

1. **Max — De fundamenten**
    1. 1. Max, MSP & Jitter
    1. 2. De Max-interface
    1. 3. Objecten en patch cords
    1. 4. De volgorde van uitvoering
    1. 5. Rekenen en logica
    1. 6. Arguments en attributes
    1. 7. Veelgebruikte Max-objecten
2. **MSP — Audio in Max**
    2. 1. Het principe van een audio signal chain
    2. 2. Soundfiles afspelen
    2. 3. Audio in
    2. 4. Audio Analysis
3. **Vizzie — Snel aan de slag met Jitter**
    3. 1. De basis
    3. 2. Patching en verbindingen
    3. 3. Een eerste Vizzie-chain
    3. 4. Interface en stuurdata
    3. 5. Vizzie-chain 2: modulatie en automation
    3. 6. Vizzie-chain 3: live control
    3. 7. Vizzie-chain 4: audio input
    3. 8. Vizzie-chain 5: audiovisueel instrument
4. 🚧 **Jitter** 🚧
5. **Verder leren**

1. Max — De fundamenten

Max is een visuele programmeeromgeving voor het bouwen van interactieve systemen en audiovisuele toepassingen. Je programmeert door objecten met elkaar te verbinden via *patch cords*. Zo ontstaat een netwerk, een zogenaamde *patcher*, waarin objecten gegevens uitwisselen en op elkaar reageren.

Met Max kun je geluid genereren en bewerken, beeld en video manipuleren, interactieve installaties bouwen en externe hardware aansturen.

In dit eerste deel verkennen we de basisprincipes van Max: de onderdelen, werkomgeving, objecten, dataflow, messages, getallen, triggers en logica. Het belangrijkste is dat je leert begrijpen hoe objecten met elkaar communiceren en hoe de gegevensstroom binnen een patcher werkt.

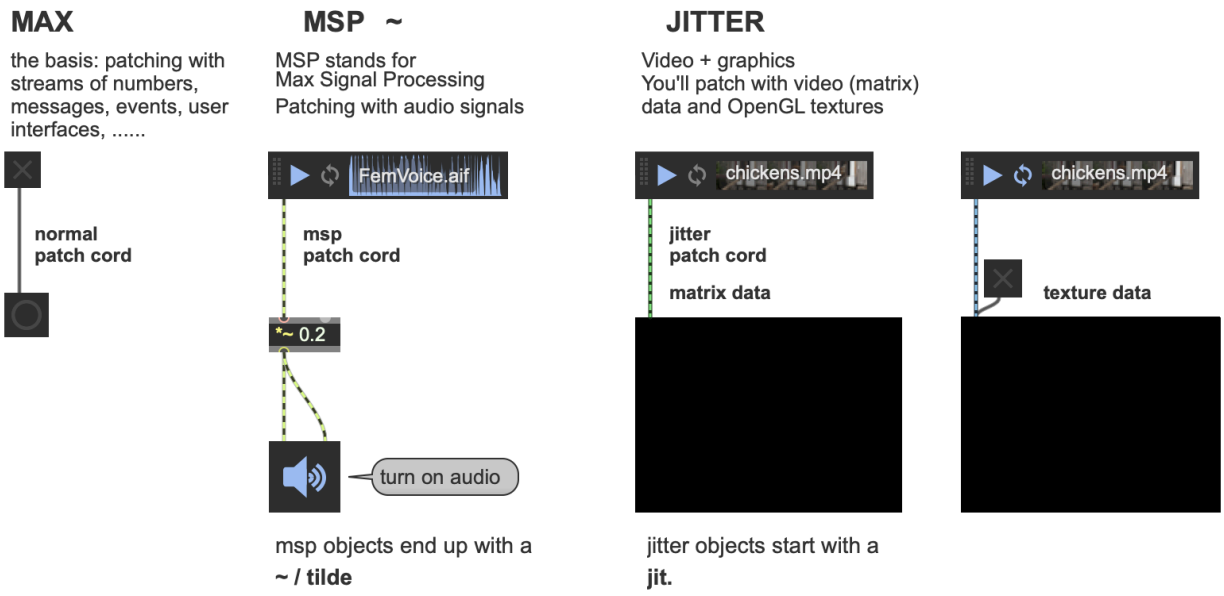
1.1. Max, MSP & Jitter

Max bestaat uit drie belangrijke onderdelen:

- **Max** vormt de basisomgeving. Je werkt met data, messages, getallen, triggers, logica en interactie. Het is de omgeving waarin je de algemene structuur en werking van je patches opbouwt.
- **MSP** (*Max Signal Processing*) is het audiogedeelte. Hiermee kun je audio opnemen, afspelen, genereren, analyseren en bewerken. MSP bevat onder andere de bouwstenen voor synthesizers, samplers en audio-effecten. **MSP-objecten** herken je aan het tilde-teken (`~`) in hun naam. Ze verwerken audio als een continue stroom van samples.
- **Jitter** is gericht op beeld en video. Je gebruikt het om afbeeldingen, videostreams en animaties te genereren, analyseren en manipuleren. **Jitter-objecten** beginnen met `jit.` en verwerken beeld als matrices of textures, frame per frame in realtime.

Het type data dat door een patch cord stroomt, herken je aan de kleur en vormgeving. Control data, audiosignalen, Jitter-matrices en Jitter-textures hebben elk hun eigen visuele weergave.

Omdat audio- en beeldsignalen continu worden verwerkt en grote hoeveelheden data bevatten, vragen ze doorgaans meer rekenkracht dan gewone control data.



Naast deze drie onderdelen beschikt Max over verschillende uitbreidingen en aanvullende omgevingen, zoals **Vizzie**, **BEAP**, **Node for Max**, **Max for Live** en **Gen** maar daarover later meer.

1.2. De Max-interface

De user interface van Max is opgebouwd rond een klassieke menustructuur, aangevuld met verschillende toolbars en panelen rondom het *patcher*-venster. Dat venster vormt het centrale werkgebied waarin je objecten plaatst en met *patch cords* met elkaar verbindt.

De verschillende toolbars

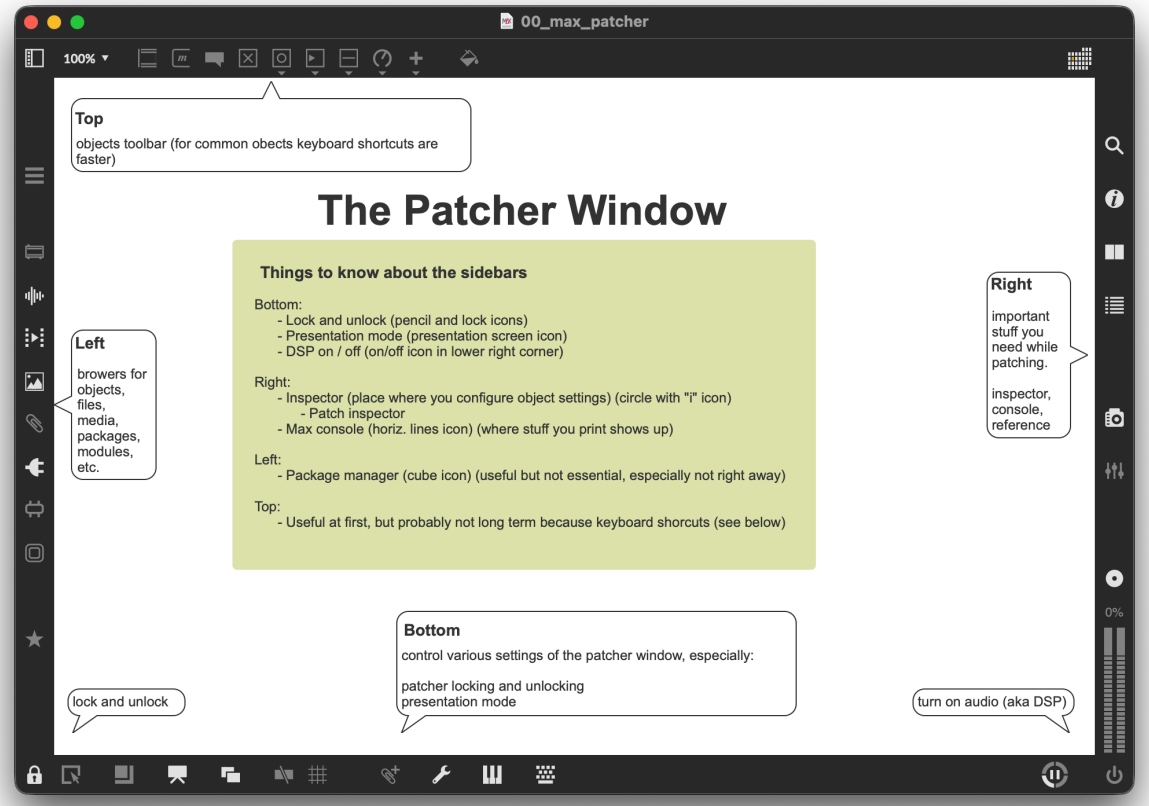
- **Boven:** toegang tot objecten, messages, comments en verschillende UI-objecten, ingedeeld volgens functie of categorie.
- **Onder:** functies voor het beheren en aanpassen van je patcher, zoals Presentation Mode, Debugging, Transport en Audio.
- **Links:** toegang tot browsers voor objecten, media, plug-ins, modules, …
- **Rechts:** toegang tot verschillende hulpmiddelen en panelen, zoals de Inspector, Reference, Max Console en Snapshots.

Edit en Lock mode

Je werkt in Max in twee verschillende modi:

- **Edit mode (ontgrendeld):** je bouwt en wijzigt de patcher. Je kunt objecten toevoegen, verplaatsen, verwijderen en verbinden via patch cords.
- **Lock mode (vergrendeld):** je gebruikt de patcher. UI-elementen zoals buttons, sliders, dials en number boxes worden interactief.

Je schakelt tussen beide modi via het hangslot-icoon, met **Ctrl+E** (macOS: ⌘E) of met **Ctrl-klik** (macOS: ⌘-klik) op de achtergrond van de patcher.



Help en Reference

Elk Max-object beschikt over een eigen **Help-file** met uitleg en voorbeelden. Zo ontdek je niet alleen wat een object doet, maar ook hoe je het in een patcher kunt gebruiken.

Je opent de Help-file door **Alt-klik** op een object, of door met de rechtermuisknop op een geselecteerd object te klikken en **Open Help** te kiezen.

De **Reference** geeft een meer technische beschrijving van een object, met informatie over de inlets, outlets, messages, arguments en attributes.

1.3. Objecten en patch cords

Max werkt volgens het principe van *dataflow programming*. Een patcher bestaat uit **objecten** die elk een specifieke taak uitvoeren. Sommige objecten verwerken getallen, andere voeren berekeningen uit, genereren triggers, spelen geluid af, manipuleren beeld, ...

Je maakt een verbinding tussen objecten door hun poorten met elkaar te verbinden via een **patch cord**. De poorten aan de bovenkant of zijkant van een object zijn de **inlets** (ingangen); de poorten onderaan zijn de **outlets** (uitgangen).

Sommige objecten hebben een gebruikersinterface (UI), zoals buttons, toggles, sliders, dials en number boxes. Daarmee kun je waarden rechtstreeks bedienen of weergeven. Andere objecten hebben geen zichtbare UI en voeren hun taak op de achtergrond uit.

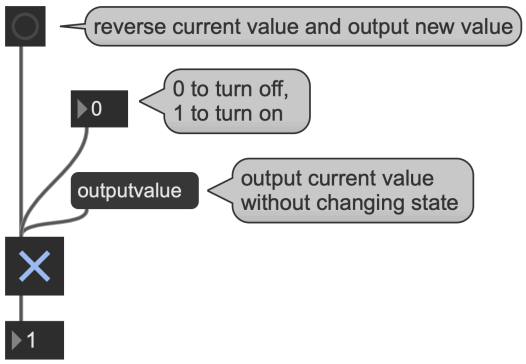
Een patcher ontstaat dus niet zomaar uit een verzameling objecten: de verbindingen bepalen hoe de objecten samenwerken en gegevens uitwisselen.

Messages, getallen en triggers

Objecten kunnen verschillende soorten informatie ontvangen en doorgeven. Dat kan bijvoorbeeld een getal zijn, maar ook een *message* of een *trigger*.

- **Getallen** geven een numerieke waarde door, bijvoorbeeld `5` of `0.75`.
- **Messages** sturen informatie, waarden of opdrachten naar een object. Een message-object kan bijvoorbeeld een commando versturen wanneer je erop klikt.
- **Bang** is een trigger die een object activeert. Je kunt het zien als een signaal dat aangeeft: voer nu je taak uit.

De manier waarop een object op deze input reageert, hangt af van zijn functie en van de inlet waarop de informatie binnenkomt.



Patch cords

De patch cords verbinden objecten via hun inlets en outlets. Je kunt de specifieke functie van een inlet of outlet nagaan door met de muis bovenop de poort te bewegen. Er verschijnt dan een tooltip met informatie over de werking ervan.

Inlets: hot en cold

Veel Max-objecten hebben meerdere inlets, maar niet elke inlet heeft dezelfde functie.

- Een **hot inlet** activeert de berekening of verwerking en genereert een output. Wanneer je met de muis over de inlet beweegt, kleurt die rood.
- Een **cold inlet** slaat een ontvangen waarde op, maar genereert op zichzelf geen output. Deze inlet kleurt blauw.

Dit principe is essentieel om de dataflow in Max te begrijpen.

Een eenvoudig voorbeeld is `+ 10`. De eerste inlet is hot en ontvangt de waarde waarop de berekening wordt uitgevoerd. De tweede inlet is cold en bepaalt de waarde die erbij wordt opgeteld.

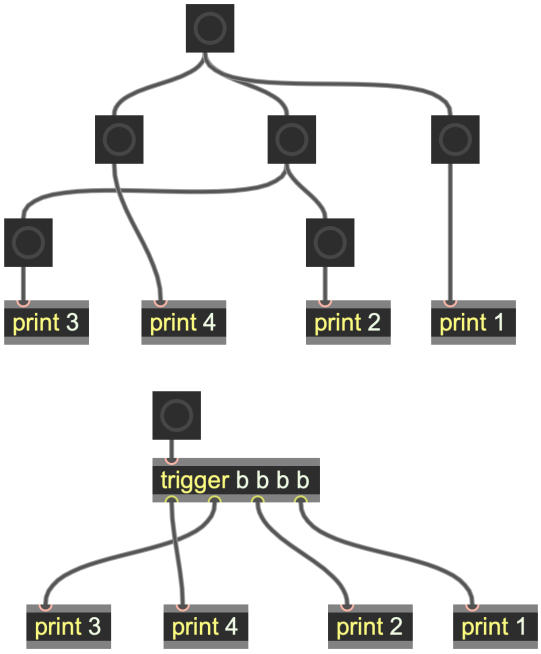


Experiment: verbind twee number boxes met de inlets van `+ 10`. Verander eerst de waarde van de hot inlet en vervolgens die van de cold inlet. Observeer wanneer het object een nieuwe output genereert.

1.4. De volgorde van uitvoering

Niet alleen de verbindingen, maar ook de volgorde waarin objecten worden geactiveerd, speelt een belangrijke rol in Max. Bij de standaard gegevensverwerking hanteert Max een vaste uitvoeringsvolgorde: van rechts naar links en van onder naar boven. Wanneer één object meerdere andere objecten aanstuurt, wordt dus eerst het meest rechtse object geactiveerd. Max verwerkt bovendien elke vertakking volledig voordat het aan de volgende begint (depth-first execution).

Het object `trigger` (afgekort `t`) helpt je om die uitvoervolgorde expliciet te bepalen. Het verdeelt een input over meerdere outlets en activeert die in een vaste volgorde, van rechts naar links. Zo kan je bepalen welke bewerkingen eerst en welke daarna worden uitgevoerd.



1.5. Rekenen en logica

Max beschikt over verschillende objecten voor rekenkundige bewerkingen en logische vergelijkingen. Daarmee kun je waarden aanpassen, vergelijken en voorwaarden opstellen.

Rekenkundige bewerkingen

Met objecten zoals `+`, `-`, `*` en `/` kun je getallen optellen, aftrekken, vermenigvuldigen en delen.

Bijvoorbeeld: `+ 5` telt 5 op bij de ontvangen waarde. Je kunt deze objecten met elkaar combineren om complexere berekeningen op te bouwen.

Logische vergelijkingen

Met vergelijkingsobjecten zoals `>`, `<`, `==` en `!=` kun je nagaan of een waarde aan een bepaalde voorwaarde voldoet.

Een vergelijkingsobject zoals `> 5` geeft `1` als de input groter is dan 5, en `0` wanneer dat niet het geval is.

Door deze output te verbinden met bijvoorbeeld een `toggle`, `gate` of `select`, kun je voorwaarden en beslissingen in je patcher verwerken.

Met `select` (`sel`) kun je specifieke inputwaarden herkennen. Objecten zoals `if` en `expr` maken complexere voorwaarden en berekeningen mogelijk.

Experiment: bouw een eenvoudige patcher die een getal vergelijkt met een grenswaarde. Gebruik het resultaat om een andere actie te activeren.

1.6. Arguments en attributes

Je kunt objecten bij het aanmaken meteen bepaalde eigenschappen of beginwaarden meegeven. Dat doe je met *arguments* en *attributes*.

Arguments

Arguments bepalen onder andere de beginwaarden of standaardwerking van een object. Je plaatst ze rechtstreeks achter de objectnaam.

Bijvoorbeeld:

- `+ 5` : telt standaard 5 op bij de input.
- `metro 500` : genereert standaard een `bang` om de 500 milliseconden.

Attributes

Attributes bepalen eigenschappen van een object en herken je aan het `@`-teken.

Bijvoorbeeld: `metro 500 @active 1` start de metronoom automatisch wanneer de patcher wordt geladen.

Attributes worden bijzonder veel gebruikt in Jitter, maar komen ook voor in Max en MSP.

Veel arguments en attributes kun je tijdens het gebruik aanpassen via messages of UI-objecten zoals number boxes en `attrui`. Zo'n wijziging past echter niet automatisch de tekst in de objectbox aan. Dat kan soms verwarrend zijn: de actuele instelling van een object hoeft dus niet overeen te komen met wat je in de objectbox ziet.

1.7. Veelgebruikte Max-objecten

Onderstaande lijst is geen volledige referentie, maar een overzicht van objecten die je regelmatig zult tegenkomen. Gebruik ze als vertrekpunt om verder te experimenteren en de bijbehorende Help-files te verkennen.

Data en control

Object	Functie
`button`	Genereert een `bang`.
`toggle`	Schakelt tussen 0 en 1.
`metro`	Genereert herhaaldelijk `bang`-messages.
`counter`	Telt in opeenvolgende stappen.
`loadbang`	Activeert een proces bij het laden van de patcher.
`delay`	Stelt een `bang` tijdelijk uit.
`trigger (t)`	Verdeelt input over meerdere outlets en bepaalt de uitvoervolgorde.
`gate / switch`	Regelt de doorgifte van data.
`send / receive`	Wisselt data uit zonder directe patch cords.

Getallen en berekeningen

Object	Functie
`+, -, *, /`	Rekenkundige bewerkingen.
`expr`	Verwerkt wiskundige expressies.
`pack / unpack`	Bundelt of splitst data in lijsten.
`int / float`	Slaat een geheel getal of kommagetal op en geeft het door.
`scale / zmap`	Herschaalt waarden naar een ander bereik.

Logica en routing

Object	Functie
`select (sel)`	Herkent specifieke inputwaarden.
`==, >, <, >=`	Vergelijkingsoperatoren.
`if`	Voorwaardelijke expressies.
`route`	Routeert data op basis van overeenkomende selectors.
`zl`	Verwerkt lijsten, bijvoorbeeld door ze te groeperen of te splitsen.
`prepend / append`	Voegt elementen toe aan het begin of einde van een lijst.

User interface

Object	Functie
`slider` / `dial`	Interactieve bediening van waarden.
`number / flonum`	Numerieke invoer en weergave.
`message`	Verstuurt messages en commando's.
`comment`	Voegt tekst toe aan de patcher.
`umenu`	Pop-upmenu met keuzemogelijkheden.
`attrui`	Toont en wijzigt attributes.
`panel`	Visueel achtergrond- en layoutobject.
`preset`	Slaat instellingen op en roept ze terug.

Interactie

Object	Functie
`mousestate`	Leest muispositie en muisknoppen uit.
`key / keyup`	Detecteert toetsenbordinvoer.
`modifiers`	Detecteert speciale toetsen zoals Shift, Cmd en Alt.

2. MSP — Audio in Max

MSP is het audiogedeelte van Max. Het bevat honderden objecten voor het genereren, verwerken, analyseren en afspelen van geluid. In deze korte introductie verkennen we slechts een klein stukje van de uitgebreide mogelijkheden, net genoeg om audio in onze projecten af te spelen en data uit geluidssignalen te halen en te analyseren.

MSP-objecten herken je aan het tilde-teken (`~`) in hun naam, bijvoorbeeld `cycle~` , `gain~` of `dac~` .

De patch cords van MSP zijn visueel te onderscheiden van gewone Max-verbindingen: ze worden weergegeven als een dikkere, grijs-geel gestreepte lijn. Multichannel-signalen (`mc.`) hebben een eigen blauw-grijze kleurweergave.

Met MSP kun je onder andere software-instrumenten, synthesizers, samplers en audio-effecten bouwen. Deze introductie beperkt zich tot het afspelen van geluidsfiles en werken met de audio-ingang voor het sturen of beïnvloeden van beeld.

2.1. Het principe van een audio signal chain

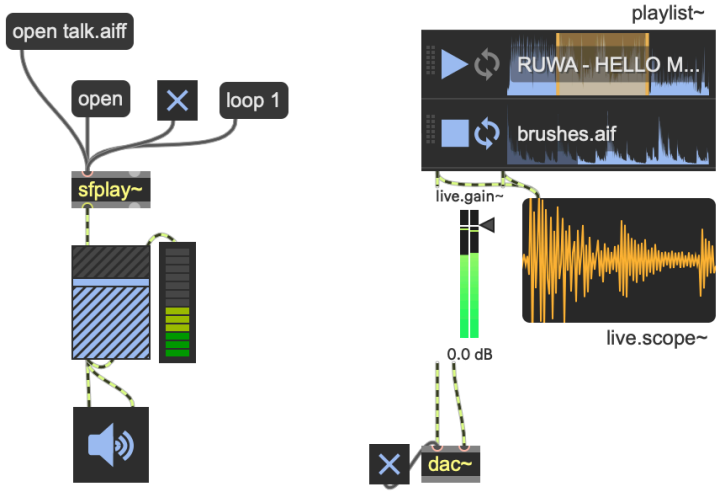
MSP werkt met een continue stroom van digitale audiosamples. Bij een sample rate van 44.100 Hz worden bijvoorbeeld 44.100 samples per seconde verwerkt.

Een audio patch bestaat uit een *signal chain*: een opeenvolging van objecten die het audiosignaal genereren, verwerken en doorgeven.

Twee principes zijn hierbij belangrijk:

1. Een actief audiosignaal: de signal chain moet correct opgebouwd en **aan** zijn om audio te verwerken.
2. Een audio output: om geluid te horen moet het signaal uiteindelijk bij een audio output terechtkomen. Daarvoor gebruik je bijvoorbeeld `dac~` of `ezdac~` . DAC staat voor digital-to-analog converter.

2.2. Soundfiles afspelen



Met een `sfplay~` -object kun je audiobestanden laden en afspelen. Het object ondersteunt o.a. WAV, AIFF, MP3 en M4A bestanden. Via messages zoals `open` , `1` en `0` beheer je respectievelijk het openen, starten en stoppen van de audioweergave.

Met `playlist~` kun je meerdere geluidsbestanden verzamelen, ordenen en afspelen. Elk bestand wordt weergegeven als een afzonderlijke clip, waarin je eventueel een specifiek fragment kunt selecteren. Je kunt clips herschikken door ze te verslepen of naar een ander `playlist~` -object verplaatsen via het bolletje aan de linkerkant.

De audiouitgang van deze objecten verbind je met een volume-knop, bijvoorbeeld `gain~` of `live.gain~` , en vervolgens met een audio-outputobject zoals `ezdac~` of `dac~` . Een `gain~` maakt het mogelijk om het uitgangsniveau aan te passen. Met een bijhorende `meter~` kun je bovendien het signaalniveau visueel opvolgen.

`live.gain~` combineert een volumefader met een level meter. Met `live.scope~` bekijk je de waveform van een signaal.”
`

Audio-instellingen

De globale audio-instellingen vind je in het venster **Audio Status**. Je opent dit via **Options > Audio Status** of door in een vergrendelde patcher te dubbelklikken op een `dac~` of `adc~`.

Hier kun je onder andere de volgende instellingen aanpassen:

- **Audio driver:** bepaalt welke driver gebruikt wordt voor de audioverbinding.
- **Input device:** selecteert de microfoon of andere audio-ingang.
- **Output device:** selecteert de luidsprekers, hoofdtelefoon of andere audio-uitgang.
- **Sample rate:** bepaalt hoeveel samples per seconde worden verwerkt.
- **Vector size en bufferinstellingen:** bepalen hoe audio in blokken wordt verwerkt en hebben invloed op de latency en de stabiliteit van de audioweergave.

Op macOS wordt doorgaans CoreAudio gebruikt. Op Windows hangt de beschikbare driver af van de configuratie, bijvoorbeeld WASAPI, MME of ASIO.

2.3. Audio in

Om live audio in je patch binnen te brengen, gebruik je het object `adc~` of `ezadc~`. ADC staat voor *analog-to-digital converter* en ontvangt een audiosignaal van een geselecteerde audio-ingang, bijvoorbeeld een microfoon of audio-interface.

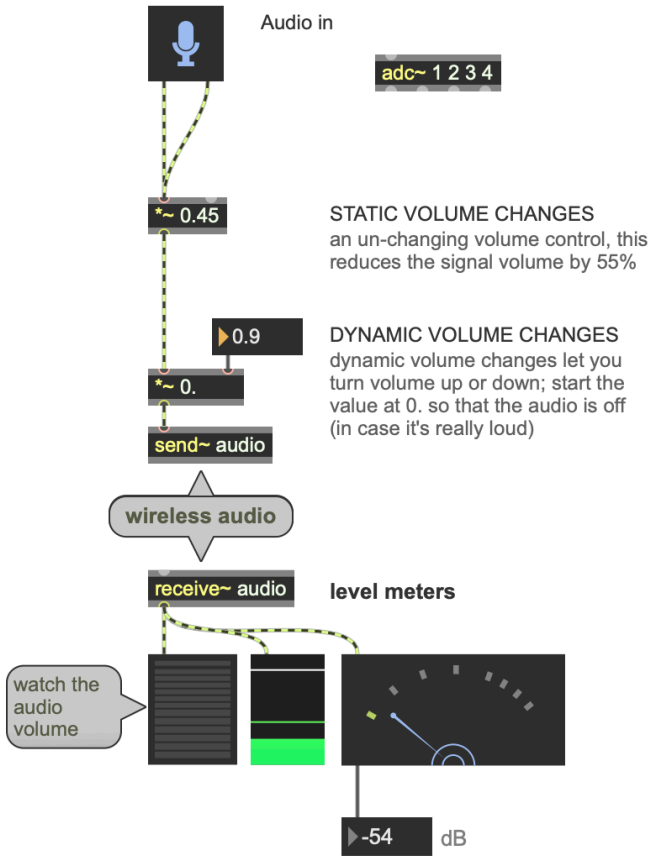
Het binnenkomende signaal kun je rechtstreeks doorsturen naar een audio-uitgang (`dac~`), of eerst bewerken en analyseren. Met volume-regelaars zoals `live.gain~` kun je het ingangsniveau aanpassen.

In de onderstaande patch wordt het microfoonsignaal eerst door `*~ 0.45` gestuurd. Hierdoor wordt het oorspronkelijke volume met 55% verminderd. Vervolgens passeert het signaal een tweede volumeregeling via `*~ 0.`, die gekoppeld is aan een `flonum`-object. Hiermee kun je het volume op elk moment aanpassen.

Daarna wordt het audiosignaal via `send~` doorgestuurd naar een ander deel van de patch. Met `receive~` kun je het daar opnieuw opvangen. Zo kun je audio tussen verschillende onderdelen van een patch verdelen zonder lange patch cords te moeten trekken.

Om het ingangssignaal visueel op te volgen, kun je verschillende soorten meters gebruiken. Naast de eerder besproken `meter~` maken we hier ook kennis met `live.meter~` en `levelmeter~`. Deze objecten geven op verschillende manieren het actuele audioniveau weer.

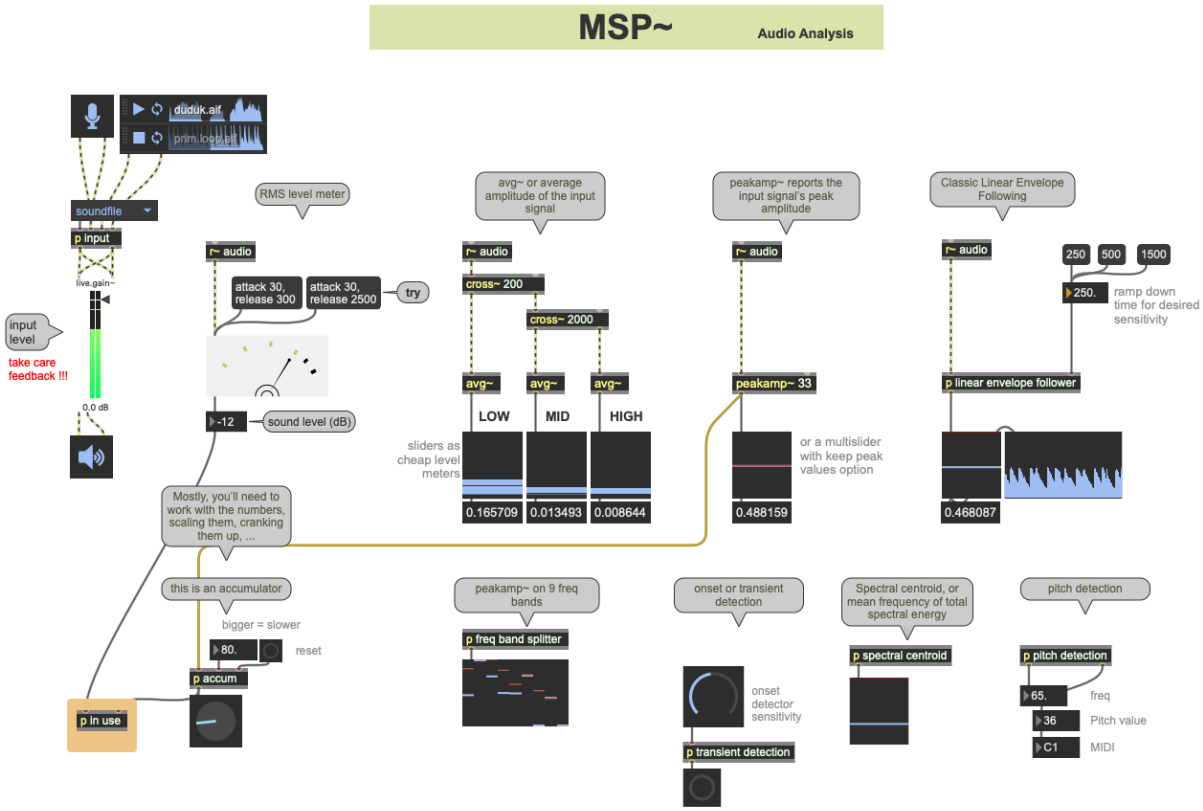
Een visuele meter is niet alleen handig om het volume op te volgen, maar ook om clipping (oversturing) te herkennen. Zeker bij live audio is het belangrijk om het ingangsniveau voldoende laag te houden, zodat het signaal niet vervormt.



2.4. Audio Analysis

Audio analysis maakt het mogelijk om eigenschappen van een geluidssignaal te meten en deze informatie te gebruiken als stuurdata binnen een patch. Zo kun je bijvoorbeeld het volume, de toonhoogte of de frequentie-inhoud van een geluid analyseren en de resultaten koppelen aan andere parameters.

In plaats van audio uitsluitend te beluisteren, kun je het signaal dus ook gebruiken om andere processen aan te sturen. Denk bijvoorbeeld aan een lichtintensiteit die reageert op het volume van muziek, visuals die veranderen op basis van de frequentie-inhoud of animaties die reageren op het ritme van een geluid.



Amplitude en volume meten

Een eerste mogelijkheid is het analyseren van de amplitude (de signaalsterkte) van een geluid. Hiervoor bestaan verschillende objecten die elk op een iets andere manier werken.

- `average~` berekent de gemiddelde amplitude over een bepaalde tijdsperiode. Hiermee kun je de algemene geluidssterkte opvolgen. Een langere meetperiode zorgt voor een stabielere, maar tragere respons.
- `peakamp~` meet de piekamplitude van het signaal binnen een bepaald tijdsvenster. Hierdoor kun je korte, luide pieken detecteren die bij een gemiddelde meting minder duidelijk naar voren komen.

Een veelgebruikte toepassing is het opsplitsen van het audiosignaal in verschillende frequentiegebieden. Door het signaal op te splitsen in verschillende frequentiebanden, kun je de lage, midden- en hoge frequenties onafhankelijk van elkaar analyseren.

In de patch hierboven gebeurt dit met behulp van `cross~` en `avg~` (verkorte schrijfwijze van `average~`). De verschillende frequentiegebieden worden afzonderlijk geanalyseerd en leveren elk een waarde op. Deze waarden kunnen vervolgens gebruikt worden om bijvoorbeeld drie verschillende visuele elementen aan te sturen.

Voor meer geavanceerde toepassingen kun je FFT-technieken (Fast Fourier Transform) gebruiken. Hierbij wordt het audiosignaal omgezet naar een frequentiespectrum, waardoor je de energie in verschillende frequentiegebieden afzonderlijk kunt bekijken en meten.

Envelope follower

Een envelope follower volgt de amplitudeontwikkeling van een geluidssignaal doorheen de tijd. Het resultaat is een geleidelijk veranderende waarde die een indicatie geeft van het waargenomen volume.

Dit is bijzonder nuttig wanneer je een parameter wilt laten reageren op de dynamiek van een geluid zonder dat elke kleine fluctuatie onmiddellijk zichtbaar wordt. Je kunt bijvoorbeeld de helderheid van een visual laten toenemen wanneer het volume stijgt en opnieuw laten afnemen wanneer het geluid zachter wordt.

Dat kan bijvoorbeeld door een `avg~` met `slide~` te verbinden. Hiermee kun je de metingen afvlakken. Door de snelheid van de respons aan te passen, kun je bepalen hoe snel de parameter reageert op veranderingen in het audiosignaal.

Onset detection

Onset detection detecteert plotselinge veranderingen in een geluidssignaal, bijvoorbeeld het begin van een noot, een drumslag of een percussieve attack.

In tegenstelling tot een envelope follower, die de amplitudeontwikkeling blijft volgen, richt onset detection zich op het herkennen van nieuwe gebeurtenissen in het signaal.

Hierbij kun je bijvoorbeeld een visuele animatie telkens opnieuw laten starten wanneer een drumslag wordt gedetecteerd. Ook kun je een trigger gebruiken om een bepaalde actie of gebeurtenis in je patch te activeren.

De gevoeligheid van de detectie speelt hierbij een belangrijke rol. Bij een te hoge gevoeligheid kunnen ook kleine fluctuaties als een onset worden herkend. Bij een te lage gevoeligheid worden subtiele geluiden mogelijk niet gedetecteerd.

Pitch detection

Pitch detection probeert de fundamentele frequentie van een geluidssignaal te bepalen en vertaalt deze naar een toonhoogte.

Dit werkt het best met relatief stabiele, duidelijke tonen, zoals een stem of een melodisch instrument. Bij complexe geluiden, akkoorden of percussie kan het moeilijker zijn om één duidelijke toonhoogte te bepalen.

De gevonden frequentie kan vervolgens worden omgezet naar MIDI-noten of gebruikt worden om andere parameters aan te sturen. Zo kun je bijvoorbeeld de positie, kleur of grootte van een visueel object laten veranderen afhankelijk van de gespeelde toonhoogte.

Spectral centroid

De spectral centroid geeft een indicatie van de helderheid van een geluid. Het wordt berekend als het zwaartepunt van het frequentiespectrum.

Een geluid met relatief veel hoge frequenties heeft doorgaans een hogere spectral centroid dan een geluid dat vooral uit lage frequenties bestaat.

Dit maakt de spectral centroid interessant voor toepassingen waarbij visuals reageren op de klankkleur van een geluid. Zo kun je bijvoorbeeld een visual scherper, helderder of intenser maken naarmate het geluid meer hoge frequenties bevat.

3. Vizzie — Snel aan de slag met Jitter

3.1. De basis

Wat is Vizzie?

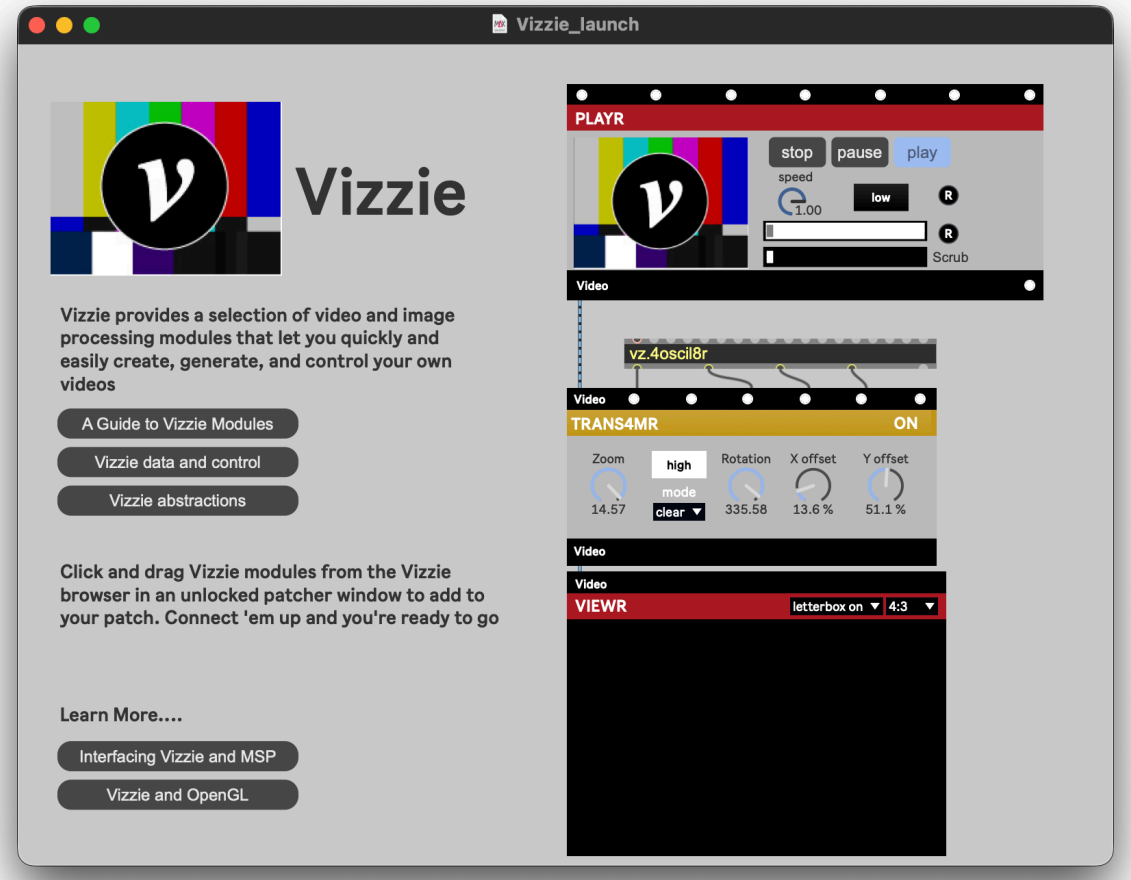
Vizzie is een verzameling kant-en-klare modules voor realtime videoverwerking binnen Max. Door verschillende modules met elkaar te verbinden, bouw je snel een VJ-opstelling, een interactieve video-installatie of een realtime videoeffectketen.

Vizzie is gebaseerd op Jitter. Elke Vizzie-module is een *wrapper* rond een gewone Max-abstraction: een verzameling Jitter-objecten die samen een specifieke functie uitvoeren en voorzien zijn van een toegankelijke User Interface.

Vizzie is bijzonder geschikt voor rapid prototyping, maar ook om de basisprincipes van computational video te leren kennen zonder meteen met de onderliggende Jitter-objecten te moeten werken.

Waar vind je de Vizzie-modules?

Via het menu **Extras > Vizzie_launch** krijg je een overzicht-patcher van de verschillende Vizzie-modules en maak je kennis met de basisfuncties van Vizzie.



De modules zelf vind je via het Module-icoon onderaan de linker toolbar van het patcher-venster. Hiermee open je de Module Browser, waar je de verschillende Vizzie-categorieën en -modules terugvindt.

Via het driehoekje naast Vizzie kun je de modules per categorie bekijken: Control, Effect, Generate, Input, enzovoort.

Een module sleep je vanuit de browser naar een ontgrendelde patcher.

> **Tip**
> Beweeg de muis over de verschillende UI-elementen van een module. De tooltips geven extra informatie over de bijbehorende parameters en functies.

De acht families

De Vizzie Browser verdeelt de modules in acht families of categorieën:

Familie	Functie	Voorbeelden
CONTROL	Beeld routeren, parameters bedienen of manipuleren en externe hardware koppelen.	FADR, 2ROUTR, 2DEALR
EFFECT	Het uiterlijk van het beeld veranderen door kleur of tijdsbewerking.	BRCOSR, DELAYR, SKETCHR
GENERATE	Genereren veranderende beeldpatronen of stuurdata voor visuele composities. Zo voeg je variatie, ritme en toeval toe.	BFGENER8R, 1EASEMAPPR, TWIDDLR, ATTRACTR, WANDR, STEPPR, TOGGLR, CYCLR
INPUT	Videobronnen in de patcher brengen.	PLAYR, AVPLAYR, GRABBR
MIX/COMPOSITE	Verschillende beelden combineren en mengen.	XFADR, 4MIXR, CHROMAKEYR, MODEMIXR
OUTPUT	Het resultaat weergeven of opnemen.	VIEWR, PROJECTR, RECORDR
TRANSFORM	De geometrie, positie en oriëntatie van het beeld veranderen.	CROPR, PIXL8R, ROTATR, ZOOMR
UTILITY	Hulpmodules voor de uitwisseling tussen Vizzie-data en andere Max- of BEAP-stuurwaarden.	AUDIO2VIZZIE

> **Tip**
> Op de ***Vizzie_launch***-patcher heb je een link naar ***A Guide to Vizzie Modules*** waarin een duidelijke beschrijving staat van elke categorie en elke module.

3.2. Patching en verbindingen

Vizzie werkt, net als Max, volgens het dataflow-principe. Het beeld stroomt van een inputmodule via eventuele effecten en transformaties naar een outputmodule.

Elke module beschikt over afzonderlijke inlets en outlets voor video en data. Sommige modules hebben daarnaast ook verbindingen voor audio.

Video, data en audio

- Video: de beeldstroom die door de verschillende modules wordt verwerkt.
- Data: stuurwaarden die parameters van de modules beïnvloeden.
- Audio: enkele modules beschikken over audio-inlets of -outlets. Hiermee kun je bijvoorbeeld audio analyseren en de resultaten omzetten naar stuurwaarden voor video.

De verschillende aansluitingen hebben dus elk een eigen functie. Het is belangrijk om bij het patchen de juiste inlet of outlet te gebruiken.

> **Tip**
> Beweeg de muis over een inlet of outlet om een tooltip te openen. Die geeft aan welk type data verwacht wordt en welke parameter je ermee kunt aansturen.

Videodata: matrix of texture?

In Jitter kan beeld op twee manieren worden opgeslagen, verwerkt en getoond: als *matrix* (CPU) of als *texture* (GPU).

In Vizzie wordt bij voorkeur met textures gewerkt. Textures worden door de GPU (of grafische processor) verwerkt en zijn doorgaans veel efficiënter voor realtime videobewerking. Matrix-data blijft echter ondersteund.

Een belangrijk aandachtspunt is dat de output van je input-module bepaalt welk type beelddata de rest van je patch ontvangt.

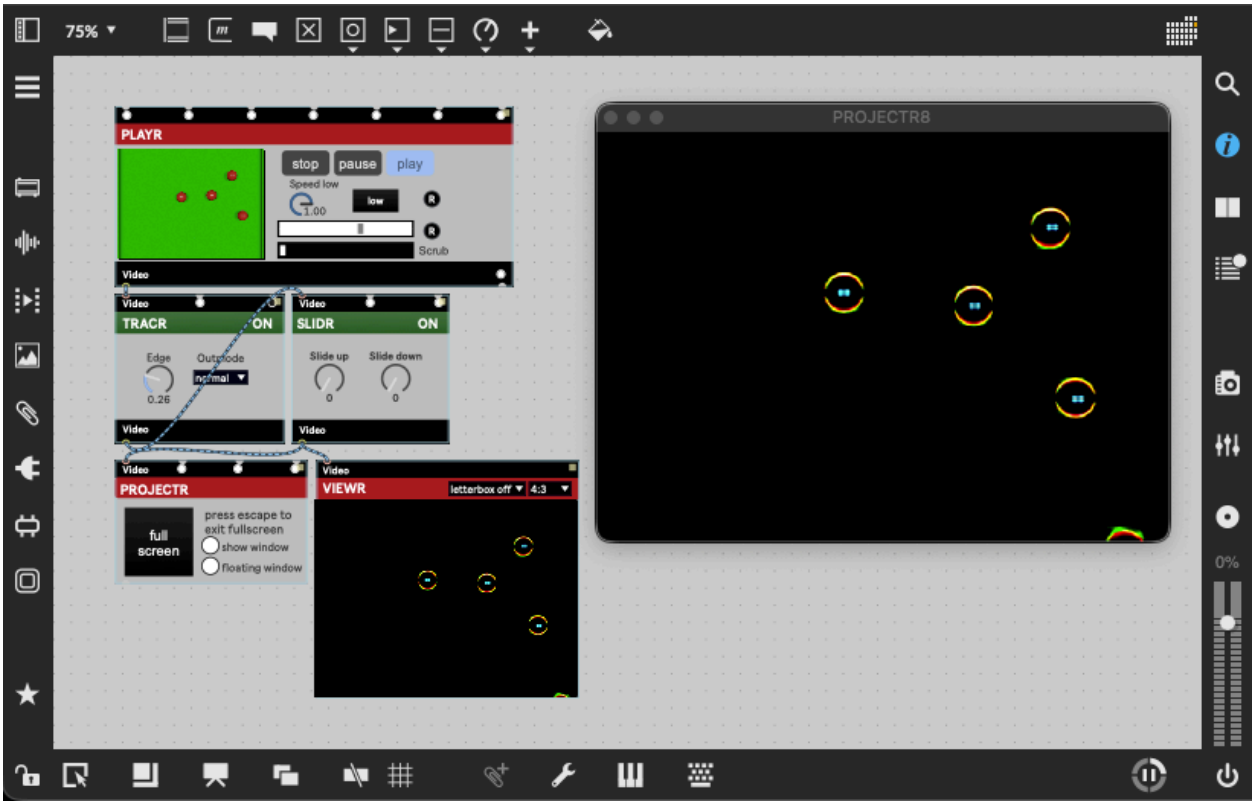
- Een klassieke `jit.movie` zonder texture-output stuurt matrix-data door.
- Met `jit.movie @output_texture 1` kun je texture-data genereren.
- De Vizzie modules `PLAYR` en `GRABBR` genereren standaard textures.

Wanneer je Vizzie-modules combineert met gewone Jitter-objecten, moet je er dus op letten dat de gebruikte beelddata compatibel is. Soms moet je matrix- en texture-data naar het juiste formaat omzetten. Dat kan bijvoorbeeld met de Utility-module `MATRIX2TEXTURE`.

Meer achtergrondinformatie: Vizzie 2: A word about textures and matrices.

3.3. Een eerste Vizzie-chain

Een eenvoudige Vizzie-patcher bestaat uit een input, een of meerdere bewerkingen en een output.



Start een nieuwe patcher, voeg de volgende modules toe en verbind ze met elkaar:

1. **INPUT**: `PLAYR` om een videobestand te laden.
2. **EFFECT of TRANSFORM**: een of meerdere effect- of transform-modules na elkaar naar keuze.

3. **OUTPUT**: `VIEWR` om het resultaat in een venster te bekijken.
4. **OUTPUT (fullscreen)**: `PROJECTR` om het eindbeeld in een apart venster te tonen, met fullscreenfunctie.

Meerdere videobronnen combineren

Je kunt hierop verder bouwen door een tweede videobron toe te voegen, bijvoorbeeld `GRABBR` (de webcam).

Met `XFADR` kun je twee beelden mengen. Het werkt als een crossfader: wanneer de ene bron sterker wordt, neemt de andere af. Past het resultaat je niet, probeer dan andere mix-modules zoals `MODEMIXR`.

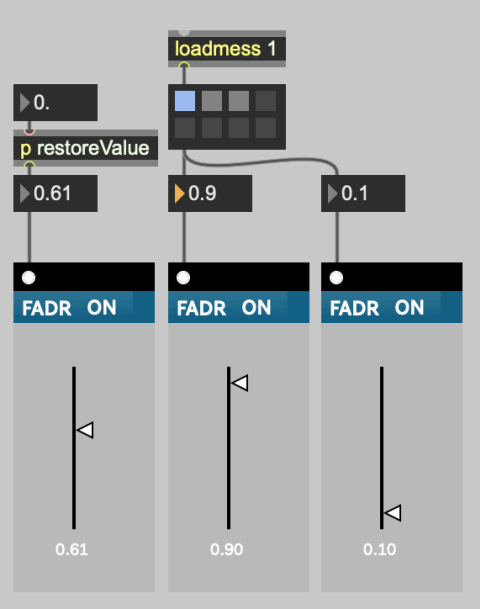
3.4. Interface en stuurdata

De Vizzie-modules beschikken over verschillende instelbare eigenschappen, zogenaamde *parameters*. In je Vizzie-patcher kun je deze aanpassen via bedieningselementen zoals aan-uitknoppen, draaiknoppen en dropdown-menu's. Hiermee bepaal je onder andere hoe een module een beeld genereert of bewerkt.

Parameters bedienen en bewaren

De UI-elementen van Vizzie-modules (knoppen, sliders en draaiknoppen) bewaren enkel een *standaardwaarde* in de patcher. Je laatst gebruikte instellingen worden dus niet automatisch onthouden, waardoor je na het heropenen van een patcher vaak opnieuw op zoek moet naar de juiste regeling. Er zijn enkele manieren om hiermee om te gaan:

- **Snapshots**: sla verschillende combinaties van parameterinstellingen op en roep ze later opnieuw op als presets. Zo bewaar je meerdere configuraties van je compositie en schakel je snel tussen variaties.
- **`preset` -object**: met een eenvoudig Max-object zoals `preset` laad je bij het opstarten van de patcher een basisconfiguratie.



Stuurdata: de waarden 0. tot 1.

Een belangrijk principe van Vizzie is dat alle knoppen van de modules werken met floating-point waarden tussen `0.` en `1.`, aangevuld met de messages `on` en `off`.

Wat die waarden precies betekenen, hangt af van de parameter die je aanstuurt.

Bijvoorbeeld:

- Bij `ROTATR` kan `0.` overeenkomen met geen rotatie en `1.` met een volledige rotatie.
- Bij een crossfader kan `0.` overeenkomen met alleen bron A en `1.` met alleen bron B.

Omdat de modules dezelfde basisconventie gebruiken, kun je verschillende generators eenvoudig met verschillende parameters verbinden.

Waarden herschalen

Niet alle databronnen werken met waarden tussen 0 en 1. Een MIDI-controller stuurt bijvoorbeeld meestal waarden tussen 0 en 127.

Om deze waarden bruikbaar te maken in Vizzie, moet je ze herschalen. In Max kan dat met `zmap` of `scale` of met een eenvoudige berekening, bijvoorbeeld: `/ 127.`. Zo wordt een MIDI-waarde tussen 0 en 127 omgezet naar een floating-point waarde tussen 0 en 1.

3.5. Vizzie-chain 2: modulatie en automation

De mogelijkheden van Vizzie worden pas echt interessant wanneer je modules met elkaar gaat combineren en moduleren. In plaats van parameters handmatig in te stellen, kun je ze automatisch laten veranderen door stuurdata uit andere modules.

Een **GENERATE**-module produceert patronen van stuurdata. Afhankelijk van de gekozen module kunnen dat bijvoorbeeld periodieke bewegingen, willekeurige waarden of sequencerpatronen zijn. Door de output van een generator te verbinden met een parameter van een effectmodule, kun je de werking ervan voortdurend laten veranderen. Zo ontstaan dynamische beeldbewerkingen die automatisch evolueren.

Je kunt ook verschillende generate-modules onderling verbinden. De output van de ene generator beïnvloedt dan de werking van een andere. Ook de output van een effectmodule (ja, video zelf) kan opnieuw als stuurbron worden gebruikt. Zo bouw je een netwerk van onderling verbonden modules waarin beeld en stuurdata elkaar voortdurend beïnvloeden.

Start een nieuwe patcher of bouw verder op de vorige. Breng één of meerdere **Generate**-modules in je patch.

Enkele mogelijkheden naargelang het gewenste gedrag:

- Regelmatig en voorspelbaar: `TOGGLR`, `TWIDDLR`, `OSCIL8R`, `4OSCIL8R`
- Willekeurig en onvoorspelbaar: `RANDOMIZR`, `WANDR`
- Stapsgewijs en ritmisch: `CYCLR`
- Organisch en chaotisch: `ATTRACTR`

Naast de generate objecten vind je onder CONTROL nog enkele modules die de stuurdata kunnen bewerken.

- `JITTR` brengt subtiele storing in de datastroom
- `AVERAGR` en `DATASLIDR` verzachten of vervlakken de input data
- `INVERTR` draait de waarden om, een waarde van `0.1` wordt `0.9`, `0.2` wordt `0.8`, enzovoort.

Door deze bronnen te combineren, kun je complexe en voortdurend veranderende beeldprocessen opbouwen.



Feedback

In het bovenstaande voorbeeld wordt het eindbeeld opnieuw in de keten teruggevoerd en gemengd met het oorspronkelijke beeld. Zo ontstaat een feedbacklus waarbij het beeld zichzelf voortdurend beïnvloedt. Dit kan leiden tot complexe, onverwachte en steeds veranderende visuele patronen.

Feedback is een interessant principe binnen audiovisuele kunst en realtime beeldverwerking. Je kunt niet alleen beeld terugkoppelen, maar ook stuurdata opnieuw als input gebruiken. Zo ontstaan systemen waarin beeld en data elkaar wederzijds beïnvloeden.

Experiment: experimenteer met het terugkoppelen van beeld en stuurdata. Onderzoek hoe kleine veranderingen in de verbindingen of parameters het gedrag van je patch beïnvloeden.

3.6. Vizzie-chain 3: live control

Je kunt Vizzie ook live bedienen. In plaats van parameters met de muis te verslepen, stuur je ze aan met de muispositie, het toetsenbord of een externe controller. Zo wordt je patch een instrument waarmee je tijdens een performance beeld kunt sturen. Het voorbeeld hieronder gebruikt een eenvoudige keten: een `PLAYR`, een kleureffect (`MUTIL8R`), een `XFADR` om het originele en het bewerkte beeld te mengen, en een `VIEWR`.

Het toetsenbord: de klassieke methode

Met het object `key` kun je toetsaanslagen gebruiken als trigger. Vergrendel de patcher, typ een letter en bekijk in de number box welke code `key` teruggeeft. Die code is de ASCII-waarde van de toets: `97` is de a, `122` de z, `101` de e en `114` de r. Dit zijn de vier eerste letters van de bovenste rij op een AZERTY-toetsenbord.

Met `route 97 122 101 114` stuur je elke toets naar een eigen outlet. Elke outlet activeert een `button`, die op zijn beurt een message met een vaste waarde verstuurt (`0.2`, `0.4`, `0.6` en `0.8`). Deze waardes zijn op hun beurt verbonden met het *set the current scrub point*-inlet van `PLAYR`. Zo verander je met een druk op de toets de afspeelpositie van de video.

Het toetsenbord: key mapping

Max heeft ook een ingebouwde manier om toetsen aan UI-objecten te koppelen, zonder dat je toetscodes hoeft op te zoeken:

1. Klik op het toetsenbord-icoon in de onderste toolbar om de keymap-modus te activeren.
2. Klik met de rechtermuisknop op een UI-object, bijvoorbeeld een `toggle`, en kies *Enable for Mapping*.
3. Druk op de toets die je aan het object wilt koppelen.

In het voorbeeld zijn de toetsen **q** en **s** aan twee toggles gekoppeld. Elke toets verstuurt via een `button` een message (`read blading.mov` en `read dozer.mov`) naar de meest rechtse inlet van `PLAYR`. Zo wissel je met een toetsaanslag van videobestand.

> **Note**
> Het bestand dat je met `read` opent, moet in het search path van Max of in dezelfde map als je patch staan.

De muis als controller

Het object `mousestate` leest de positie van de muis uit. In dit voorbeeld zit het verpakt in een subpatcher (`p mousestate`) die de horizontale muispositie omzet naar een waarde tussen `0.` en `1.`, het bereik dat Vizzie verwacht. Verbind de output met een inlet van een module en je stuurt die parameter aan door de muis te bewegen.

> **Tip**
> Weet je niet welke parameter een inlet aanstuurt? Beweeg de muis over de inlet. De tooltip toont wat er verwacht wordt.

Externe controllers: MIDI in Vizzie 2

Vizzie heeft twee modules die specifiek met MIDI-data omgaan:

- `MIDICTRLR` zet de knoppen, faders en andere regelaars van je MIDI-controller om in Vizzie-stuurdata. De module heeft enkel outputs, die je verbindt met de data-inlets van andere modules.
- `MIDIGRABBR` vangt binnenkomende MIDI-berichten op en maakt ze beschikbaar als stuurdata voor je patch.

Zo hoef je geen eigen Max-objecten te bouwen om MIDI uit te lezen.

> **Note**
> Een MIDI-controller stuurt waarden tussen `0` en `127`, terwijl Vizzie `0.` tot `1.` verwacht. Bij de eigen modules hoef je daar niet over na te denken. Gebruik je zelf Max-objecten zoals `ctlin`, dan moet je de waarden herschalen, zoals eerder uitgelegd.

Experiment: bouw een live-set waarin minstens drie parameters door verschillende bronnen worden aangestuurd: de muis, een toets of MIDI-controller (via `MIDICTRLR`) en een LFO of random generator.

3.7. Vizzie-chain 4: audio input

Audio is een krachtige modulatiebron. Je kunt de dynamiek of de frequentie-inhoud van geluid omzetten naar stuurdata, zodat het beeld meebeweegt met de muziek. In deze patch wordt op verschillende manieren een audiosignaal omgezet naar waarden tussen `0.` en `1.`.

Voorbereiding

- **Audio moet actief zijn.** De `ezadc~` - en `ezdac~` -iconen worden blauw wanneer audio aanstaat. Je kunt erop klikken om audio aan of uit te zetten, of het power-icoon rechtsonder in Max gebruiken.
- Gebruik `ezadc~` voor microfooninput of een audioplayer waarin je een audiobestand sleept.
- Met `ezdac~` hoor je het geluid.

> **Warning**
> Gebruik bij experimenten met een microfoon bij voorkeur een hoofdtelefoon om feedback tussen luidspreker en microfoon te vermijden.

Methode 1: AUDIO2VIZZIE

De eenvoudigste manier is de module `AUDIO2VIZZIE`, dat het binnenkomende audiosignaal analyseert en omzet naar Vizzie-stuurdata.

In het voorbeeld is de output verbonden met de crossfade-inlet van een `XFADR`. Twee `PLAYR`-modules worden zo gemengd: hoe luider de audio, hoe meer het beeld verschuift van de ene video naar de andere.

Methode 2: frequentiebanden met AUDIOSPLITTR

`AUDIOSPLITTR` splitst het audiosignaal op in vier frequentiebanden. In het voorbeeld liggen de splitsingsfrequenties op 250, 1000 en 5000 Hz. Elke band levert een eigen stuurwaarde op. Zo kun je bijvoorbeeld de lage frequenties aan één parameter koppelen en de hoge aan een andere.

De amplitudewaarden van de afzonderlijke frequentiebanden zijn afhankelijk van het ingangssignaal en kunnen relatief klein zijn. In het voorbeeld wordt een band daarom met `* 3.` vermenigvuldigd, zodat het volledige bereik van `0.` tot `1.` benut wordt. De waarden van de drie banden sturen de inlets van `PRIMR` aan.

Experiment: laat een bas-band de crossfade aansturen en een hoge band een andere effectparameter. Vergelijk het resultaat met `AUDIO2VIZZIE`.

Methode 3: een envelope follower

Wil je meer controle, dan bouw je zelf een envelope follower in MSP (zie *Audio Analysis*). In het voorbeeld (links) zit die in een subpatcher, `p envelopeFollower`.

De envelope follower stijgt snel bij luide geluiden en zakt daarna langzaam terug, afhankelijk van de ingestelde **decay time**. Die tijd in milliseconden kies je met een number box, die in de patch via messages (`500`, `1000` en `1500`) wordt ingesteld. De output stuurt de crossfade van een `XFADR` aan.

- Een korte decay time geeft een nerveuze, ritmische reactie.
- Een lange decay time geeft een zachte, vloeiende overgang.

3.8. Vizzie-chain 5: audiovisueel instrument

Toevoegen: presentation mode

Bouw een interactieve Vizzie-patcher waarin je minstens twee videobronnen combineert en verschillende parameters live aanstuurt.

- Gebruik minstens één videobron en twee effect- of transformmodules.
- Combineer minstens twee verschillende bronnen van stuurdata (bijvoorbeeld een generator, audio, muis of MIDI).
- Experimenteer met feedback en modulatie.
- Gebruik `PROJECTR` om je compositie op een tweede scherm te tonen.

Onderzoek hoe je met relatief eenvoudige verbindingen een interessant en veranderlijk audiovisueel systeem kunt opbouwen.

4. 🚧 Jitter 🚧

Waar Vizzie gebruikmaakt van kant-en-klare modules, laat Jitter je toe om de onderliggende beeldverwerking rechtstreeks op te bouwen met Max-objecten.

5. Verder leren

- Cycling '74 Documentation
- Cycling '74 Tutorials
- Cycling '74 Forum
- Vizzie Help-files en voorbeeldpatchers.
- Jitter Recipes.