# 26-Contrôle MIDI de la vidéo

## La relation MIDI – vidéo

Lorsque Max a été développé, il était principalement destiné au contrôle interactif d'instruments de musique via MIDI. Avec l 'augmentation de la vitesse des processeur des l'ordinateur, il est devenu pratique d'utiliser Max pour traiter les signaux audio directement avec MSP, et pour traiter de grandes matrices de données comme les images vidéo avec Jitter. La grande puissance de Max est qu'il vous donne accès à toutes les informations numériques dans chacun de ces domaines —MIDI, audio et vidéo — et vous aide à programmer des corrélations intéressantes entre eux. Ce didacticiel se concentre sur l'utilisation des données MIDI entrantes pour contrôler certains aspects de la lecture vidéo dans Jitter.

Les deux principaux avantages de l'utilisation du MIDI pour contrôler la vidéo sont 1) l'avantage d'utiliser une interface physique - un curseur, une molette, etc. - plutôt qu'une souris qui fait glisser un contrôle à l'écran, et b) la possibilité de créer des relations intéressantes entre la musique et la vidéo en temps réel. Il existe de nombreux contrôleurs MIDI qui fournissent des interfaces physiques utiles pour la vidéo: des banques de curseurs multiples sur des boîtes de fader ou des mélangeurs numériques, des molettes, etc. Même la plupart des claviers de synthétiseurs ont des boutons, des curseurs, des molettes et des pédales en plus des touches évidentes de type piano. Pour ce tutoriel, nous nous limiterons aux commandes qui sont couramment disponibles sur la plupart des claviers MIDI.

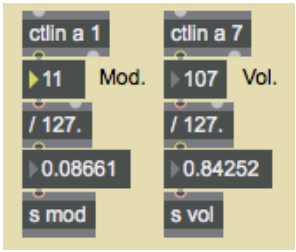
Ce didacticiel part du principe que vous disposez d'un clavier de synthétiseur MIDI à 61 touches avec une molette de modulation et une molette de pitchbend connectés à votre interface MIDI.

Dans ce patch de tutoriel, nous allons lire un film, nous utiliserons les notes du clavier MIDI pour nous déplacer dans le film, et nous utiliserons différents types de messages MIDI pour appliquer des effets à la vidéo et la modifier en temps réel.

## Mappage de données MIDI à utiliser comme paramètres de contrôle vidéo

Les données dans les messages du canal MIDI (notes, pitchbend, aftertouch, etc.) sont comprises entre 0 et 127. Pour la plupart des cas, les attributs des objets Jitter s'attendent à des arguments compris entre 0 et 1, surtout lorsqu'il s'agit de traiter des matrices 2D à 4 plans de données graphiques, comme c'est le cas avec la vidéo. Ainsi, l'une des premières tâches que nous devons effectuer est de mapper les données MIDI dans la plage appropriée pour contrôler les paramètres des objets Jitter. Dans le patch du didacticiel, nous montrons quelques exemples de la façon de procéder.

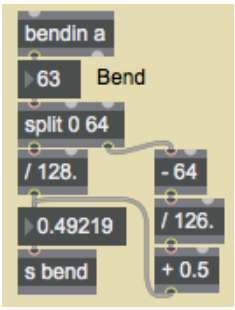
Dans la partie jaune du patch, nous avons divers objets d'entrée MIDI, pour recueillir les données générées par un contrôleur de clavier attaché à *ctlin* **1** pour la molette de modulation, *ctlin* **7** pour la pédale de volume, *bendin* pour la molette de pitchbend et *notein* pour le clavier . (Si nécessaire, double-cliquez sur les objets ci-dessus pour choisir un port MIDI.) Les contrôles les plus directs pour notre utilisation sont les contrôleurs continus comme la molette de modulation et la pédale de volume. Il est très simple de faire correspondre leurs valeurs dans la plage de 0 à 1, en divisant simplement par 127,0.




*Mappez les données de contrôle MIDI dans une plage de 0 à 1 plus utile*

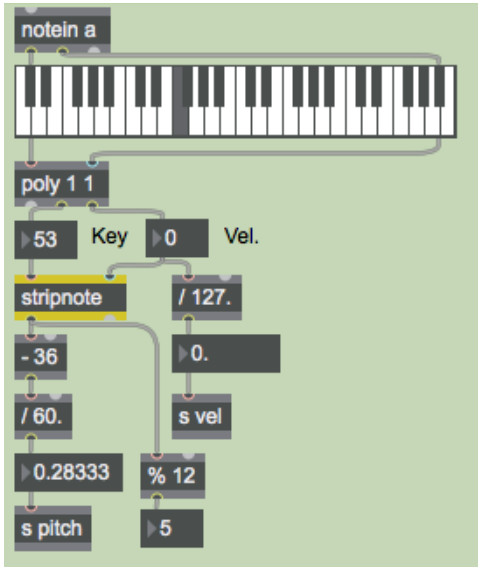
Même si les deux contrôleurs ont une plage de 0 à 127, la position de repos normale de la molette de modulation est à 0 (modulation désactivée), tandis que la position de repos habituelle de la pédale de volume est à une position non nulle telle que 100 ou 127 (volume activé). Ainsi, ils peuvent nous être utiles de manière légèrement différente pour contrôler les effets vidéo.

La molette de pitchbend utilise encore une autre position normale. Sa position de repos est à 64 et elle revient à cette position lorsqu'elle est libérée par l'utilisateur. Nous voulons donc qu'elle nous donne une valeur de 0,5 au repos. Le problème est que 64 n'est pas ***exactement*** à mi-chemin entre 0 et 127; si nous divisons simplement par 127, une valeur de bending de 64 nous donnera un résultat d'environ 0,504. Nous devons donc traiter différemment les valeurs de ***courbure vers le bas*** et de ***courbure vers le haut***, comme le montre l'exemple suivant.




*Il y a 64 valeurs de pitchbend en dessous de la valeur centrale et 63 au dessus du centre.*

Pour les informations de hauteur du clavier, le problème est un peu plus compliqué. Tout d'abord, la plupart des claviers ne disposent pas de touches pour toutes les hauteurs allant de 0 à 127; le clavier normal à 5 octaves envoie les numéros de touches MIDI 36 à 96. Mais plus important encore, nous sommes généralement concernés non seulement à la hauteur "pitch" (hauteur entre 0 et 127), mais aussi à la signification musicale de la classe de hauteur (C, C #, D, etc.). Dans notre patch, nous utilisons les deux manières de considérer la hauteur. Nous mappons la plage des touches 36 à 96 dans la plage de paramètres 0 à 1 et nous dérivons la classe de hauteur avec un objet *%* **12**. (Tous les C seront **0**, tous les C # seront **1**, etc.)




*Utilisation de la valeur de touche de note-on pour dériver la hauteur et la classe de hauteur.*

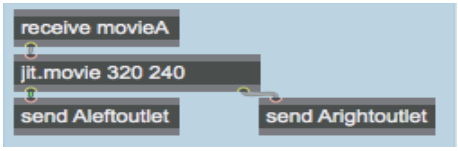
Dans l'exemple ci-dessus, nous utilisons quelques autres objets pratiques pour affiner les données de note entrantes. L'objet *poly* **1 1** ne permet qu'un seul message d'activation de note à la fois; il désactive la note précédente (l'envoie avec une vélocité de **0**) avant de transmettre la nouvelle note. C'est parce que nous voulons suivre un seul numéro de touche à la fois. Si l'utilisateur joue avec la technique du ***legato*** (il joue une note avant de relâcher la précédente) ou joue plusieurs notes d'un accord "simultanément" (c'est-à-dire presque simultanément; rien n'est vraiment simultané en MIDI), il peut être difficile de dire quelle note notre patch suit réellement. L'objet *poly* **1 1** garantit que toutes les notes, à l'exception de la plus récemment jouée, seront désactivées dans notre patch, même si les notes sont toujours maintenues enfoncées sur le clavier MIDI. L'objet *stripnote* supprime les messages de note-off, de sorte que seuls les numéros de touche de note-on seront transmis. Nous ne voulons pas suivre les hauteurs lorsque les notes sont désactivées, nous voulons seulement obtenir le numéro de touche lorsque la note est jouée pour la première fois.

> Remarque: nous avons configuré le patch de sorte que vous n'ayez pas vraiment besoin d'un clavier MIDI pour l'essayer. Vous pouvez jouer des pseudo-notes (silencieuses) en cliquant sur les touches de l'objet *kslider*, et vous pouvez générer d'autres valeurs en glissant sur les boîtes de *nombres* intitulées ***Mod***., ***Vol***., ***Bend***, ***Key*** et ***Vel***. Inutile de dire que la souris sera un peu moins gratifiante qu'un clavier MIDI comme interface physique, mais vous pouvez au moins tester le patch et essayer les choses qui sont expliquées dans ce chapitre.

• Essayez votre clavier MIDI (et vos molettes) pour vérifier que les messages MIDI entrent bien dans Max. Si ce n'est pas le cas, double-cliquez sur les objets d'entrée MIDI pour sélectionner le périphérique d'entrée approprié.

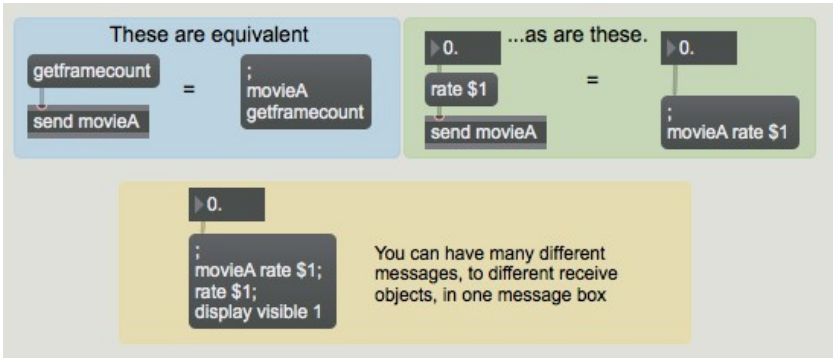
## Utiliser *send* et *receive*

Le fonctionnement de ce patch peut être un peu difficile à suivre car nous faisons un usage abondant des objets *send* et *receive*. Nous faisons cela principalement pour éviter ce qui serait autrement un fouillis ridicule de cordons de connexion. C'est particulièrement approprié ici parce que nous allons envoyer de nombreux messages dans/et hors de l'objet *jit.movie* depuis/vers de nombreux endroits dans le patch. Nous utilisons donc des objets *receive* et *send* pour l'entrée et la sortie de l'objet *jit.movie*, et tous les autres objets du patch peuvent désormais communiquer avec lui à distance.



Nous pouvons communiquer avec *jit.movie* peu importe où il se trouve.

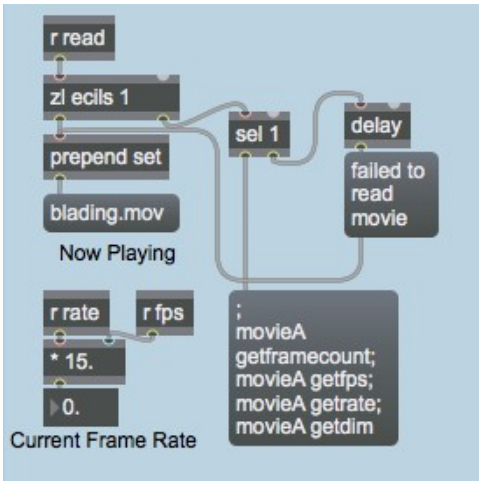
Au cas où vous ne maîtriseriez pas l'utilisation d'un point-virgule **(;)** dans une boîte de *message*, nous prendrons un moment pour souligner que vous pouvez mettre un point-virgule, le nom d'un objet *receive* et un message dans une boîte de *message*, et lorsque cette boîte de message est déclenchée, elle fonctionnera exactement comme si vous aviez envoyé ce message dans un objet *send*. Voir l'exemple suivant.



Un point-virgule dans une boîte de *message* revient à utiliser un objet *send*.

Traçons donc ce qui se passe lorsque nous envoyons un message **read** à *jit.movie*.
• Cliquez sur la boîte de *message* rouge qui dit : movieA read blading.mov ; movieA vol 0.

Cela ouvre le film ***blading.mov***. Lorsque *jit.movie* a terminé l'opération d'ouverture du film, il envoie un message **read** par sa sortie droite. Si le fichier a été ouvert avec succès, le message complet sera: **read blading.mov 1**. (S'il échouait, le dernier argument ne sera pas **1.**) Ce message est envoyé à l'objet *receive* **Arightoutlet** dans la région violette dans le coin gauche du patch. Nous utilisons les objets *route* pour détecter tous les messages que nous attendons de cette sortie et les acheminer aux endroits appropriés ailleurs dans le patch. Les arguments du message **read** sont envoyés à l'objet *r* **read** dans la région verte dans le coin inférieur droit du patch. Avec les objets *zl* **ecils 1** et *sel* **1**, nous vérifions si le dernier argument du message est un **1**. Si c'est le cas, cela signifie que la lecture a réussi, et nous allons donc récupérer les attributs du film.



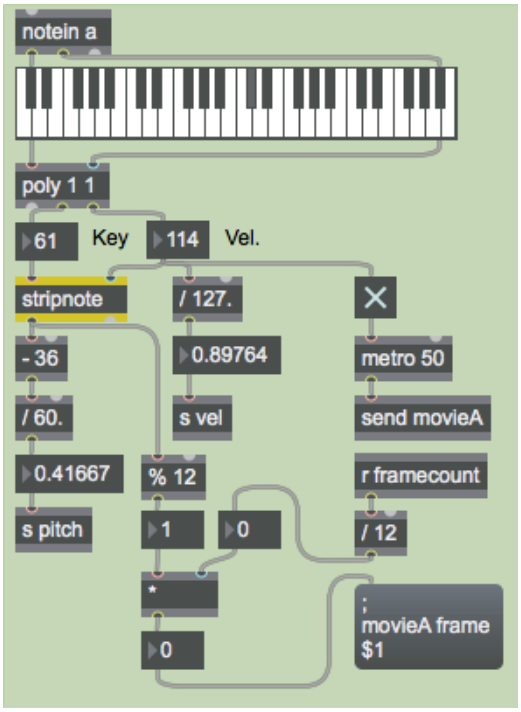
*Si le film a été lu avec succès, vous obtenez ses attributs* ***framecount***, ***fps***, ***rate*** *et* ***dim***.

Le reste du message **read** sera le nom du fichier vidéo, nous le plaçons donc dans une boîte de *message* pour montrer à l'utilisateur le fichier en cours de lecture. Si la lecture a échoué, l'objet *sel* **1** déclenchera le message **failed to read movie**. (L'objet *delay* est là pour s'assurer que le message d'échec est placé dans la boîte de *message* après le nom du fichier.)

## Utilisation de notes MIDI pour déclencher des clips vidéo

Nous avons choisi d'utiliser la classe de hauteur de la note jouée sur le clavier MIDI pour décider où aller dans la vidéo. (Il existe bien entendu de nombreuses manières d'utiliser le MIDI pour naviguer dans un film ou sélectionner différents segments vidéo. Il s'agit simplement de la méthode que nous avons choisie pour ce tutoriel. Dans un chapitre ultérieur, nous montrerons comment basculer d'une vidéo à l'autre.) Ainsi, nous prenons le nombre total d'images dans le film (attribut **framecount** de *jit.movie*) et le divisons par 12. Nous utilisons ensuite la classe de hauteur de chaque note (*key %* **12**) pour sauter à un certain douzième du film.

Le film ***blading.mov*** est une vidéo de 12 secondes composée de douze montages de 1 seconde. Donc, dans ce cas, chaque classe de hauteur différente nous amène à une scène différente de ce court métrage. (Bien sûr, cela est très pratique puisque que nous l'avons prévu ainsi. Mais en utilisant le nombre réel d'images du film, nous avons fait en sorte que notre patch puisse diviser avec succès un film de ***n'importe quelle*** longueur.)




La classe de hauteur (5) de F au-dessus du do médian nous amène à cadrer 75,5/12 dans le film.

La vélocité de la note-on déclenche le *toggle* qui démarre le *metro* qui frappe d'un **bang** l'objet *jit.movie*, et la vélocité de la note-off arrête le *metro*.

• Cliquez sur le *toggle* intitulée ***Show/Hide Display Window*** pour rendre la fenêtre d'affichage visible. Jouez quelques notes sur votre clavier MIDI (ou cliquez sur le *kslider*) pour accéder à différents points du film.

Les matrices de l'objet *jit.movie* sont envoyées à un objet *send* **Aleftoutlet**, et arrivent finalement à la *jit.pwindow* via un objet *receive* **display**. Où vont les messages **jit_matrix** entre *send* **Aleftoutlet** et *receive* **display**? Ils vont en fait dans un sub-patch pour le traitement des effets vidéo. Mais avant

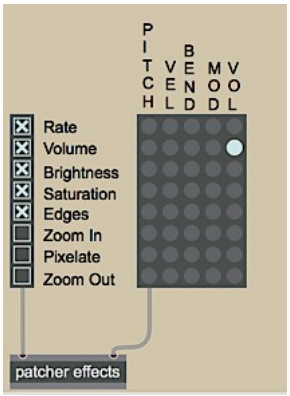
d’examiner ce sub-patch, nous allons voir comment nous avons l'intention de contrôler ces effets.

## Acheminement des informations de contrôle

Plus tôt dans ce chapitre, nous avons vu les différentes manières dont les données MIDI entrantes sont mappées dans la plage 0 à 1 pour pouvoir être utilisées dans le contrôle des attributs de Jitter. Si vous regardez dans la région jaune du patch, vous pouvez voir que les informations de contrôle vont à cinq objets *send*: *s* **pitch**, *s* **vel**, *s* **bend**, *s* **mod** et *s* **vol**. Ce sont cinq sources différentes de contrôle MIDI, et nous allons les utiliser pour contrôler jusqu'à huit effets vidéo différents liés en série. Le lien des effets ressemble à ceci:

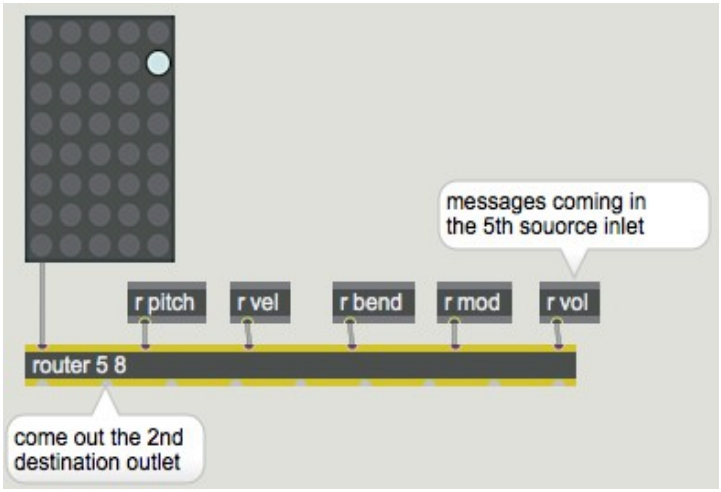
film -> ***contrôle de la fréquence*** -> ***contrôle du volume*** -> ***contrôle de la luminosit***é -> ***contrôle de la saturation*** -> ***détection des bords*** -> ***contrôle du zoom avant*** -> ***pixelisation*** -> ***contrôle du zoom arrière*** -> ***fenêtre d'affichage***.

Pour une polyvalence maximale, nous aimerions pouvoir contrôler n’importe lequel de ces effets avec n'importe laquelle de nos sources MIDI. Pour ce faire, nous utilisons une combinaison d'objets conçus à cet effet: *matrixctrl* et *router*. L'objet *router* prend les messages dans ses différentes entrées et les achemine en interne vers l'une quelconque de ses sorties que vous spécifiez. L'objet *matrixctrl* (conçu pour contrôler les objets MSP *matrix~* et l'objet Max *router*, et non pour contrôler les matrices de Jitter en soi) fournit une interface utilisateur pour spécifier ces routages. Examinez l’objet *matrixctrl* dans la région bleue du patch.




*Acheminer la cinquième entrée (VOL) vers la deuxième sortie (Volume)*

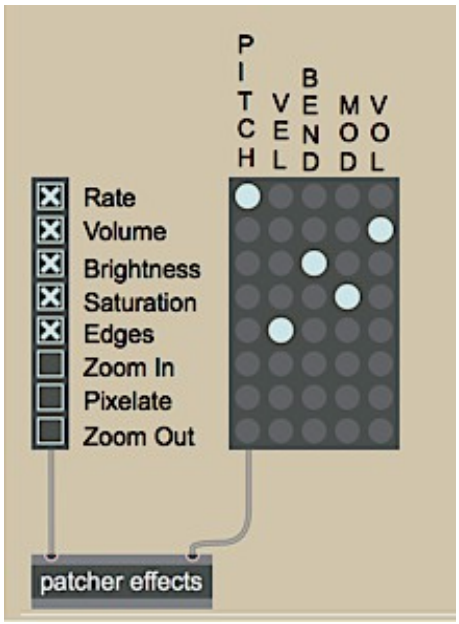
*matrixctrl* affiche les sources d'entrée sur les lignes de grille verticales et les destinations de sortie sur les lignes de grille horizontales. Donc, si nous voulons acheminer les messages de la cinquième entrée source d'un objet *router* vers la deuxième sortie de destination, nous devons cliquer sur le point de la grille où ils se rencontrent. Dans cet exemple, nous demandons d’acheminer les données de ***vol*** pour contrôler l’effet de ***volume***. En cliquant sur ce point de la grille de *matrixctrl*, on envoie un message à un objet *router* qui lui demande d'effectuer cette connexion source–destination en interne. (Cliquer à nouveau efface le point rouge et rompt la connexion dans le *router*.) Dans notre programme, l'objet *router* est à l'intérieur du *sub-patch* **effects**, mais s'ils étaient dans le même patch, leur connexion ressemblerait à l'exemple suivant.




*router* est le "patchbay" pour les messages Max, et *matrixctrl* est son interface utilisateur.

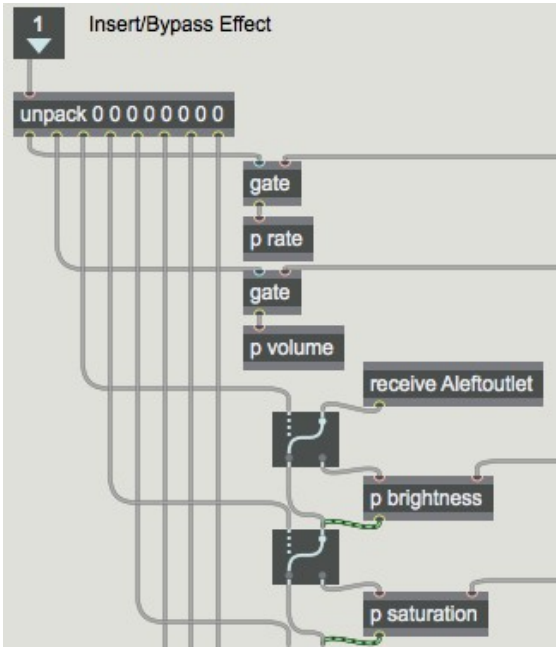
## Contourner (« by passer») des parties du patch

Si nous n'utilisons pas certains des effets vidéo à un moment donné (par exemple, si nous ne souhaitons pas de zoom ou de pixelisation), nous devons contourner ces effets particuliers. Dans le sub-patch ***effects***, nous utiliserons les objets *Ggate* et *gate* pour contourner certains des effets. Pour permettre à l'utilisateur de contrôler facilement les effets à utiliser et ceux à contourner, nous avons configuré des cases à cocher dans le patch principal, en utilisant l'objet *radiogroup*. Lorsque l'utilisateur clique sur l'une des cases à cocher, *radiogroup* envoie le statut d'activé/désactivé de toutes les cases à cocher par sa sortie, et nous pouvons utiliser cette information pour changer le routage des *Ggates* dans le sub-patch.




*Les effets Zoom avant, Pixéliser et Zoom arrière sont complètement ignorés.*

• Double-cliquez sur  l'objet **effects** du patcheur pour voir le contenu du sub-patch ***effects***.



La sortie de *radiogroup* est utilisée pour permuter les objets *gate* et *Ggate* dans le sub-patch.

Dans le sub-patch, l'objet *receive* **Aleftoutlet** reçoit les messages *jit_matrix* du *jit.movie* dans le patch principal. Dans l'exemple ci-dessus, l'objet *Ggate* achemine le message *jit_matrix* autour du sub-patch *p* **brightness** - contournant cet effet - et le prochain objet *Ggate* achemine le message à travers le sub-patch *p* **saturation**. Ainsi, les objets *Ggate* servent de commutateurs ***Insert***/***Bypass*** pour chaque effet, et les cases à cocher dans le *radiogroup* fournissent l'interface utilisateur pour ces commutateurs. A la fin de cette chaîne d'effets, la matrice est finalement transmise à un objet *send* **display**, qui l'envoie la matrice à la fenêtre ***Display***.

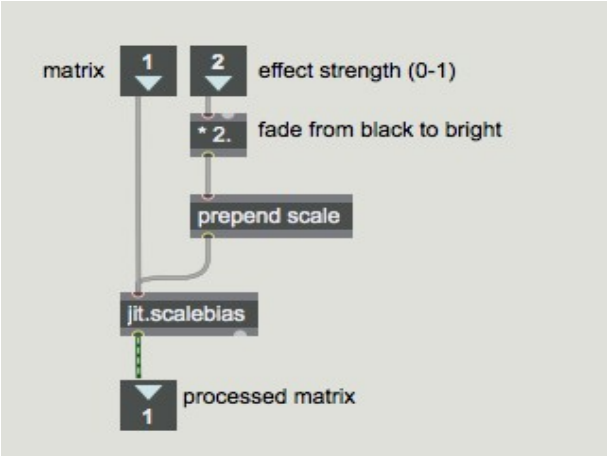
Ainsi, dans le patch principal, nous disposons de deux commandes distinctes permettant à l'utilisateur de configurer l'acheminement des données de contrôle et des effets. Avec l'objet *matrixctrl*, les données MIDI d'une source (ou de plusieurs sources) peuvent être acheminées vers un (ou plusieurs) des effets. Avec les cases à cocher du *radiogroup*, l'utilisateur peut choisir d'insérer des effets ou de contourner entièrement un ou plusieurs effets.

• Fermez la fenêtre du sub-patch [***effects***]. Utilisez les cases à cocher pour sélectionner les effets vidéo que vous souhaitez insérer, et utilisez *matrixctrl* pour affecter des sources MIDI au contrôle de ces effets. Jouez avec différentes combinaisons pour voir quels types de contrôle sont les plus intuitifs (et fonctionnent dans le contexte d'une performance au clavier) pour chaque effet.

## Contrôle des effets vidéo par l'utilisateur

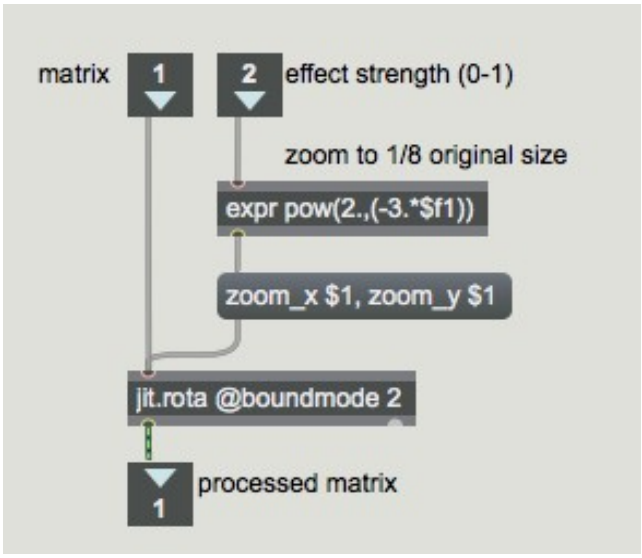
Chaque effet vidéo de ce patch de tutoriel est assez simple, nous ne les décrirons donc pas en détail. Nous allons simplement souligner quelques caractéristiques qui pourraient vous être utiles pour décider comment vous voulez appliquer le contrôle utilisateur en temps réel des effets vidéo .

• Double-cliquez à nouveau sur l'objet **effects** du patcheur pour ouvrir la fenêtre de sub-patch **effects**. Double-cliquez sur l'objet *p* **brightness** pour voir l'un de ces sub-patch d'effets vidéo.




*Les données 0 à 1 sont mises à l'échelle dans la plage 0 à 2 pour contrôler la luminosité.*

Le message *jit_matrix* arrive dans l'entrée gauche et les données de contrôle (dans la plage de 0 à 1) arrivent dans l'entrée droite. Les données de contrôle sont mises à l'échelle dans une plage appropriée, et sont utilisées pour modifier un attribut dans un objet Jitter. La matrice traitée est transmise à l'effet suivant dans la chaîne. Les sub-patchs *p* **saturation** et *p* **zoom** fonctionnent de manière assez similaires. Le sub-patch *p* **zoomout** est également similaire, mais utilise une expression mathématique légèrement plus complexe, illustrée dans l'exemple suivant.



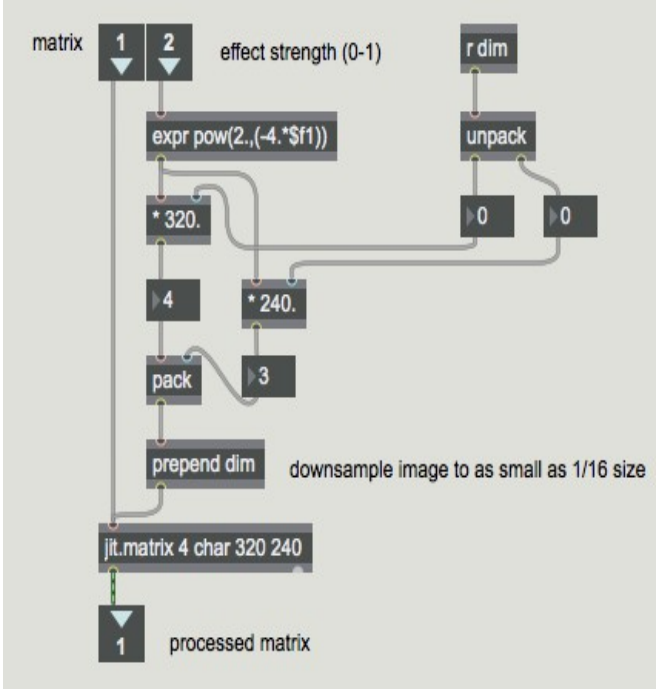

*Les données 0 à 1 sont remappées en tant que courbe exponentielle de 1 à 0,125.*

Dans l'exemple ci-dessus, les données de contrôle entrantes (0 à 1) sont utilisées pour calculer l'exposant d'une puissance de 2. Lorsque les données de contrôle sont à 0, l'expression sera $2^0 = 1$. Lorsque les données de contrôle sont 1, l'expression sera $2^{-3} = 0{,}125$. Ainsi, la valeur de contrôle réelle est inversée pour avoir une signification inverse et pour décrire une courbe exponentielle au lieu d'un changement linéaire.

Dans le sub-patch *p* **edges**, l'objet qui crée réellement l'effet est un objet de détection de bords Sobel appelé *jit.sobel*. Ce que nous contrôlons est le mélange entre l'image d'entrée d'origine et la sortie du détecteur de bords. Nous ne faisons donc que contrôler le paramètre **xfade** d'un objet *jit.xfade* (décrit en détail dans le ***tutoriel 8***).

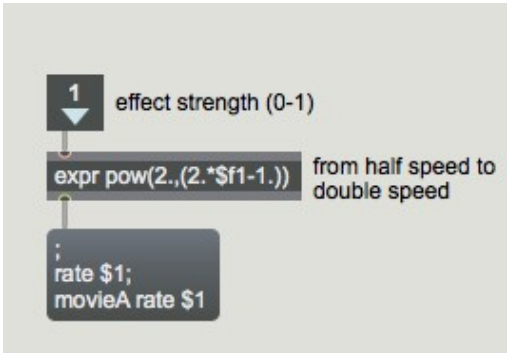
Le sub-patch *p* **pixelate** réduit les dimensions de la matrice de l'image (processus connu sous le nom de ***sous-échantillonnage***), de sorte qu'une partie des données doivent être éliminée et que l'image sera pixélisée lorsqu'elle sera affichée dans la fenêtre **Display** 320x240. (Cette méthode de

pixellisation est détaillée dans le ***didacticiel 14***.) Nous avons obtenu les dimensions de l'image originale en récupérant l'attribut **dim** de *jit.movie* (dès la première lecture du film), nous utilisons donc nos données de contrôle pour mettre à l'échelle ces dimensions par un facteur compris entre 1 et 0,0625, et nous utilisons ces nouvelles dimensions pour définir l'attribut **dim** d'un objet *jit.matrix*, comme le montre l'exemple suivant.




*Le sous-échantillonnage d'une image entraîne sa pixellisation lorsqu'il est sur-échantillonné à ses dimensions d'origine.*

Les sub-patchs *p* **rate** et *p* **volume** sont un peu différents, car nous ne traitons pas réellement une matrice dans ces sub-patchs, nous changeons simplement un attribut de *jit.movie* dans le patch principal. L'exemple suivant montre le contenu du sub-patch *p* **rate**.




Envoyez le résultat à n'importe quel objet *r* **rate**, et utilisez-le également pour définir l'attribut **rate** de *jit.movie*.

## Sommaire

L'interface physique offerte par les contrôleurs MIDI vous donne un bon moyen de contrôler la vidéo en temps réel dans Jitter, et en particulier de faire des corrélations entre la musique et la vidéo. Chaque type de contrôleur - clavier, molette de pitchbend, molette de modulation, pédale de volume, etc. - implique un type différent de mappage de contrôle. Toutes les données des messages des canal MIDI se situent entre 0 et 127, mais la façon dont vous mappez ces données pour contrôler les attributs de Jitter varie en fonction de l'effet que vous essayez de produire. Dans ce

patch, comme point de départ, nous avons mappé toutes les données MIDI pertinentes dans la plage 0 à 1, puis nous avons mis à l'échelle cette plage comme nécessaire pour chaque effet vidéo.

Étant donné que les objets Jitter reçoivent tellement de messages différents, il est souvent nécessaire d'utiliser une boîte de *message* pour créer le message désiré. Si vous retrouvez à diriger de nombreux messages différents vers le(s) même(s) endroit(s) à partir de différentes régions du patch, vous pouvez envisager d'utiliser la fonction d'envoi de *message* à distance de la boîte de message - point-virgule plus le nom d'un objet *receive* – afin de réduire la pagaille de vos patchs.

Si vous avez besoin d'envoyer des messages Max depuis de nombreuses sources différentes vers de nombreuses destinations différentes, et que vous avez besoin de la possibilité de reconfigurer le routage des messages source vers les destinations souhaitées, l'objet *router* fonctionne bien comme une ***baie de patch*** configurable pour les messages Max. L'objet *matrixctrl* fournit une interface utilisateur prête à l'emploi pour configurer le patch source-destination interne d'un *router*. Dans ce patch, nous avons utilisé un *matrixctrl* et un *router* pour permettre à l'utilisateur de diriger l'un des cinq types de données de contrôle MIDI vers l'un des huit effets vidéo différents. Nous avons utilisé un objet *radiogroup* pour créer une banque de cases à cocher qui agissent comme des commutateurs d'***insertion***/***de contournement*** pour les effets vidéo.