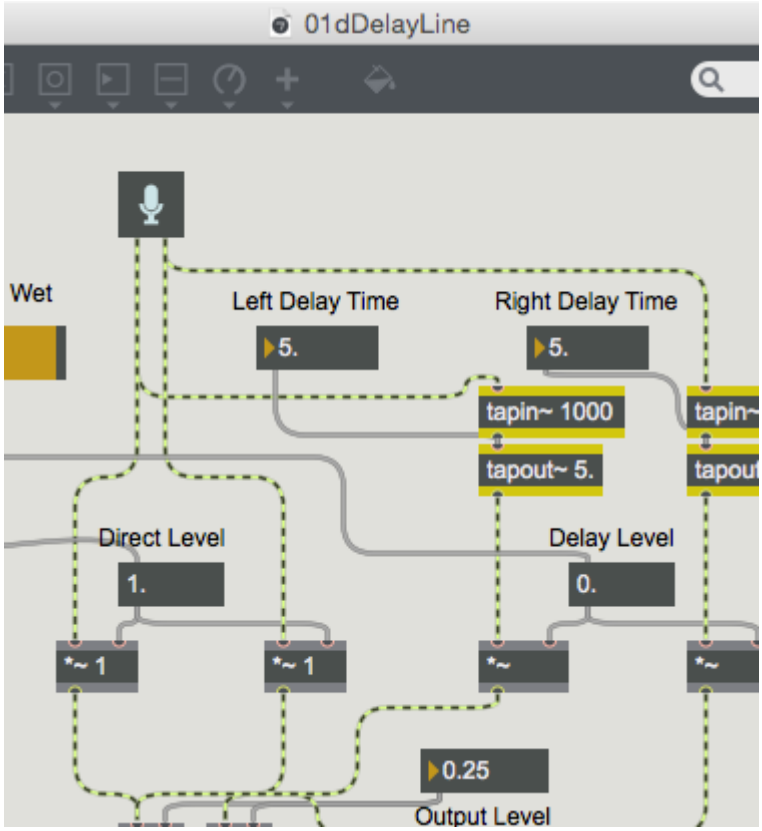
phasewrap~

poltocar~

Kontrolloppgave til øvelsen *Simple Delay Lines*

Denne øvelsen går inn på grunnleggende forsinkelsesteknikker *(delay)* (Roads s. 432-435). Forsinkelser av signalet er en av de mest sentrale teknikkene i musikalsk signalbehandling og brukes i alt fra romklang og ekko til korefekter (chorus) og filteralgoritmer.

Gå gjennom øvelsen *Simple Delay Lines* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 70

Spill av en lydfil og legg inn en mulighet for å skru av og på repetisjon av lyden (looping). Legg til to forsinkelser *(delay)*, en for venstre kanal og en for høyre. Venstre kanal skal ha en forsinkelse på 800 ms og høyre kanal skal ha en forsinkelse på 1050 ms. Det skal være et skyvepotensiometer *(fader)* for volumet til direktelyden og et skruknapp (dial) for volumet til den forsinkede lyden.

Nye objekter introdusert i denne øvelsen:

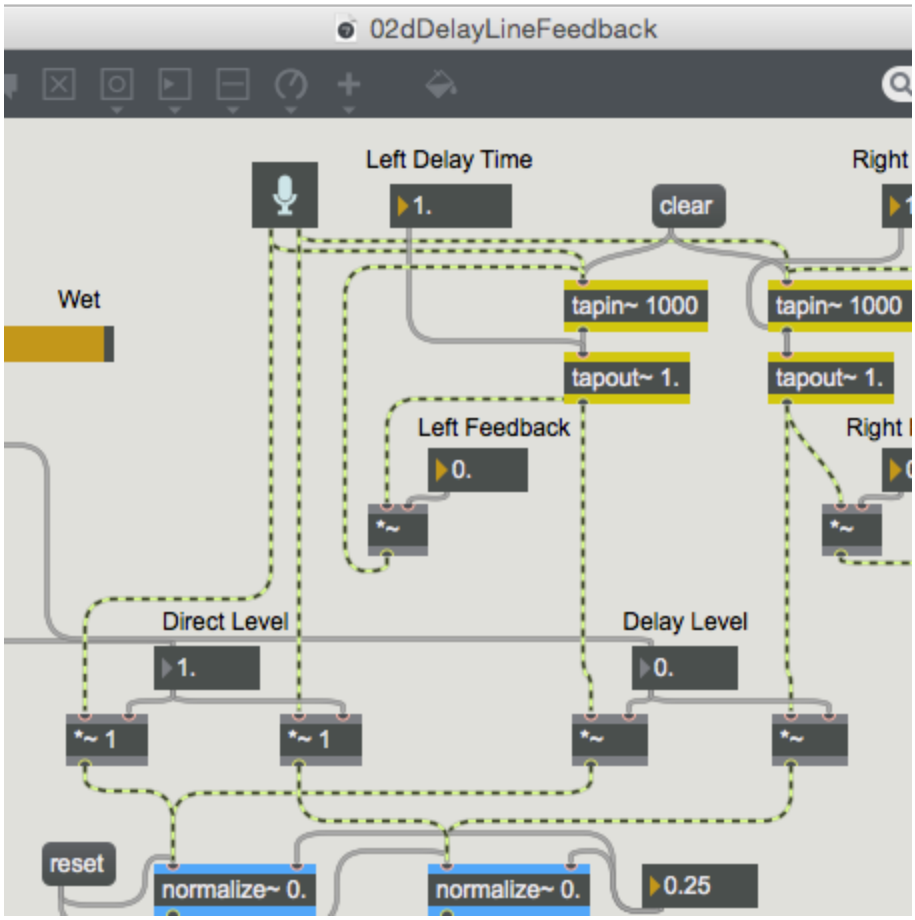
tapin~

tapout~

Kontrolloppgave til øvelsen *Delay Lines with Feedback*

Denne øvelsen går videre der vi slapp i forrige øvelse hvor vi lagde forsinkelser *(delay)*. I denne øvelsen går vi videre og legger blant annet til en tilbakekobling *(feedback)* (Roads s. 432-435).

Gå gjennom øvelsen *Delay Lines with Feedback* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 71

Spill av en lydfil og legg inn en mulighet for å skru av og på repetisjon av lyden (looping). Legg til to forsinkelser *(delay)*, en for venstre kanal og en for høyre. Begge kanalene skal ha mulighet for tilbakekobling *(feedback)*.

Start med en kort forsinkelsestid, f.eks. 25 ms., og mens du spiller lyd gjennom patchen, øk amplituden på tilbakekoblingen *(feedback)* inntil du hører en resonans. Resonansen burde inntre når tilbakekoblingsamplituden er mellom 0.75 og 0.95.

Når du har fått en resonans i lyden, still inn forsinkelsestiden slik at lyden resonerer med en grunntone på 50 Hz i venstre kanal og 66,67 Hz i høyre kanal. I følge øvelsen finner vi forsinkelsens resonans på følgende måte:
1000/delaytid = resonansfrekvens.

Med følgende verdier skulle vi finne riktig resonans:
1000/20 = 50
1000/15 = 66,67

Altså; for å få en resonans på 50 Hz og 66,67 Hz må vi ha en forsinkelsestid på henholdsvis 20 og 15 ms.

Nye objekter introdusert i denne øvelsen:

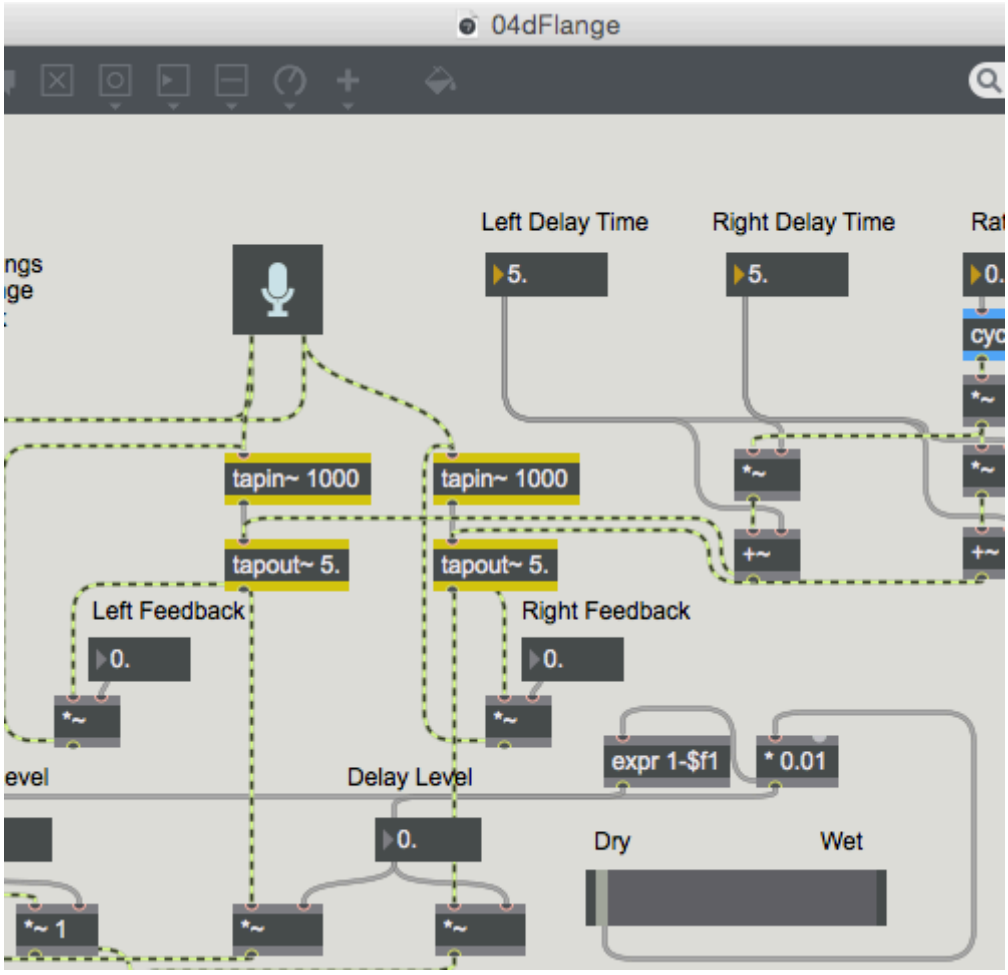
tapin~

tapout~

Kontrolloppgave til øvelsen *Flanger*

Vi vil hoppe over øvelsen *Automatic Gain Control with Feedback*. Selv om automatisk kontroll over lydstyrken kan være et praktisk verktøy i lange forsinkelses løkker (delay loops), har vi ikke bruk for dette nå. I stedet vil vi gå rett til øvelsen *Flanger*. Denne øvelsen går inn på variable forsinkelsestider (Hammer s. 79 og Roads s. 437) og hva dette har å si for både tonehøyde og klanglige aspekter.

Gå gjennom øvelsen *Flanger* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 72

Spill av en lydfil og legg inn en mulighet for å skru av og på repetisjon av lyden (looping).

Legg til to forsinkelser (delay), en for venstre kanal og en for høyre. Begge kanalene skal ha mulighet for tilbakekobling (feedback) og modulasjon.

I stedet for kun modulasjon med cosinus *(cycle~)*, benytt følgende grunnleggende bølgeformer: cosinus, trekant, sagtann og firkant.

Om du er usikker på hvordan du skal lage disse bølgeformene se på hjelpefilen til «*cycle~*». Se på den tab-en som er merket med «?». Her er det som vanlig et område med betegnelsen «See Also». Her kan du finne beslektede objekter som skulle sette deg i stand til å velge andre bølgeformer.

De fire bølgeformene skal du kunne velge fra en meny.

Fordelingen mellom «vått» og «tørt» signal velger du med en MIDI-kontroller, f.eks. modulasjonshjulet, og et *«slider»*-objekt.

Nye objekter introdusert i denne øvelsen:

noise~

rand~

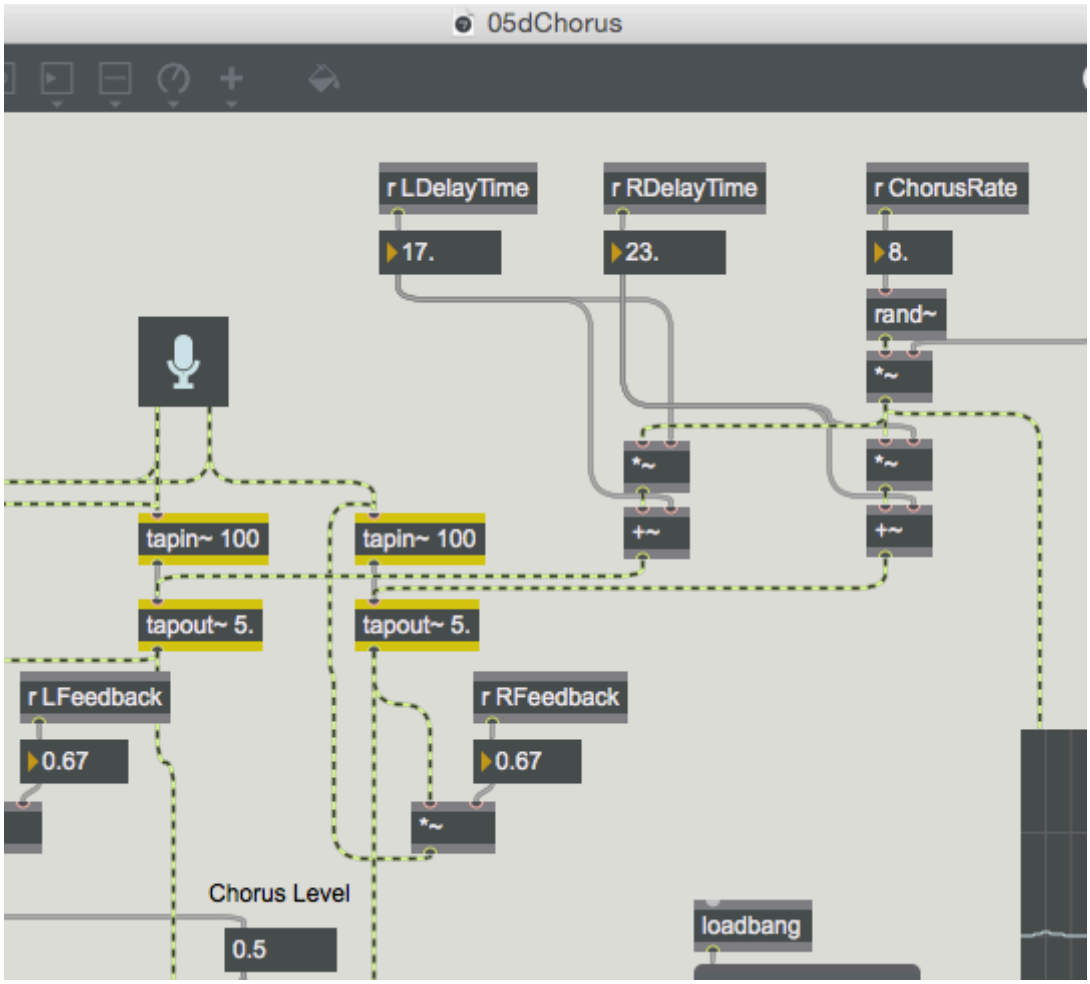
tapin~

tapout~

Kontrolloppgave til øvelsen *Chorus*

Denne øvelsen går inn på variable forsinkelsestider og hvordan vi kan modulere forsinkelsen med tilfeldigheter (Hammer s. 79 og Roads s. 439).

Gå gjennom øvelsen *Chorus* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 73

Spill av en lydfil og legg inn en mulighet for å skru av og på repetisjon av lyden (looping). Legg til to forsinkelser (delay), en for venstre kanal og en for høyre. Begge kanalene skal ha mulighet for tilbakekobling (feedback) og modulasjon.
MSP-øvelse 30 avslutter med en beskrivelse av en enda rikere koreffekt (chorus) som kan bygges ved å øke antall forsinkelses-tappinger (delay taps) med ulik tilfeldig modulasjon på hver enkelt forsinkelse.

Bygg en slik koreffekt med et *tapin~*-objekt koblet til fire *tapout~*-objekter. Hvert *tapout~*-objekt skal ha sin egen tilfeldige modulasjon for å skape en rik koreffekt. Innstillingene skal lagres med et *preset*-objekt.

Koreffekten er mest tydelig på tonale lyder som f.eks. gitar, piano eller vibrafon. Perkussive lyder med mye støy, som for eksempel trommesett, gir ikke den samme effekten.

Nye objekter introdusert i denne øvelsen:

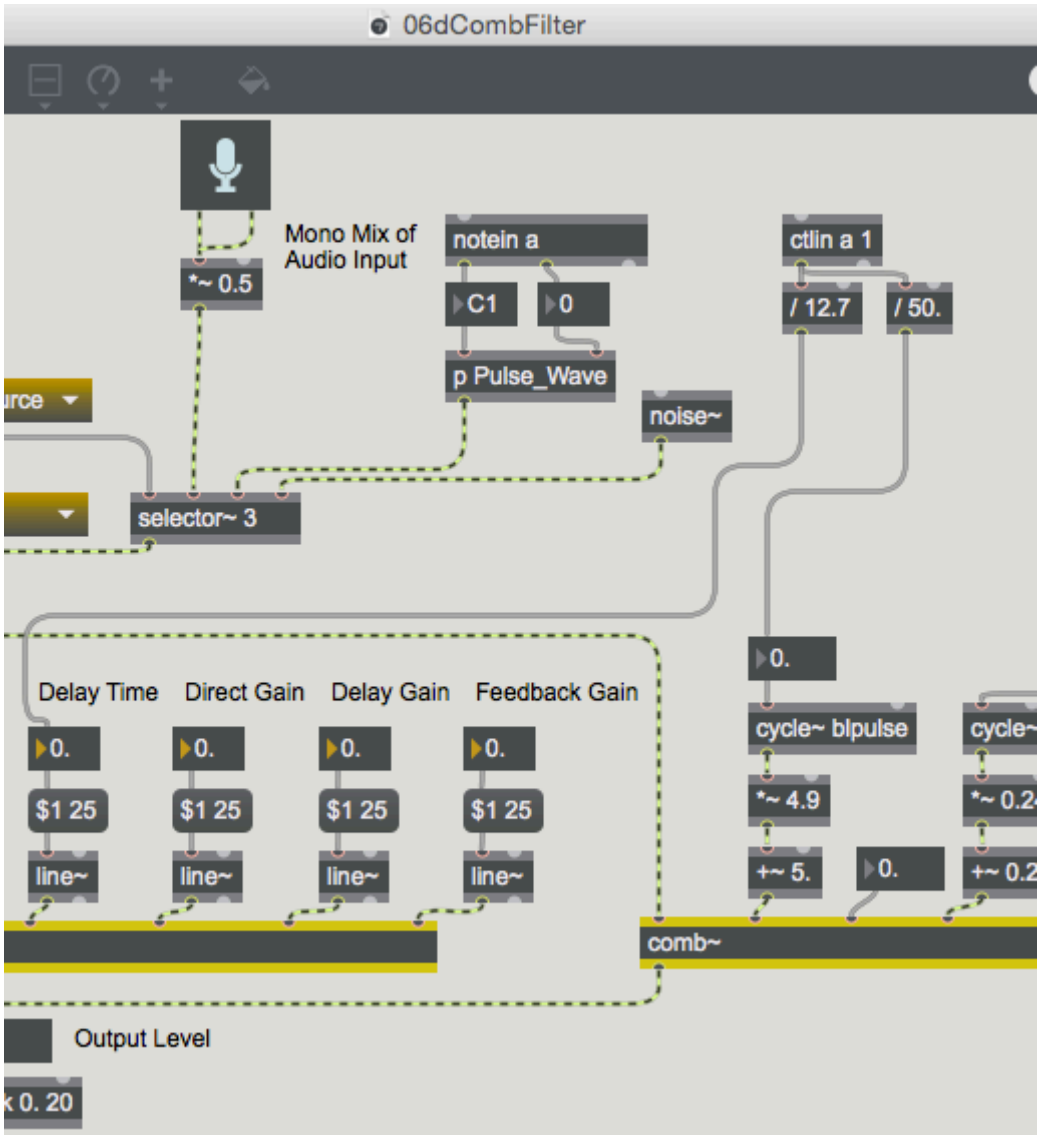
rand~

tapout~

Kontrolloppgave til øvelsen *Comb Filter*ing

Denne øvelsen går gjennom en spesiell variant av forsinkelser, nemlig kamfilteret (Hammer s. 66 og Roads s. 194). Kamfilterobjektet (*comb*~) opererer med svært små forsinkelser i området mellom 0 og 10 ms.

Gå gjennom øvelsen *Comb Filtering* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 74

Spill av en lydfil og legg inn en mulighet for å skru av og på repetisjon av lyden (looping). Legg til et kamfilter (*comb*~-objektet). Kamfilteret skal moduleres.

Parametrene f*orsinkelsestid* (Delay Time), f*orsinkelsesnivå* (Delay Gain) og *tilbakekoblingsnivå* (Feedback Gain) skal moduleres av en cosinus-kurve (*cycle*~). De ulike parametrene skal operere innenfor følgende verdier:

Forsinkelsestid: 0 - 9.9
Forsinkelsesnivå: 0 - 0.5
Tilbakekoblingsnivå: 0 - 0.9

Direktenivået (Direct Gain) skal settes manuelt og ikke moduleres.

Innstillingene skal lagres med et *preset*-objekt.

Nye objekter introdusert i denne øvelsen:

comb~

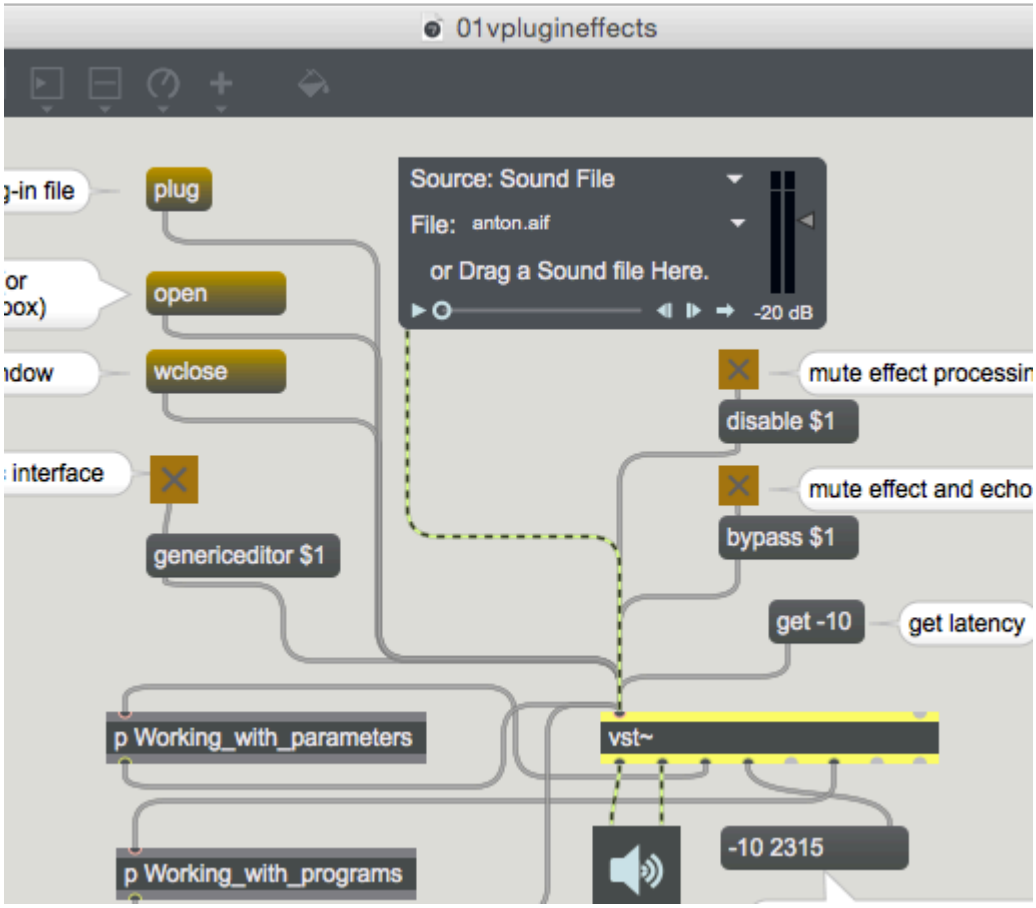
gain~

Kontrolloppgave til øvelsen *Plugin Effects*

Dette er den siste øvelsen. De resterende øvelsene om bruk av plugin instrumenter, Max for live patcher og komprimering, vil ikke bli gjennomgått da disse øvelsene er nokså spesialiserte og ligger utenfor rammene for dette kurset.

Det er noen ganger det kan være en fordel å bruke ferdige plugins til å gjøre ting som er vanskelige å få til i Max, som for eksempel gode romklanger. Denne øvelsen går inn på hvordan vi blant annet kan styre slike plugins på en fleksibel måte.

Gå gjennom øvelsen *Plugin Effects* og den tilhørende øvelsespatchen. Om det er noen punkter du er usikker på, gå tilbake og repeter det som er uklart. Om det er et objekt du ikke er helt sikker på, les referansen eller hjelpefilen til objektet.



Når du har gått igjennom hjelpefilene til de ulike filterobjektene og er sikker på at du skjønner hvordan filtrene fungerer, gå videre til oppgaven.

Oppgave 75

Spill av en lydfil og legg inn en mulighet for å skru av og på repetisjon av lyden (looping).

La lyden gå gjennom en delay i form av en *vst~*- eller *AU*-plugin. På OSX anbefaler vi å bruke pluginen «AUDelay». På Windows kan du bruke en annen delay plugin. Det finnes også VST plugins på nettet som man kan laste ned.

75.1
Styr parametrene til pluginen fra Max. «Feedback»-parametret skal være så høyt som mulig, og «dry/wet mix» skal være 100 prosent «wet».

75.2
I tillegg skal delaytiden automatiseres på to forskjellige måter:

75.2.1 La delaytiden svinge mellom svært kort og litt lengre tid med en frekvens på 0.125 Hz.

75.2.2 La delaytiden ha tilfeldige verdier mellom svært kort og litt lengre tid.

Legg inn muligheten for å velge delaytid som svinger, delaytid med tilfeldige verdier, og fast delaytid.

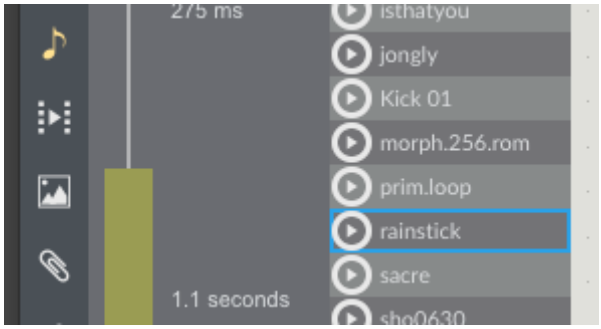
Nye objekter introdusert i denne øvelsen:

vst~

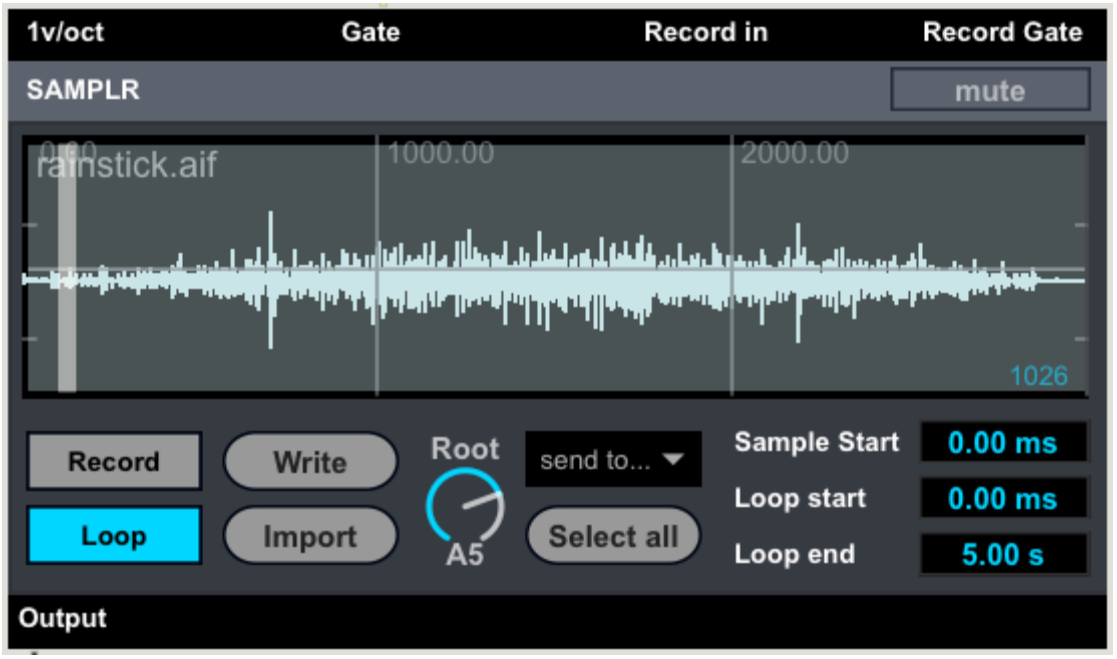
BEAP — effektkjede

I denne siste BEAP-oppgaven tar vi utgangspunkt i en enkel lydfilavspiller, men gjør lyden mer interessant med en kjede av forskjellige lydbehandlingsverktøy. På denne måten vil de ulike lydbehandlingsverktøyene virke inn på hverandre på komplekse måter.

Gå til *Oscilllator*-kolonnen og velg *Samplr*-patchen. Gå til *Input*-kolonnen, velg *Gate Pad*-patchen og koble til *Gate in* på *Samplr*-patchen. Koble dette til en *Stereo Output*-patch og trykk på *Gate Pad*-patchen for å starte avspillingen av en lydfil. Gå til *Audio*-menyen til venstre i patchervinduet, og bytt ut lydfilen med «Rainstick»:



Skru på *Loop* i *Samplr*-patchen for at lydfilen skal spilles av kontinuerlig. Med *Root* kan man transponere lyden opp og ned. Velg *A5* for å transponere lyden ned til en dyp tone.

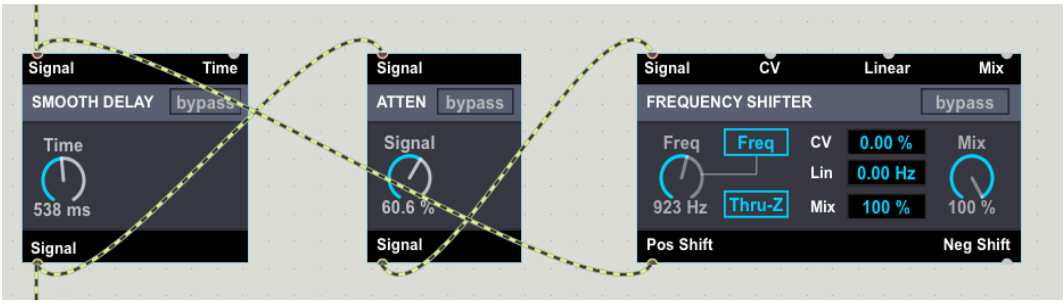


Gå deretter til *Effects*-kolonnen og velg *Classic Vocoder*-patchen. Still *Filter Q* på 82.6 og *Noise Threshold* 0.00. Med denne innstillingen fungerer vocooderen som en resonant filterbank.

Gå til *Effects*-kolonnen og velg *Smooth Delay*-patchen. Gå deretter til *Level*-kolonnen og velg *Attenuator*-patchen. Sett *Time* i *Smooth Delay* til 538 ms og *Signal* i *Attenuator* til 61.4 %. Koble utgangen fra *Voccoder* til inngangen på *Smooth Delay* og utgangen fra *Smooth Delay* videre ut til stereoutgangen.

Deretter kobler du utgangen fra *Smooth Delay* til inngangen på *Attenuator* og utgangen fra *Attenuator* til innganen på *Smooth Delay*. Du har nå laget en forsinkelse med tilbakekobling (på engelsk: *delay with feedback*).

Etter dette setter du en frekvensskifter i tilbakekoblings-kjeden. Gå til *Effects*-kolonnen og velg *Frequency Shifter*-patchen. Sett frekvens (*Freq*) til 923 Hz og *Mix* til 100%. Koble utgangen fra *Smooth Delay* til inngangen på *Attenuator*, utgangen fra *Attenuator* til innganen på *Frequency Shifter*, og *Pos Shift*-utgangen fra *Frequency Shifter* til inngangen på *Smooth Delay*.

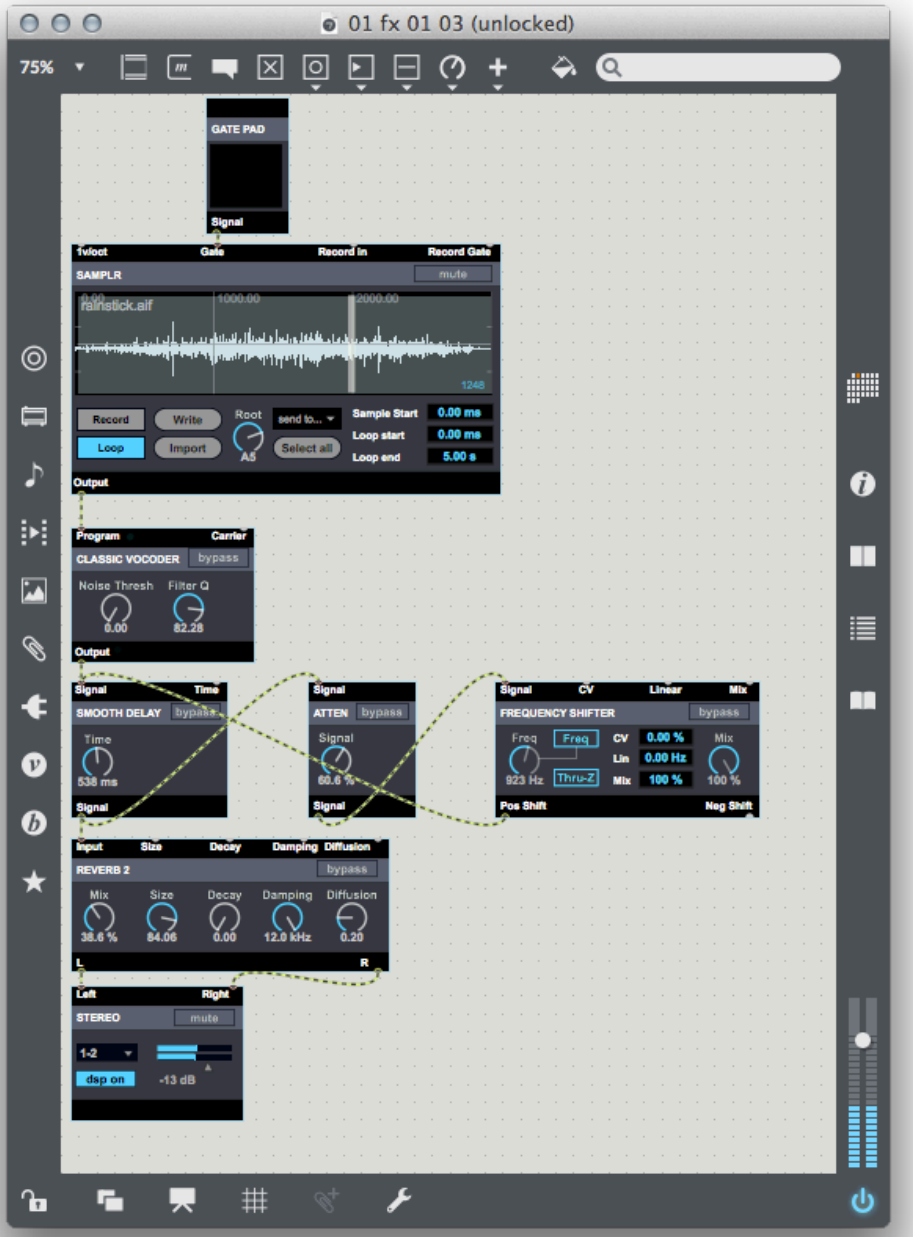


Du har nå laget en forsinkelse med tilbakekobling som også har en frekvensskifter i tilbakekoblings-kjeden.

Til slutt går du til *Effects*-kolonnen og velger *Reverb 2*-patchen med disse innstillingene:

Eksperimenter gjerne med andre lyder, men lagre den ferdige patchen med lydfilen «Rainstick».

Husk å velge «Autosave Snapshot» for å lagre innstillingene, og «Embed Patcher in Parent» for at «Samplr»-patchen skal huske hvilken lydfil du har valgt.



Hele patchen