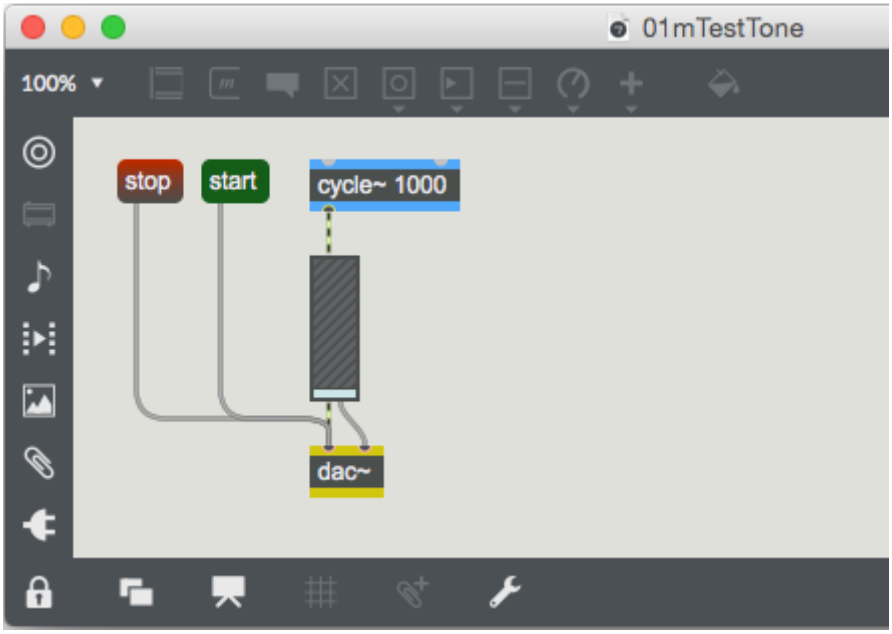
Kontrolloppgave til øvelsen *Test Tone*

Det er mulig å arbeide med Javascript i Max, men dette er et såpass spesialisert område av Max at vi ikke skal se på det i dette kurset. Vi går heller videre til MSP-delen av kurset. I denne øvelsen skal vi nærme oss MSP programmering for første gang. Siden vi her jobber med digital lyd og ikke bare data som i Max så vil vi komme inn på endel nye områder som akustikk, digital lyd, syntese og signalbehandling. Gå til Help->Reference og velg delen *Tutorials* som ligger under «MSP». Før du setter i gang med første tutorial, *Test Tone - Making sound*, kan det være lurt å ta en titt på de innledende kapitlene «Introduction», «How Digital Audio Works», «How MSP Works - Max Patches and the MSP Signal Network» og «Audio I/O - Audio input and output with MSP».

For de som ønsker å gå dypere inn i ulike problemstillinger rundt digital lyd, syntese og signalbehandling anbefales Øyvind Hammers tekst *Digital lydbehandling* fra 1997 produsert ved NOTAM. Teksten kan lastes ned *her*. For ytterligere utdypende informasjon les Curtis Roads bok *The Computer Music Tutorial* fra 1996 utgitt på MIT Press. Begge tekstene vil bli referert til i resten av kurset.
Den første MSP-øvelsen er en enkel generering av en sinustone (Hammer s. 48 og Roads s. 90).

Gå gjennom øvelsen *Test Tone* 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 34

Ved hjelp av fire sinusgeneratorer lag en c-moll akkord fra enstrøken C. Frekvensene til denne akkorden er:

261.6 Hz
311.1 Hz
392 Hz
523.3 Hz

For å unngå at signalet blir forvrengt kan du benytte signaldivisjonsobjektet (/~) slik at du kan dele summen av de fire sinusgeneratorene på fire. Hvis du ikke gjør dette blir lydstyrken for sterk og vi får et forvrengt signal, eller mer presist: vi får «klipping» av signalet (Roads s. 255).

Nye objekter introdusert i denne øvelsen:

cycle~

dac~

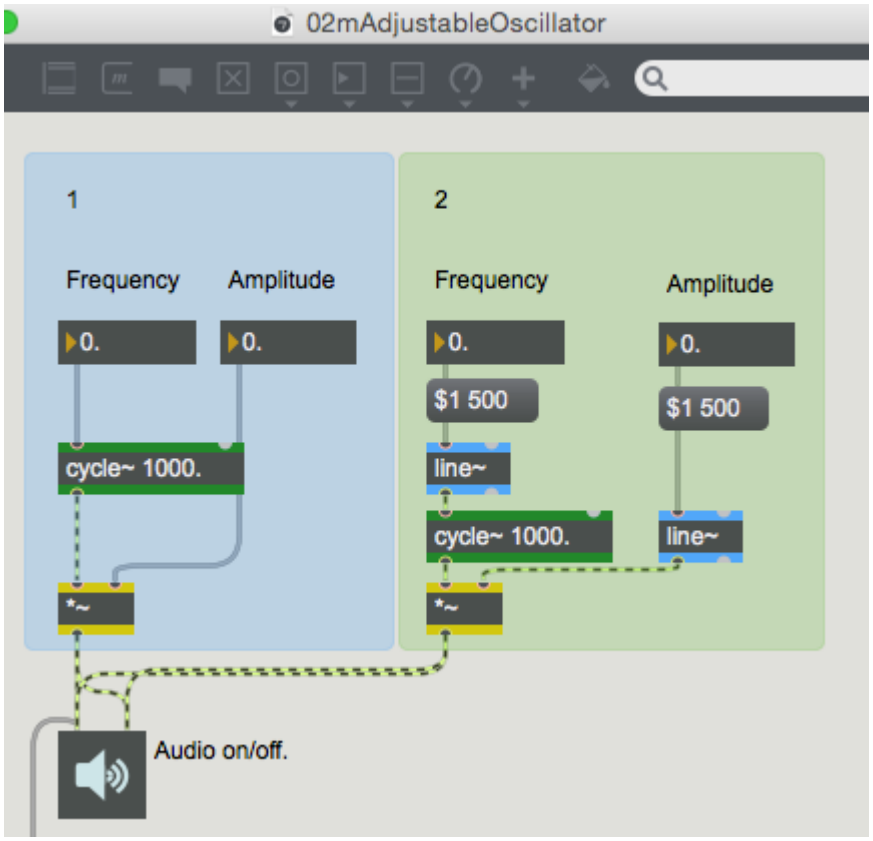
/~

gain~

Kontrolloppgave til øvelsen *Adjustable Oscillator*

Denne øvelsen fortsetter fra forrige øvelse, men legger til en såkalt omhyllingskurve (*envelope* på engelsk) (Hammer s. 48 og Roads s. 90).

Gå gjennom øvelsen *Adjustable Oscillator* 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 35

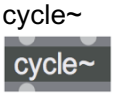
Ta utgangspunkt i sinusakkorden fra forrige oppgave. Legg til en omhyllingskurve (på engelsk kalles dette en envelope). Det skal være to knapper som brukes til å velge to ulike omhyllingskurver.

35.1 Lydstyrken i den første omhyllingskurven skal starte på 0, og gå opp til maksimum styrke i løpet av 10 millisekunder, for så å gå ned til null igjen i løpet av 2 sekunder.

35.2 Lydstyrken i den andre omhyllingskurven skal starte på 0, og gå opp til maksimum styrke i løpet av 5 sekunder, for så å gå ned til null igjen i løpet av 20 millisekunder. Om du er i tvil om hvordan oppgaven skal løses, les referansen eller hjelpefilen til *line*~-objektet.

Nye objekter introdusert i denne øvelsen:

cycle~



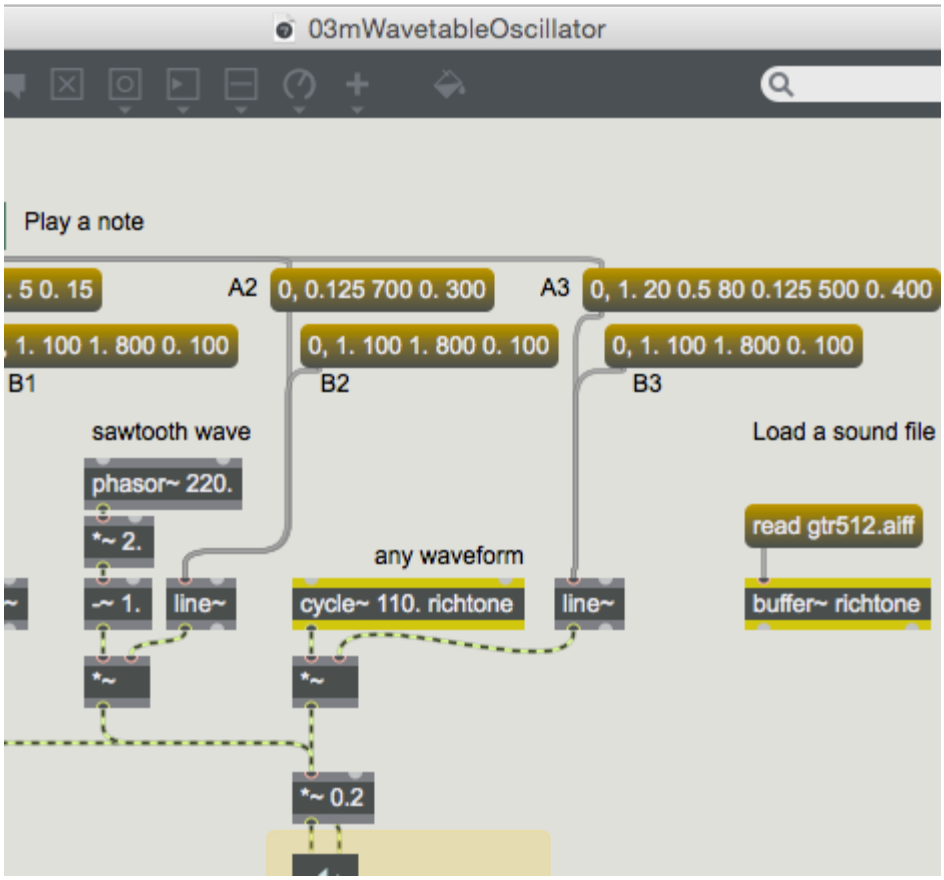
dac~

line

Kontrolloppgave til øvelsen *Wavetable Oscillator*

Denne øvelsen går inn på en viktig synteseform, nemlig bølgetabell-syntese (Hammer s. 48 og Roads s. 91).Vi ser også på hvordan vi kan blande ulike synteseformer for å oppnå ønsket resultat.

Gå gjennom øvelsen *Wavetable Oscillator* 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 36

36.1 Fortsett der du slapp i forrige oppgave med sinusakkorden. Behold den første omhyllingskurven som startet på 0 og gikk til maksimum styrke i løpet av 10 millisekunder for så å gå ned til null igjen på 2 sekunder. Ta bort den andre omhyllingskurven.

36.2 Legg til en sagtannoscillator med samme frekvens som grunntonen i akkorden din (261,6 Hz). Denne skal ha samme omhyllingskurve som sinusakkorden.

36.3 Legg til den samme bølgetabelloscillatoren (Wavetable Oscillator) som du har sett på i øvelsen, altså den som bruker lydfilen «gtr512.aiff» som bølgetabell. Denne bølgetabelloscillatoren skal også ha samme omhyllingskurven som sinusakkorden. Lydfilen «gtr512.aiff» skal settes som et argument til bufferen.

36.4 Legg til slutt til en hvitt støy-kilde (white noise) som har følgende omhyllingskurve: Lydstyrken skal starte på 0 og gå til maksimum styrke i løpet av 5 millisekunder for så å gå ned til null igjen i løpet av 15 millisekunder.

Nye objekter introdusert i denne øvelsen:

buffer~

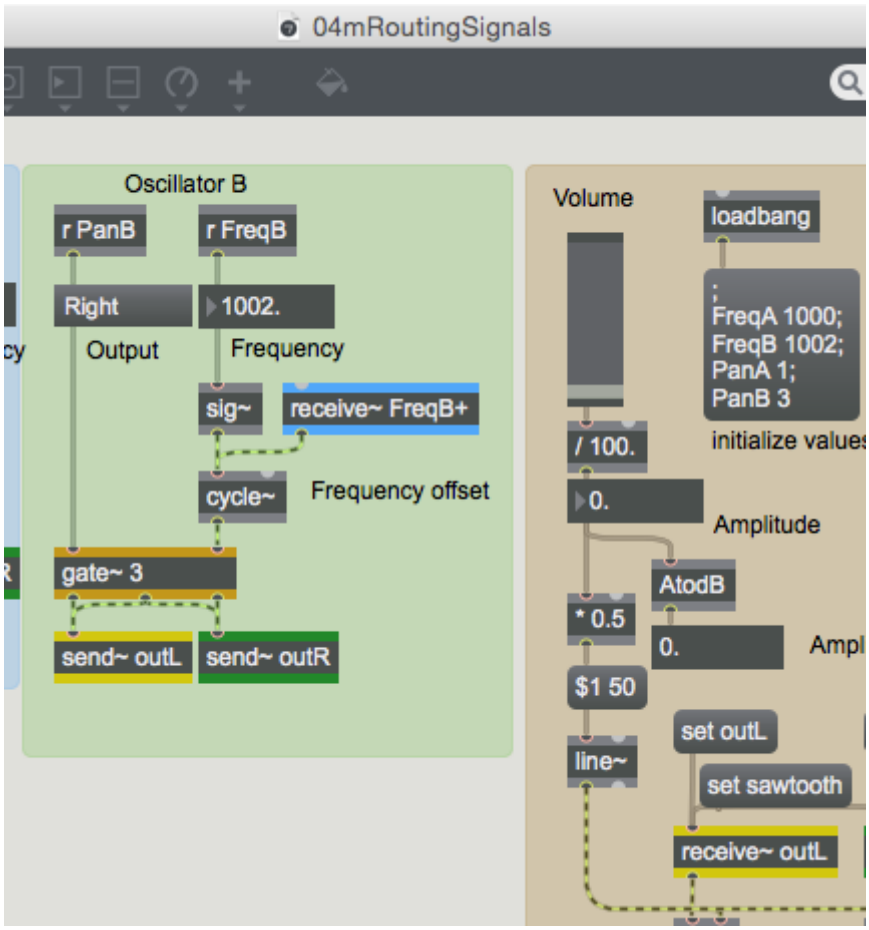
noise~

ezdac~

Kontrolloppgave til øvelsen *Routing Signals*

Denne øvelsen går igjennom viktige objekter som *send*~ og *gate*~ samt bruk av «*line*~»-objektet til å lage glissando.

Gå gjennom øvelsen *Routing Signals* 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 37

Ta utgangspunkt i sinusakkorden fra tidligere oppgaver:

37.1 Lag en rutine som starter med å spille de 4 tonene i sinusakkorden, men som gradvis glir ned til en felles tone, en enstrøken C (261.6 Hz). Dette skal skje i løpet av åtte sekunder. Dette kan du løse med å bruke *scale*-objektet som du lærte om i øvelsen *Procedural Drawing — Creating procedural code*.

37.2 Ved å trykke på en ny knapp glir de opp igjen til de fire tonene.

37.3 Monter en volumkontroll-slider på sluttutgangen. I stedet for å bruke /~-objektet etter sinusakkorden så bruker du *~-objektet. Dette er den vanlige måten å gjøre dette på siden *~-objektet bruker litt mindre cpu.

37.4 Bruk kombinasjonen av *umenu*-objektet og *gate*~-objektet for å velge mellom å dirigere lyden til venstre eller høyre kanal.

Nye objekter introdusert i denne øvelsen:

sig~

send~

receive~

gate~

phasor~

Kontrolloppgave til øvelsen *Basics Review*

Denne øvelsen består av tre mindre deløvelser som oppsummerer mange av de aspektene av MSP som vi har gått gjennom frem til nå. Den kan også brukes av deg til å evaluere om du har fått med deg de grunnleggende teknikkene.

Gå gjennom øvelsen *Basics Review* 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 38

MSP Tutorial 6 består av tre ulike deløvelser. Løs alle tre, og legg de inn i separate subpatcher så det blir mer oversiktlig. Gjør følgende endringer:

38.1 I øvelse 1 forandrer du til 15 stk. 1-sekunds lange noter i stedet for 10.

38.2 I øvelse 2 skal det fades inn en sinusoide fra 4. deltone i stedet for 3. deltone.

38.3 I øvelse 3 forandrer du lydfilen i bufferen fra «gtr512.aiff» til «bd+hh.aiff». Siden disse to lydfilene ligger i den samme katalogen trenger du ikke bekymre deg for at Max ikke skal finne den nye lydfilen. Husk å gi et nytt navn til bufferen samt riktig navn til bufferen som cycle~-objektet leser fra, slik at du ikke påvirker de to andre øvelsene.

38.4 Skift grunnfrekvens i øvelse 3 fra 329.627533 Hz til 100 Hz. Husk å gjøre nødvendige forandringer slik at denne frekvensforandringen ikke påvirker de to andre øvelsene.

Kontrolloppgave til øvelsen *Additive Synthesis*

Når vi i denne øvelsen sirkler inn *additiv syntese* (Hammer s. 49 og Roads s. 115) begynner det å bli morsomt. Additiv syntese ble udødeliggjort i klassiske komposisjoner som Karlheinz Stockhausens *Studie I* (1953) og *Studie II* (1954). Samtidig er additiv syntese en teknikk som kan være vanskelig å beherske selv i moderne programvaresammenhenger. Gå til: Help-> examples/synths/very-special-guest for en veldig spesiell gjest.

Gå gjennom øvelsen *Additive Synthesis* 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 39

Lag en kirkeklokke med additiv syntese og et grafisk brukergrensesnitt.
På internett finnes det en mengde informasjon om kirkeklokker. På denne siden fant vi en analyse av en klokke fra St. Martins kirke i Bremhill, Wiltshire i Sørvest-England. Klokken er definert ut i fra avvik fra en felles frekvens i hele settet på seks klokker. Denne felles frekvensen er 696.2Hz.

Avviket er definert i cent. Cent er tempererte halvtoner med en underdeling på 100 slik at f.eks. en liten sekund fra c til ciss er 100 cent, mens 50 cent blir midt i mellom c og ciss, altså et intervall på en såkalt kvarttone.

Listen med avvik (fra 696.3 Hz) i cent er som følger:

-2236
-1290
-878
673
875
1210

Om vi regner dette om til ferdige midinotenummer blir det som følger:

54.579002
64.043976
68.163979
83.673981
85.693977
89.043976

Legg merke til at vi bruker MIDI-notenummer med desimaler. Om vi ikke hadde brukt desimaler hadde klokkens overtoner blitt tempererte, slik tonene er på et piano, og mye av karakteren til instrumentet hadde forsvunnet.

Regnet om i Hz blir listen som følger:

191
330
419
1026
1154
1400

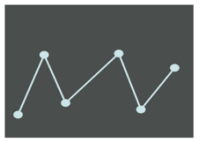
39.1 Med disse seks deltonene er du i stand til å imitere en kirkeklokke med additiv syntese.

39.2 Total varighet: 4000 ms.

39.3 Bruk *function*-objektet for å lage et grafisk brukergrensesnitt. La de øverste deltonene klinge kortest og de dypeste lengst slik de gjør i kirkeklokken i virkeligheten.

Viktig objekt:

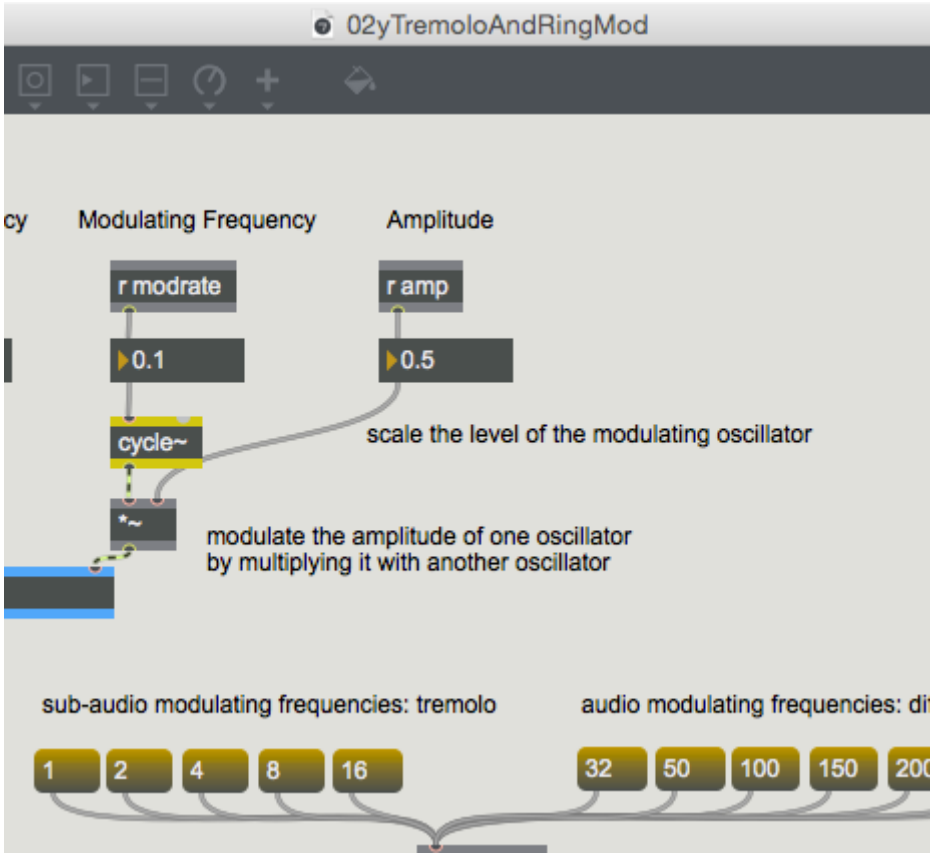
function



Kontrolloppgave til øvelsen *Tremolo and Ring Modulation*

Denne øvelsen går inn på en av de eldste teknikkene i tradisjonell lydbehandling; tremolo og ringmodulasjon (Hammer s. 47 og Roads s. 216).
Ringmodulasjon var mye brukt i den tidlige elektroniske musikken, og er i dag å høre på mange innspillinger innen vidt forskjellige musikkgenre. Et klassisk stykke musikk med ringmodulasjon er Karlheinz Stockhausens *Mantra* (1970) for to klaver og ringmodulasjon.

Gå gjennom øvelsen *Tremolo and Ring Modulation* 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 40

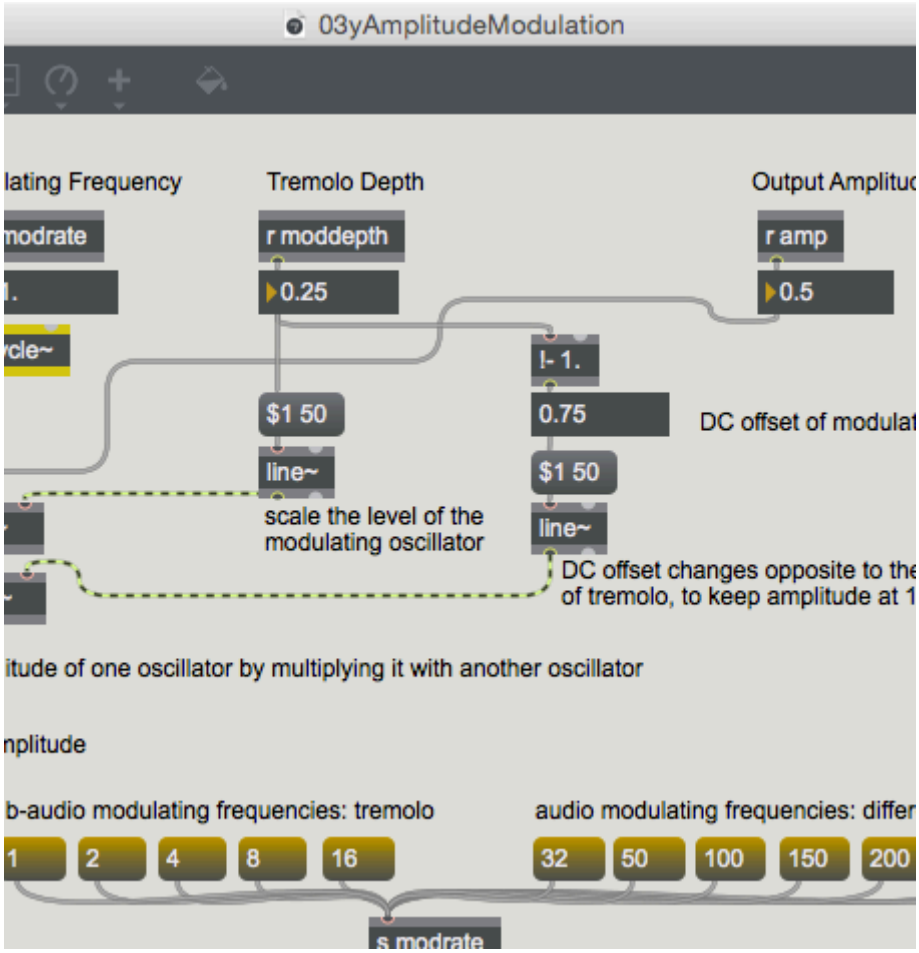
Ta utgangspunkt i klokkeklangen fra forrige oppgave og ringmoduler denne med en frekvens på 100 Hz.

Legg merke til at den metalliske klokkeklangkarakteren beholdes uansett hva slags frekvens du modulerer med. Grunnen er at karakteren til spekteret i en kirkeklokke ikke er harmonisk men består av deltoner som står i et mer komplekst forhold til hverandre. Derfor vil et inharmonisk spekter med ringmodulasjon oppfattes beslektet med det opprinnelige spekteret mens en lydkilde med et harmonisk spekter, f.eks. en menneskestemme eller en gitartone, vil oppleves som svært forvrengt ved bruk av ringmodulasjon.

Kontrolloppgave til øvelsen *AM Synthesis*

Denne øvelsen går videre fra ringmodulasjon til det beslektede fenomenet amplitudemodulasjon (Hammer s. 47 og Roads s. 221). Hovedfordelen ved amplitudemodulasjon kontra ringmodulasjon er at intensiteten av virkningen kan varieres ved å endre amplituden til modulatoren. I oppgaven setter vi flere oscillatorer etter hverandre i serie slik at de modulerer hverandre.

Gå gjennom øvelsen *AM Synthesis* 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 41

Lag en amplitudemodulasjon på samme måte som i øvelsen, men la fire sinusoscillatorer modulere hverandre:

41.1 Start med en sinustone på 440 Hz.

41.2 Moduler denne med en frekvens på 100 Hz og en modulasjonsdybde (Tremolo Depth) på 1.0 (maks amplitude).

41.3 Moduler dette resultatet igjen med en frekvens på 1 Hz og en modulasjonsdybde på 0.3.

41.4 Moduler dette resultatet til slutt med en frekvens på 8 Hz og en modulasjonsdybde på 0.2.

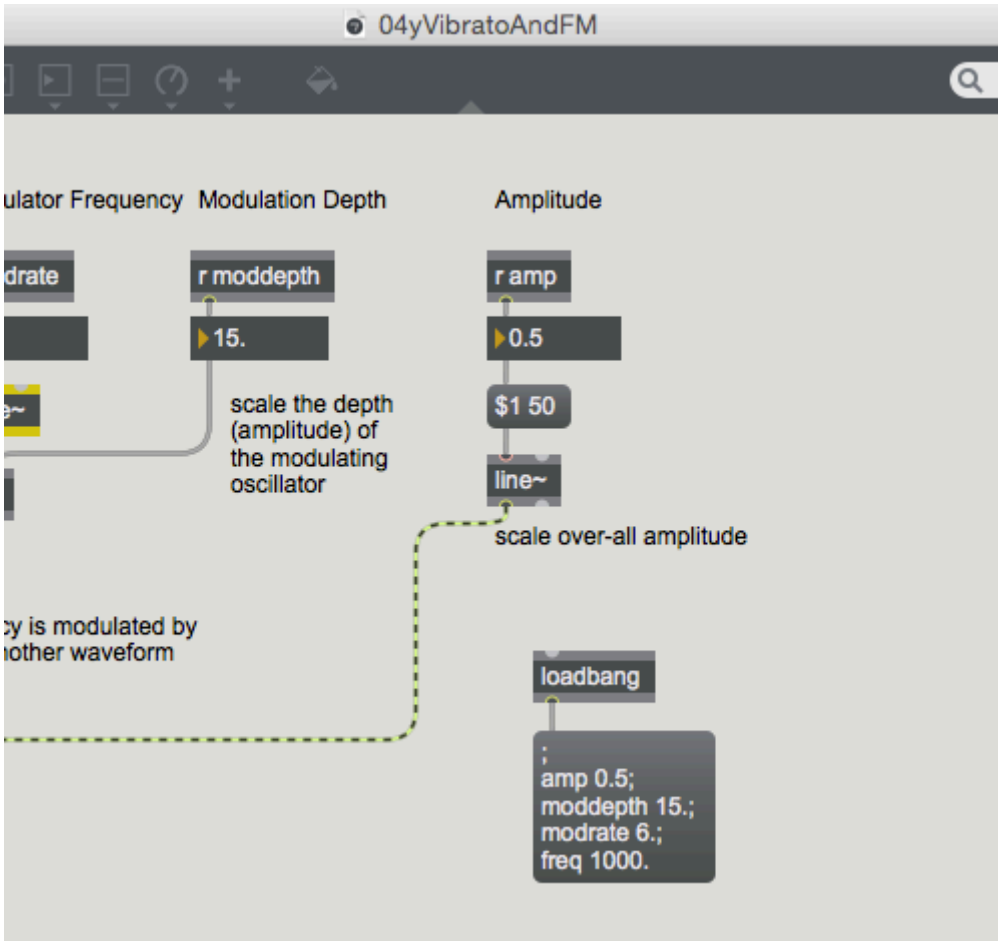
På denne måten får du effekten av en sinusoscillator som blir amplitudemodulert flere ganger, slik at vi får flere pulserende lag i lyden.

Kontrolloppgave til øvelsen *Vibrato and Frequency Modulation*

I den klassiske elektroniske musikken har syntese basert på frekvensmodulasjon, såkalt FM, (Hammer s. 49 og Roads s. 224-250) vært benyttet siden 1960-tallet. Denne teknikken dominerte store deler av det populærmusikalske lydbildet på 1980-tallet, hovedsakelig på grunn av populariteten til Yamahas DX7 synthesizer (1983), for så å nesten forsvinne helt tidlig på 90-tallet. Teknikken fikk deretter en renessanse med en rekke plugin-instrumenter med frekvensmodulasjon.

I denne oppgaven skal vi i tillegg til frekvensmodulasjon også se på hva som skjer når vi kombinerer frekvensmodulasjon med amplitudemodulasjon.

Gå gjennom øvelsen *Vibrato and Frequency Modulation* 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 42

I denne oppgaven skal vi kombinere frekvensmodulasjon og amplitudemodulasjon.

42.1 Lag en frekvensmodulert klang med følgende parameter:

Bærefrekvens (Carrier Frequency): 300 Hz

Modulasjonsfrekvens (Modulator Frequency): 400 Hz

Modulasjonsdybde (Modulation Depth): 1300 Hz

42.2 Amplitudemoduler den frekvensmodulerte klangen med samme type amplitudemodulasjon som du brukte i forrige øvelse, men med følgende parameter:

Modulasjonsfrekvens: 250

Modulasjonsdybde (Tremolo Depth): 0.5

42.3 Moduler denne lyden igjen med en amplitudemodulasjon med følgende parameter:

Modulasjonsfrekvens: 1

Modulasjonsdybde: 0.3

42.4 Moduler denne lyden igjen med en amplitudemodulasjon med følgende parameter:

Modulasjonsfrekvens: 8

Modulasjonsdybde: 0.15

På denne måten får du effekten av en frekvensmodulert klang som igjen blir klanglig forvrengt av en amplitudemodulasjon som igjen blir modulert med lav frekvens to ganger, noe som gir flere pulserende lag i den relativt komplekse lyden.

Kontrolloppgave til øvelsen *FM Synthesis*

I denne øvelsen går vi inn på de litt mer komplekse variantene av frekvensmodulasjon (Hammer s. 49 og Roads s. 224-250).

I den forrige oppgaven gjorde vi en enkel frekvensmodulasjon der vi definerte bærefrekvens (Carrier Frequency), modulasjonsfrekvens (Modulator Frequency) og modulasjonsdybde (Modulation Depth). Denne metoden gir mange interessante resultater, problemet er bare at parametrene *bærefrekvens, modulasjonsfrekvens* og *modulasjonsdybde* ikke sier så mye om det faktiske klanglige resultatet.

Derfor er det mer vanlig å oppgi parametrene i frekvensmodulasjonen som bærefrekvens (Carrier Frequency), harmonisitetsforhold (Harmonicity ratio) og modulasjonsindeks (Modulation index), der harmonisitetsforholdet er Fm/Fc og modulasjonsindeksen er Am/Fm. De ulike parametrene utrykkes altså i forhold til hverandre. Vi skal ikke gå inn på de ulike formalitetene i frekvensmodulasjon her men heller se på de faktiske resultatene av disse parametrene. (for de som ønsker å fordype seg i dette kan Hammers og Roads' tekst være en grei start).

Harmonisitetsforholdet definerer hvilke frekvenser som er tilstede i lyden og hvorvidt de står i et harmonisk eller uharmonisk forhold til hverandre, mens modulasjonsindeksen påvirker «skarpheten» i klangen ved å påvirke den relative styrken i overtonene.

Litt forenklet kan vi derfor si at:

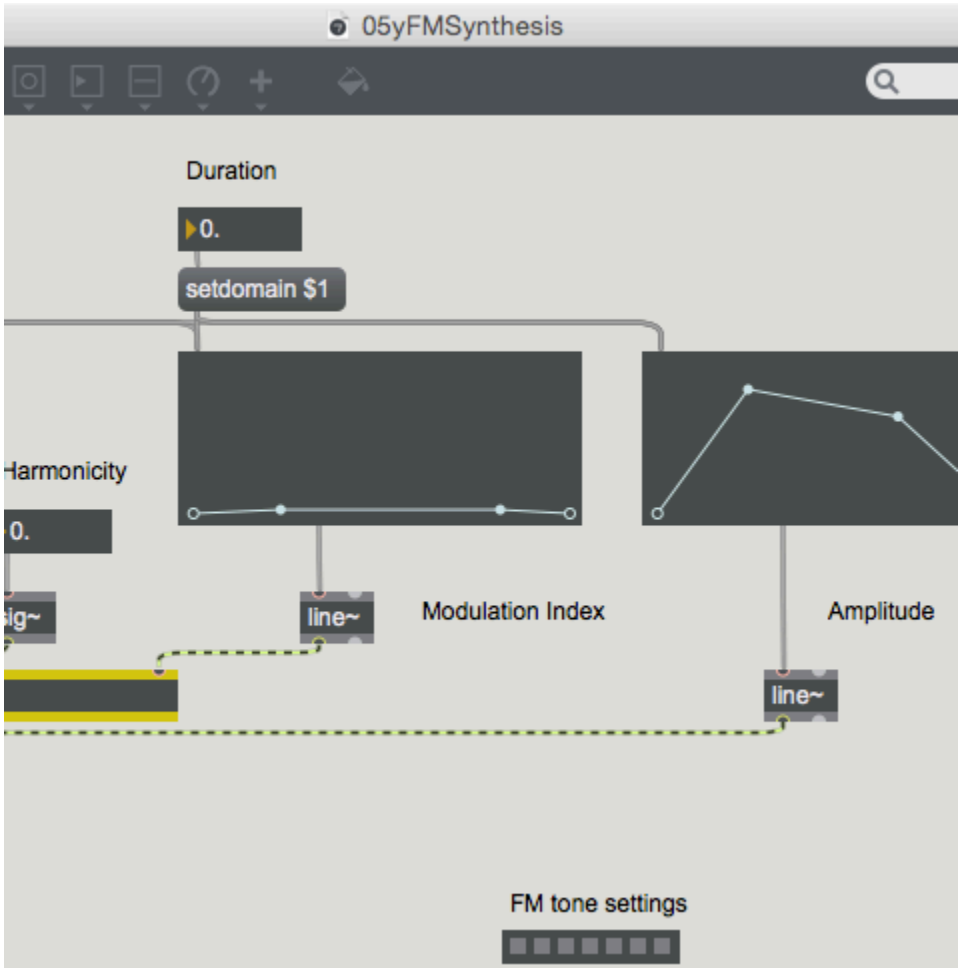
Bærefrekvensen gir grunntonen.

Modulasjonsindeksen bestemmer hvor mye diskant vi skal ha i klangen.

Harmonisitetsforholdet bestemmer hvor forvrengt spekteret skal være.

En simulering av «enkle» lyder som en gitartone (Hammer s. 31) eller en menneskestemme (Hammer s. 35) kan ha et harmonisitetsforhold på 1, mens en simulering av en uharmonisk klang som en kirkeklokke kanskje heller bør ha et harmonisitetsforhold på 1.64. For å gjøre kirkeklokkeeksempelet komplett lager vi en omhyllingskurve (envelope) på modulasjonsindeksen slik at den starter med en høy verdi og ender med en lav verdi. På denne måten kan vi simulere anslaget på kirkeklokken.

Gå gjennom øvelsen *FM Synthesis* 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.

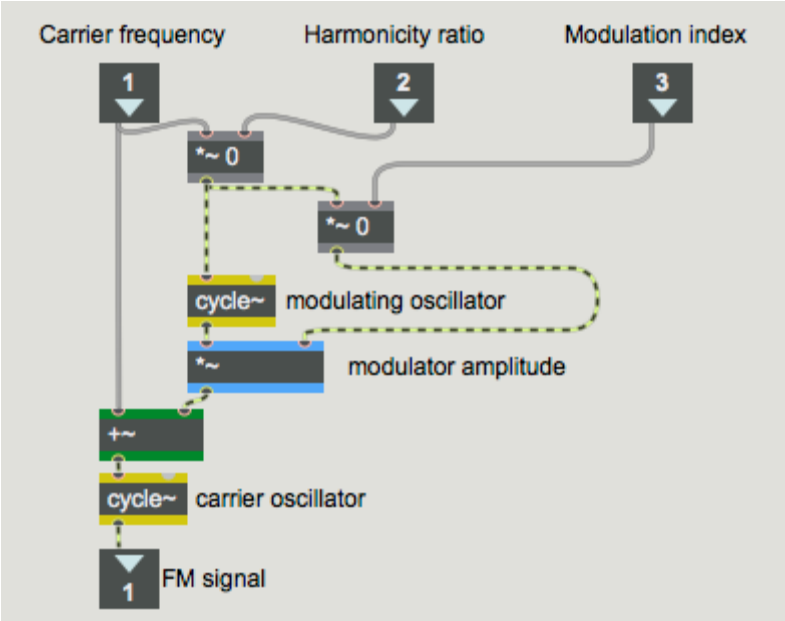


Gå igjennom de ulike delene av patchen og vær sikker på at du skjønner hvordan alle delene fungerer. Når alt er klart, gå videre til oppgaven.

Oppgave 43

Om du dobbeltklikker på abstraksjonen *simpleFM~* så ser du at denne frekvensmodulasjonen er nokså enkelt oppbygd, og bare består av to sinusoscillatorer, tre multiplikasjonsobjekter og et addisjonsobjekt.

All prosessering foregår selvsagt i lyddomenet siden alle verdiene må oppdateres på sampleratenivå for å unngå ulike former for forvrengninger og bieffekter.



Vi skal som sagt ikke gå inn på alle tekniske aspekter her men holde oss til bruk av de tre parametrene bærefrekvens, harmonisitetsforhold og modulasjonsindeks.

Ved hjelp av disse tre parametrene, lag en patch med presets som simulerer følgende fire lyder:

1. kirkeklokke
2. cembalo
3. nylonstreng gitar
4. romskip som lander

BEAP — virtuell analog syntese

Store deler av konseptet bak BEAP er inspirert av modulære analoge synthesizere. Inngangen for å styre parametrene kalles for eksempel for *CV*. *CV* står for *Control Voltage* (kontrollspenning) og er en betegnelse som er vanlig i forbindelse med analoge synthesizere. For å styre lydstyrke bruker vi en BEAP-patch som kalles *VCA* (Voltage Controlled Amplifier). Dette er også en betegnelse som blir brukt i synthesizerterminologien. Selv om BEAP er en del av et dataprogram er mye av konseptet hentet fra modulære analoge synthesizere. I denne øvelsen skal vi simulere en enkel analog synthesizer.

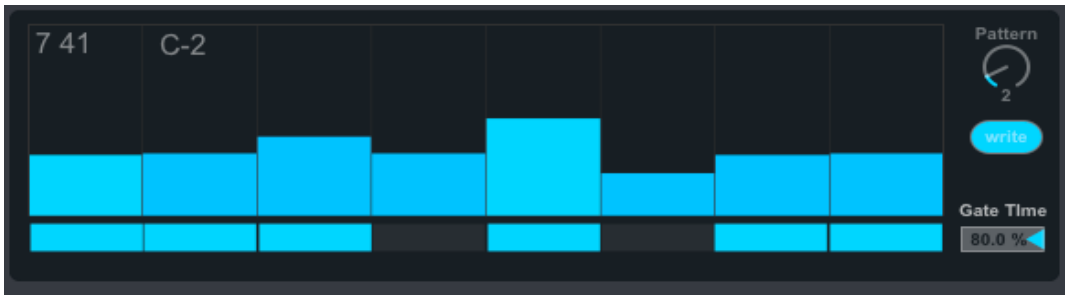
Start med å koble en BEAP *Oscillator*-patch til en BEAP *Stereo Output*-patch. BEAP *Oscillator*-patchen simulerer en analog oscillator og har de tradisjonelle bølgeformene sinus, trekant, sagtann og firkant. Den kan også velge bredden på pulsen i firkantbølgen med parameteret *PW* og dette kan moduleres eksternt med pulsbreddemodulasjon (PWM). *Offset* styrer tonehøyden.

For å styre tonehøyden på oscillatoren utenfra kan vi bruke en «sequencer». gå til *Sequencer*-kolonnen og velg *Sequencer*-patchen. Koble *CV* fra BEAP *Sequencer*-patchen til *CV1*-inngangen på BEAP-oscillatoren. For å starte sequenceren kan du bruke *Global Transport*-patchen som også ligger under *Sequencer*-kolonnen. *Global Transport* trenger ikke kobles til noe.

BEAP *Sequencer*-patchen simulerer en såkalt analog step sequencer. Du skrur den av og på med *Global Transport*. Dra i sliderene for å forandre på verdiene i sekvensen.

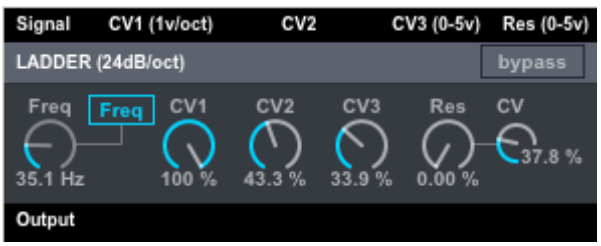
For at lyden ikke skal være på hele tiden kan vi bruke en omhyllingskurve (*Envelope*).
Velg *AD* (Attack Decay) fra *Envelope*-kolonnen. Koble *Gate* ut fra sequenceren til *Gate* inn på *AD*. Hent deretter en *VCA* fra *Level*-kolonnen. Koble *Signal* ut fra *AD* til *CV* inn på *VCA*.
Koble *Signal* ut fra oscillatoren til *Signal* inn på *VCA*.
Koble *VCA Output* til *Left* og *Right* inngangen på *Stereo*-patchen. Sequenceren trigger nå omhyllingskurven (*AD*) som styrer lydstyrken til oscillatoren ved hjelp av *VCA*. Eksperimenter med ulike settinger på *Attack* og *Decay* til *AD*.

BEAP *Sequencer*-patchen kan skru *Gate*-signalet som trigger *AD*-envelopen av og på uavhengig av tonehøydene den gir til oscillatoren. Dette gjøres i den nederste raden:



Legg merke til at vi i eksempelet over har åtte ulike verdier med tonehøyder i den øverste raden, men at vi i to av verdiene ikke har valgt en tilsvarende trigger i den nederste *Gate*-raden.

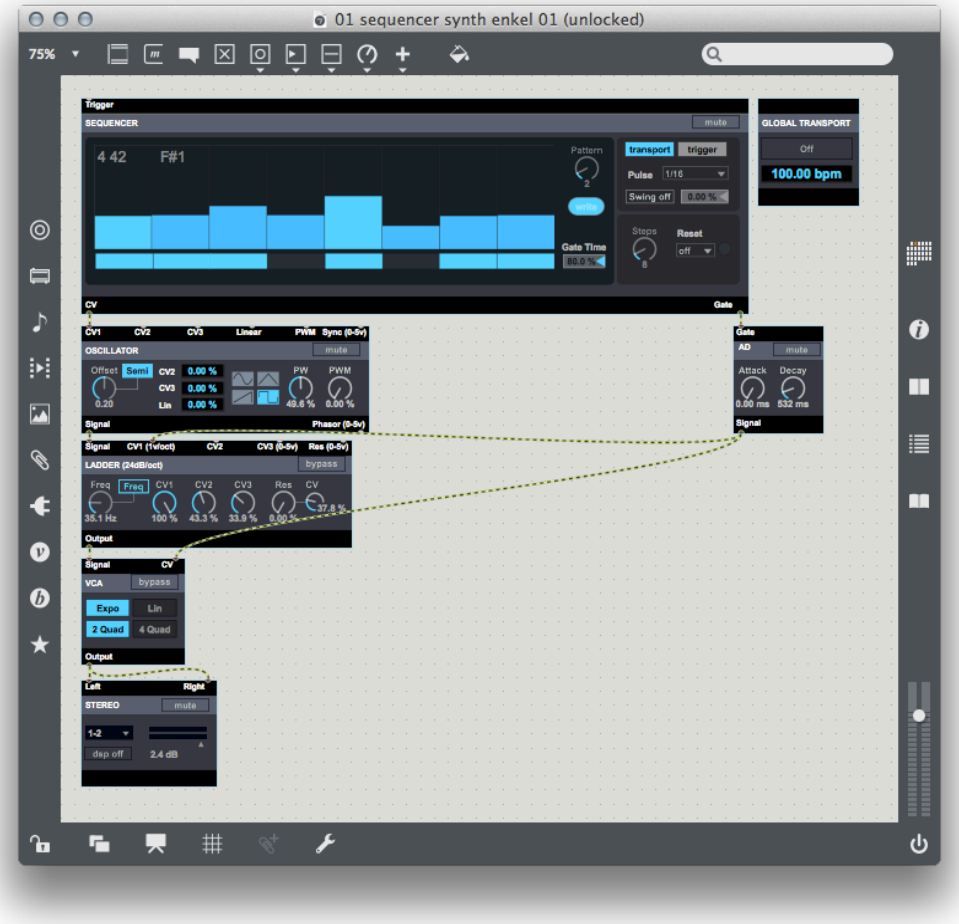
Fortsett med å koble til et filter. Bruk BEAP filter patchen *Ladder* som ligger i *Filter*-kolonnen. *Ladder* er modellert etter et såkalt Moog Ladder filter, det samme som blant annet sitter i Minimoog synthesizeren.



Koble *Signal* ut fra oscillatoren inn på *Signal* inn på *Ladder*-patchen, og videre ut fra *Output* i *Ladder*-patchen inn til *VCA*. Velg mellom *Semi* og *Freq* i filteret for å kunne stille inn filterfrekvensen i halvtoner eller Hertz. Det mest praktiske er ofte å bruke Hertz. Ved å skru på *Freq*-knappen kan du forandre filterfrekvensen.

Du kan også bruke en ommhyllingskurve (*envelope*) til å styre filterfrekvensen. Lag en kobling fra *AD Signal* utgangen til *CV1 (1v/oct)*-inngangen i *Ladder*-patchen. Skru filterfrekvensen til *Freq* knappen ned til 35 Hz og skru knappen merket med *CV1* helt opp til 100%.

Husk at du må velge «Autosave Snapshot» for å lagre innstillingene. I denne patchen trenger vi ikke å bruke «Embed Snapshot in Parent» siden vi ikke har noen BEAP-patch som skal huske hvilken lydfil du har valgt.



Hele patchen