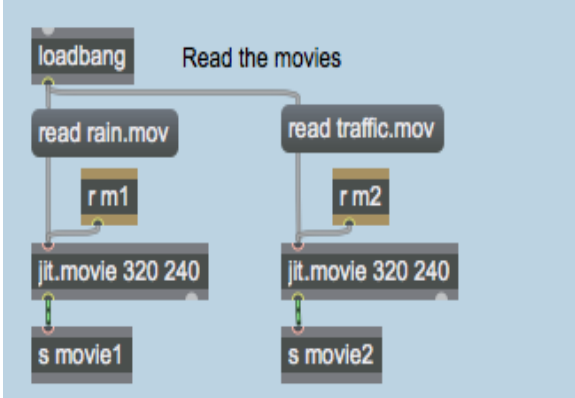
16-Utilisation de matrices Jitter nommées

Dans ce didacticiel, nous allons apprendre à utiliser l'attribut **name** de l'objet *jit.matrix* pour écrire des données de matrice provenant de plusieurs sources dans la même matrice. Nous verrons également comment mettre à l'échelle la taille des matrices lorsqu'elles sont copiées dans une nouvelle matrice et comment utiliser la file d'attente basse priorité de Max pour dé-prioriser les Max en faveur de tâches plus longues.

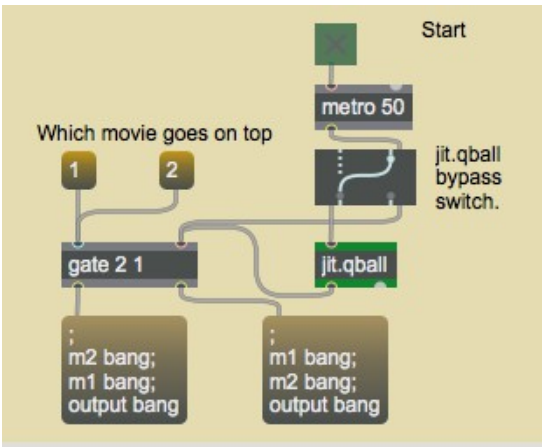
Le patch du didacticiel est divisé en cinq régions colorées. La région du milieu (bleu clair) contient deux objets *jit.movie*. Un objet *loadbang* lit deux films (***rain.mov*** et ***traffic.mov***) dans les objets *jit.movie* à l'ouverture du patch:



Lecture des films

Contrairement aux patchs des didacticiels que nous avons examinés précédemment, les objets *jit.movie* de ce patch utilisent des objets *send* et pour communiquer avec le reste du patch. Les objets *receive* nommés **m1** et **m2** transmettent des messages aux deux objets *jit.movie*. Les matrices de sortie des deux objets sont ensuite envoyées (à l'aide d'objets *send*) aux objets *receive* nommés **movie1** et **movie2** ailleurs dans le patch.

La région jaune en haut du patch contient l'objet *metro* qui pilote les processus Jitter dans le patch:



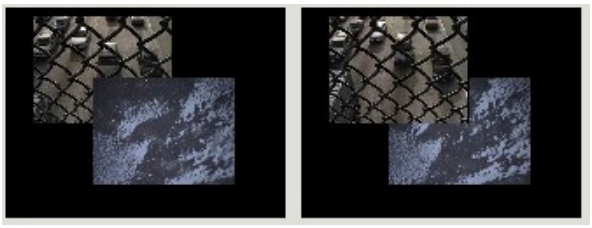
L'objet *metro* conduit l'une des deux boîtes de *message*.

• Cliquez sur le *toggle* pour démarrer le *metro*. Les trois objets *jit.pwindow* du patch vont commencer à afficher une image.

Ordre d'importance

L'objet *metro* passe par un objet *Ggate* et un objet appelé *jit.qball* (sur lequel nous aurons quelque chose à dire plus tard) dans un objet *gate*. Les messages **bang** envoyés par le *metro* sont acheminés par la *gate* vers l'une des deux boîtes de *message*. Notre matrice de sortie finale (dans la fenêtre *jit.pwindow* au bas du patch) changera en fonction de la boîte de *message* qui reçoit un **bang**.

• Cliquez sur les deux boîtes de *message* attachées à l'entrée gauche de *gate* (**1** et **2**). Remarquez comment le *jit.pwindow* en bas change:



La matrice de sortie finale change en fonction de l'ordre des messages du patch.

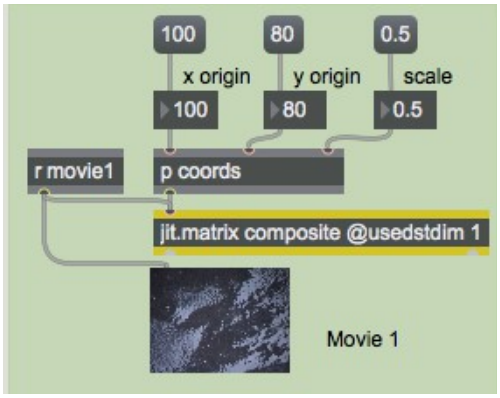
Les deux boîtes de *message* envoient toutes deux des messages **bang** aux trois mêmes objets *receive* nommés (**m1**, **m2** et **output**). La différence entre les deux est l'ordre dans lequel ils envoient les messages. La boîte de *message* de gauche (pilotée par le *metro* lorsque *gate* est à **1**) envoie le premier **bang** à l'objet *jit.movie* (avec le film de pluie). Enfin, l'objet *jit.matrix* au bas du patch reçoit un **bang**, ce qui entraîne l'envoi de notre matrice finale. La boîte de *message* de droite (pilotée par le *metro* lorsque *gate* est réglée sur **2**) inverse l'ordre des deux messages **bang** qui pilotent nos objets *jit.movie* (le *jit.movie* de gauche envoie d'abord une matrice, suivi de l'objet *jit.movie* de droite.).

L'ordre dans lequel ces messages se produisent devient pertinent que lorsque nous examinons ce qui se passe entre les deux objets *jit.movie* et l'objet final *jit.pwindow*.

> Remarque importante: Si vous n'êtes pas sûr de l'ordre dans lequel les choses se passent dans votre patch Max, vous pouvez faire une trace du patch pour voir le mode d'exécution de votre patch. Si vous définissez un point d'arrêt et de surveillance dans un cordon de connexion, vous pouvez parcourir le patch avec la commande **Auto Step** pour voir comment il s'exécute. Cependant, cela ne fonctionnera pas bien avec un *metro* en cours d'exécution, car les **bangs** du *metro* relanceront sans cesse le processus. Il est préférable d'ajouter un *button* à l'emplacement du *metro* et de l'utiliser pour le débogage.

Qu'est-ce qu'il y a dans Name?

Une fois que nos objets *jit.movie* reçoivent leurs messages **bang**, ils envoient une matrice aux objets *send* situés en dessous d'eux, qui à leur tour transmettent leurs matrices aux objets *receive* nommés **movie1** et **movie2**. Les objets *receive* (dans deux régions identiques à droite du patch) sont connectés aux objets *jit.pwindow* ainsi qu'à deux objets nommés *jit.matrix*:



L'objet *jit.matrix* nommé

Les deux objets *jit.matrix* à droite du patch (ainsi que l'objet *jit.matrix* au bas du patch au-dessus de notre fenêtre finale *jit.pwindow*) ont un **nom**. **Name** attaché à ces trois objets est **composite**. Le résultat est que ces trois objets *jit.matrix* partagent les mêmes données de matrice, qui sont contenues dans une matrice de Jitter appelée **composite**.

Une fois que nous savons que nos deux objets *jit.movie* écrivent des données dans la même matrice Jitter (via deux objets *jit.matrix* distincts partageant le même **name**), nous pouvons comprendre pourquoi l'ordre des messages **bang** est important. Si le *jit.movie* de gauche envoie sa matrice en premier, il écrit des données dans la matrice **composite**, suivi par le *jit.movie* de droite, qui écrit des données dans la même matrice. Si les deux matrices écrivent dans des cellules communes (voir ci-dessous), la matrice qui arrive en dernier écrasera toutes les données qui se trouvaient dans ces cellules auparavant.

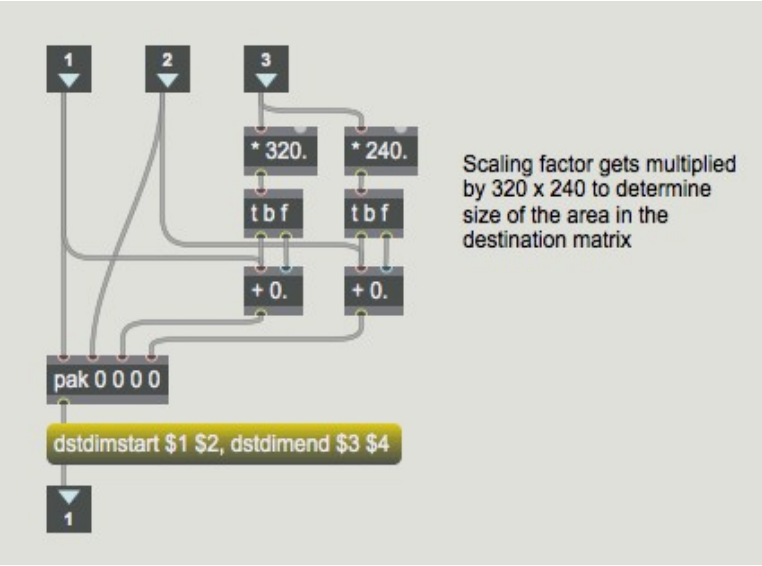
La dimension de destination

Les deux objets *jit.matrix* à droite du patch du didacticiel ont leur attribut **usedstdim** défini à **1**. Cela nous permet de mettre à l'échelle les matrices envoyées par nos objets *jit.movie* afin qu'elles n'écrivent que dans une certaine région de la matrice **composite** Jitter.

• Jouez avec les boîtes de *nombres* étiquetées **x origin**, **y origin** et **scale** connectées aux deux sub-patchs étiquetés *p* **coords**. Remarquez comment vous pouvez déplacer et redimensionner les deux images des objets *jit.movie* dans la matrice composite.


Image dans une image

Les sub-patches *p* **coords** contiennent des patchs d'aide identiques pour formater les attributs **dstdimstart** et **dstdimend** pour les objets *jit.matrix*. Ces attributs spécifient les coordonnées supérieures gauche et inférieures droite, respectivement, à utiliser lors de la copie de données dans notre matrice Jitter **composite**. L'attribut **usedstdim** indique simplement à l'objet *jit.matrix* d'utiliser ces attributs lors de la copie de données. Lorsque **usedstdim** est défini à **0**, la matrice entrante est mise à l'échelle pour remplir l'intégralité de la matrice à laquelle l'objet *jit.matrix* fait référence.



Mise à l'échelle des matrices d'entrée avant le sont écrites dans notre matrice partagée

Les trois nombres que nous envoyons dans les sub-patches sont formatés par les objets Max à l'intérieur pour générer une liste qui décrit les zones supérieure gauche et inférieure droite de la matrice de sortie que nous souhaitons remplir avec notre matrice d'entrée. La boîte de *message* avant la sortie utilise la substitution **$** pour remplacer les arguments pertinents pour les attributs par des nombres de la liste.

La dernière chose qui se passe après que de nos deux matrices aient été écrites dans la matrice **composite** est qu'un **bang** est envoyé à l'objet *receive* nommé **output**:

Le résultat final

La région située au bas du patch du didacticiel contient un troisième objet nommé *jit.matrix*. Le **bang** envoyé par le *metro* passe par un objet *trigger* qui envoie un **bang** à la matrice *jit.matrix* (ce qui provoque la sortie de sa matrice dans la fenêtre *jit.pwindow*) suivi immédiatement d'un message **clear**. Le message **clear** efface (met à zéro) toutes les cellules de la matrice Jitter nommée **composite**. Si nous n'avions pas effacé la matrice, la modification des attributs **dstdimstart** et **dstdimend** de l'un des objets *jit.matrix* pourrait entraîner l'apparition de cellules provenant d'un emplacement de sortie précédent de nos films.

Sauter la file d'attente

L'objet *jit.qball* au sommet du patch fournit un service inestimable dans le cas où Max ne peut pas répondre à nos demandes. L'objet *metro* (qui envoie des messages **bang** toutes les 50 millisecondes)

exécute trois opérations distinctes (écriture des deux matrices des objets *jit.movie* dans notre matrice Jitter nommée, ainsi qu'affichage des données et effacement de la matrice afin que nous puissions recommencer.). L'objet *jit.matrix* écrit des données dans la matrice Jitter interne (dans ce cas, notre matrice nommée **composite**) d'une manière qui lui permet d'être ***remplacée*** par un message ultérieur. Il permet également à d'autres événements Max programmés à une priorité plus élevée de se produire pendant qu'il travaille sur une tâche. Il est ainsi possible d'afficher la matrice (ou d'écrire plus de données) avant que l'opération précédente ne soit terminée, ce qui provoque des scintillements et d'autres résultats inattendus. L'objet *jit.qball* place les messages entrés dans l'objet à la fin de la file d'attente de basse priorité de Max, où ils peuvent également être ***remplacés*** par un autre message. Ainsi, si *jit.qball* reçoit un **bang** de l'objet *metro* avant que toutes les tâches actuelles de Jitter soient terminées, il attendra que tout le reste de la file d'attente à basse priorité soit terminé avant d'envoyer le **bang**. De même, si un autre **bang** survient avant que le premier **bang** ait été envoyé (c'est-à-dire s'il faut plus de 50 millisecondes pour que le reste du patch fasse tout), le premier **bang** sera remplacé (rejeté) en faveur du second. Cela vous permet de définir un taux maximal hypothétique d'événements dans un patch Max sans avoir à vous soucier de l'accumulation d'événements trop rapides pour que les objets du patch puissent suivre.

• Cliquez sur l'objet *Ggate* nommé *jit.qball* **bypass switch** afin que la sortie de l'objet *metro* contourne l'objet *jit.qball*. L'image composite dans la fenêtre *jit.pwindow* en bas commencera à scintiller, indiquant que les messages arrivent dans le désordre.

Normalement, l'envoi d'un **bang** à un objet Jitter remplace un événement déjà en attente dans cet objet (par exemple, un **bang** qui est déjà arrivé mais n'a pas encore été traité par l'objet). Cependant, l'objet *jit.qball* nous donne ce genre de contrôle sur plusieurs chaînes d'objets Jitter, remplaçant automatiquement des événements ("dropframing") pour garantir que les messages arrivent dans le bon ordre.

> Remarque: *jit.qball* a été supplanté par *qmetro* dans la plupart des applications. *Qmetro* est un objet *metro* avec le mécanisme *qball* intégré.

Sommaire

En attribuant un nom unique à plusieurs objets *jit.matrix*, vous pouvez écrire et lire une matrice Jitter commune dans différentes parties de votre patch. Vous pouvez mettre à l'échelle une matrice Jitter en la copiant dans la matrice interne d'un objet *jit.matrix* en utilisant les attributs **dstdimstart** et **dstdimend** et en fixant l'attribut **usedstdim** à **1**. L'objet *jit.qball* permet de ne dé-prioriser les événements Max en les plaçant dans la file d'attente de basse priorité où ils peuvent être remplacés par des événements ultérieurs s'ils ne disposent pas de suffisamment de temps pour s'exécuter.