Max-delen (CC Max)

Dette afsnit indeholder en kort introduktion til Max-progammeringsmiljøet samt en opridsning af, hvilken rolle det spiller i vores projekt. Hvilke målsætninger har vi haft med Max-delen, og hvilke udfordringer er vi stødt på undervejs?

Introduktion til Max med mere

Max er et udpræget grafisk baseret progammeringsmiljø, som er specielt rettet mod udviklingen af interaktive lydlige artefakter samt multimedieartefakter mere generelt[[1]](#footnote-2). Dette være sig konstruktion af synthesizere, diverse multimedieinstallationer, plugins til sequencer-applikationer, applikationer som en slags lydlige kompositioner, og så videre...

Max-miljøet indeholder en mangfoldighed af klasser/moduler, som i brug manifesterer sig som grafiske objekter med tilhørende ind- og udgange. Disse objekter kan forbindes via grafisk repræsenterede signalveje, eller der kan skabes 'usynlige' signalveje. Det er også muligt at skabe sine egne klasser, som derefter kan benyttes på linje med de fra 'fabrikken' medfølgende.

Det man skaber, når man laver en ny applikation eller en klasse, kaldes i Max en 'Patch'. En patch kan sagtens indeholde et hav af andre patches, på samme måde som f.eks. et objekt i Java kan indeholde referencer til andre objekter. Max-miljøet adskiller sig dog, i kraft af sin meget grafiske fremtræden, en hel del fra det at programmere i skriftbaserede sprog. Med sine meget udviklede moduler/klasser falder det at programmere i Max ind under en kategori, som ofte kaldes 'high level programming', og det må også siges at være noget lettere at gå til end eksempelvis Java. Dog skal det nævnes, at den grafiske opbygning har sine begrænsninger – der er ting, som er vanskelige at lave grafisk. Derfor understøtter Max også en form for scripting-beskeder, og dette er noget vi har benyttet os en del af i vores udvikling af CrowdControl.

Gennem tiden er der kommet mere funktionalitet til Max. Herunder er de mest betydningsfulde 'add-ons' Max/Msp-pakken, som tillader audio-signalbehandling og Max/Jitter-pakken, som tilføjer funktionalitet til video-signalbehandling.

Vil man redigere en patch/ lave en ny patch, er det nødvendigt at have Max installeret på den pågældende computer. Max er desværre ikke gratis men et kommercielt produkt, som distribueres af firmaet Cycling '74[[2]](#footnote-3). Dog er det muligt at afvikle en Max-patch via Max-runtime -applikationen, som er gratis og frit distribueret. Således kan man sagtens afvikle CrowdControl og loade samt konfigurere patches uden at have adgang til Max.

Vores målsætninger

Vores primære målsætning med Max-delen af CrowdControl har været at skabe en applikation (CC Max), som kan fungere som en form for ramme eller 'host', hvori man kan loade forskellige patches ind. Det er således de enkelte loadede patches, som står for den grundlæggende indsamling og behandling af input fra sensorer (vi vil referere til disse load-bare patches som 'controller patches'). Et eksempel kunne være en controller patch, som tager et input fra et kamera og genererer to outputsignaler ud fra dette input. Disse behandlede signaler er derefter tilgængelige for ramme-applikationen (og dermed også CC Flash), som kan konfigureres til at omforme og udsende signalerne som midi-data.

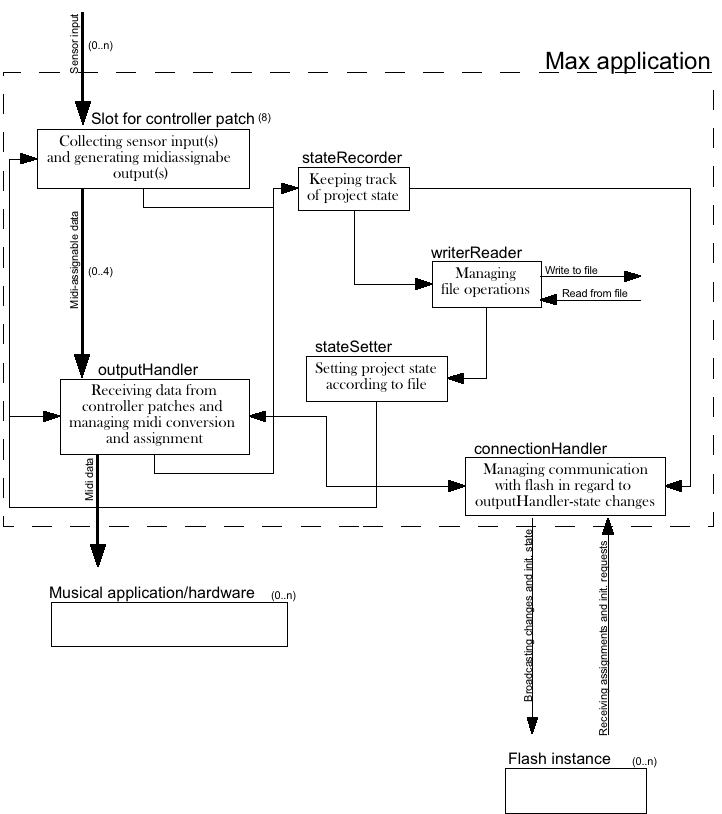
Det har været vigtigt for os at understøtte det at kunne gemme det projekt, man har arbejder på i CC Max, således at alle referencer til loadede controller patches, disses konfiguration/tilstand samt indstillinger i vinduet 'Output configuration' kan skrives til en fil. De forskelige projekt-filer skal naturligvis også kunne loades, således at alt er som da man gemte projektet.

Det har ligeledes været et mål for os at holde så meget af funktionaliteten i denne ramme-applikation som muligt, så de enkelte controller patches ikke behøver 'bekymre sig' om eksempelvis kommunikationen med flash, midi-konfiguration o.s.v.. En af grundene til dette er, at vi ønsker at det skal være så let som muligt at skabe en ny patch, der er kompatibel med systemet og dermed kan loades ind i et projekt.

Rent progammeringsmæssigt har vi søgt at benytte en objektorienteret tilgang, i den udstrækning dette er muligt i Max. Helt strengt forstået kan man muligvis ikke tale om egentlig objektorienteret programmering – et max-objekt har eksempelvis ikke metoder i klassisk forstand. Men i hvert fald er det muligt at skabe genbrugelige moduler, hvis instantieringer kan modtage parametre i deres initialisering. Desuden har vi navngivet/kommenteret alt på engelsk for at imødekomme vores tankegang om et Free Software -produkt, som det skal være muligt at videreudvikle på.

Udfordringer

Undervejs i vores proces har vi mødt en del udfordringer i forhold til at realisere de ovennævnte målsætninger. En af disse udfordringer var at realisere funktionaliteten til at kunne gemme og loade projektfiler, hvilket jo indebar, at vi kunne føre protokol over alle relevante tilstande. Hvordan skulle vi kunne holde styr på den interne tilstand for de enkelte loadede controller patches? - patches som vi ikke nødvendigvis selv har konstrueret. Dette har været et af de aspekter, vi har arbejdet en del med. Løsningen har været at skabe en skabelon for controller patches indeholdende en anelse grundfunktionalitet (så lidt som muligt i henhold til vores tanker om at holde mest funktionalitet i ramme-applikationen). Det drejer sig om en lille anordning, som gør skabelonen i stand til at navngive og holde styr på alle de bruger-konfigurerbare elementer, som måtte blive tilføjet til skabelonen. Således kan enhver controller patch, som indeholder skabelon-funktionaliteten, blive 'bedt om' at rapportere sin tilstand til det modul i ramme-applikationen, vi kalder stateRecorder. StateRecorder'en fører også protokol over tilstanden for vinduet 'Output configuration' (styret af outputHandler), om end dette fungerer på en lidt anden måde. OutputHandler'en bliver (undtaget ved initialiseringer af flash-instanser) ikke 'bedt om' at rapportere sin tilstand, men udsender derimod selv enhver tilstandsændring. Dette skyldes, at tilstandsændringer i Output configuration skal registreres med det samme og 'broadcastes' ud til eventuelle flash-instanser. Denne broadcasting varetages af modulet connectionHandler, som også modtager forespørgsler og kommandoer fra flash-instanser. I diagrammet herunder er hovedlinjerne i denne tilstandshåndtering skitseret.

En anden udfordring har været at få kommunikationen med flash til at fungere. Både mht. til det tekniske omkring Flosc-serveren (som beskrevet tidligere), men også mht. det at få defineret opbygningen af de beskeder, der skal sendes gennem denne server og ikke mindst, hvornår der skal sendes hvilke beskeder[[3]](#footnote-4).

Den måske største udfordring vi er stødt på i løbet af vores proces vedrører designet af de enkelte controller patches og i det hele taget det at designe meningsfulde interaktionsformer rettet mod en folkemængde. I og med at vi har stræbt mod at designe et åbent artefakt, hvor der lægges op til, at man nemt kan skabe sine egne interaktionsformer og controller patches, har vi lagt en del af dette ansvar over på brugerne. Dog kan vi ikke løbe fra, at vi har ansvaret for at levere et solidt udgangspunkt for videreudvikling samt at illustrere og argumentere for CrowdControls grundlæggende eksistensberettigelse. Derfor har vi konstrueret hvad man kunne kalde for en række 'factory patches', som er controller patches, der følger med CrowdControl. På denne måde er det let og hurtigt at komme i gang med produktet – også selv om man hverken har kendskab til eller adgang til Max-programmeringsmiljøet. Nogle af disse factory patches er temmelig specifikke, mht. hvilken interaktionsform de sigter mod, mens andre er mere åbne og generelle. Alle kan de dog konfigureres mere specifikt, hvis dette ønskes eller bruges som et udgangspunkt til nye patches. Som sagt er det ikke en let opgave at designe gode interaktionsformer til en koncertsituation, og derfor bliver dette også en stor udfordring for fremtidige udviklinger af CrowdControl – at udforske og skabe nye idéer til interaktionsformer, manifesteret i controller patches og sensoropstillinger.

Opsummering

Vi har i Max-delen af CrowdControl forsøgt at designe et artefakt, som på den ene side er let at komme i gang med, men som samtidig er åben for meget specifik konfiguration og opførsel. Vi har ikke haft en falsk forhåbning om, at vi kunne skabe et fuldstændig lukket og 'færdigt' artefakt, som samtidig ville være tilfredsstillende at bruge i en kreativ sammenhæng. Derfor har den åbne og modulbaserede[[4]](#footnote-5) tilgang været essentiel for os. Vores erfaringer med at programmere i Max-miljøet har generelt været positive, om end vi har fundet, at visse ting kan være svære at realisere i et sådant grafisk orienteret miljø. Det at skabe/loade nye objekter på run-time kræver eksempelvis, at man benytter sig af en form for scripting-beskeder, og netop denne scripting-del af Max er ikke særlig fyldestgørende dokumenteret.

1. Den stærke lydlige orientering afspejler sig også i navnet (Max) idet det er opkaldt efter Max Matthews som er en pionér indenfor computermusik. -Kilde: http://en.wikipedia.org/wiki/Max\_Mathews [↑](#footnote-ref-2)
2. http://www.cycling74.com/ [↑](#footnote-ref-3)
3. Se diagrammet i afsnittet om flash-delen for flere detaljer om beskederne. [↑](#footnote-ref-4)
4. Altså det at man kan indloade eksterne moduler (controller patches). [↑](#footnote-ref-5)