13-Colle et ciseaux

Dans ce didacticiel, nous allons apprendre à utiliser deux objets simples pour découper et combiner des régions rectangulaires de matrices Jitter bidimensionnelles.

Le patch du didacticiel montre deux objets Jitter qui se complètent parfaitement: *jit.scissors*, qui découpe une matrice en matrices plus petites de taille égale, et *jit.glue*, qui colle plusieurs matrices en une seule. Nous jetterons également un bref coup d'oeil à un objet Max appelé *router*, qui vous permet de router facilement des messages Max de plusieurs sources vers plusieurs destinations.

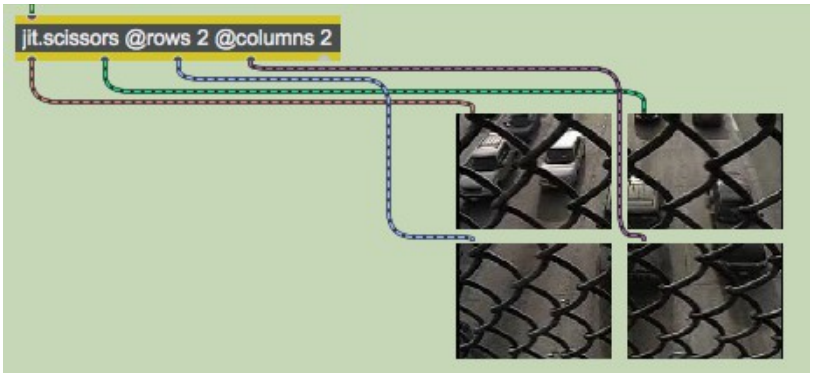
Lire le film

Le coin supérieur gauche du patch est assez simple. L'objet *loadbang* envoie automatiquement le message **read traffic.mov** à l'objet *jit.movie*, qui charge alors notre film contenant des images de circulation.

• Démarrez le *metro* en cliquant sur le *toggle* situé en haut du patch. Vous verrez le trafic apparaître dans la grande *jit.pwindow* au bas du patch. Plus intéressant encore, vous verrez l'image du trafic coupée en quadrants, chacun d'eux apparaissant dans un objet *jit.pwindow* séparé sur le côté droit .

Découper

L'objet *jit.scissors* est chargé de diviser la matrice Jitter contenant les images du trafic en quatre matrices plus petites:



L'objet *jit.scissors*

L'objet *jit.scissors* découpe une matrice Jitter de n'importe quelle taille, type ou nombre de plans en matrices Jitter plus petites, qui sont ensuite envoyées à des sorties indépendantes de l'objet. Les attributs de lignes (**rows**) et de colonnes (**columns**) spécifient combien de matrices plus petites sont créées chaque fois que l'objet reçoit une nouvelle matrice dans son entrée. Dans notre patch de didacticiel, l'objet *jit.scissors* divise l'image en quatre matrices plus petites (2 colonnes et 2 lignes). Ces matrices séparées ressortent des sorties individuelles de l'objet dans ***l'ordre majeur des***

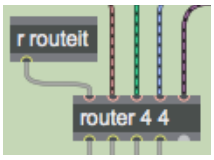
colonnes (c'est-à-dire que l'objet assigne des sorties aux matrices plus petites de gauche à droite, et ensuite de haut en bas).

Deux choses très importantes à savoir sur *jit.scissors*:

1) Le nombre de sorties que *jit.scissors* a est déterminé à la création de l'objet. Par conséquent, les attributs **rows** et **columns ne** créent des sorties que lorsqu'ils sont spécifiés dans la boîte d'objet. Par exemple, taper *jit.scissors* **@rows 10 @columns 2** créera une instance de *jit.scissors* avec 20 sorties matricielles (plus la sortie habituelle de droite pour les requêtes d'attributs), mais créer simplement un objet *jit.scissors* sans arguments vous donnera qu'une seule sortie de matrice. Vous pouvez modifier les attributs des **rows** et des **columns** avec des messages Max à l'objet, mais vous ne pourrez pas ajouter de sorties au-delà de celles initialement créés par l'objet.
2) 2) La taille (**dim**) des matrices produites par *jit.scissors* est égale à la taille des tranches de la matrice, et à la matrice originale entière. Par exemple, les quatre petites matrices de notre patch de tutoriel auront chacune une taille de 160x120 cellules, et non 320x240.

Routage des matrices

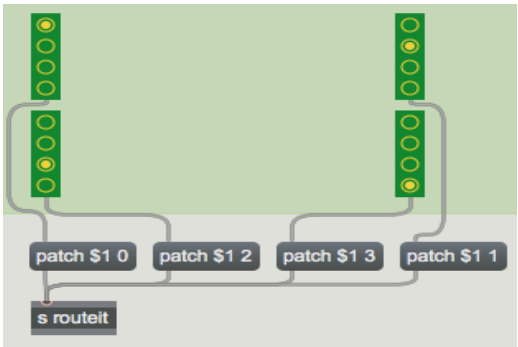
Les quatre matrices plus petites produites par *jit.scissors* dans notre patch sont chacune envoyées à deux endroits différents: aux objets *jit.pwindow* pour que nous puissions voir ce qui se passe, et à un objet Max au milieu du patch appelé *router*. Les cordons de raccordement colorés illustrent où chaque petite matrice est envoyée.



L'objet Max *router*

L'objet *router* est une combinaison des objets Max *gate* et *switch*. Il prend deux arguments (le nombre d'entrées acheminables et le nombre de sorties acheminables) et est contrôlé par les messages envoyés à l'entrée la plus à gauche. La plupart des messages que comprend *router* sont identiques à ceux de l'objet MSP *matrix ~*. Par conséquent, vous pouvez facilement utiliser *router* avec l'objet *matrixctrl*.

Les quatre entrées situées à droite de l'objet *router* prennent leur entrée des quatre sorties de matrice de notre objet *jit.scissors*. Un objet *receive* assigné au symbole **routeit** reçoit des messages de la partie inférieure droite du patch du didacticiel, qui contrôle notre objet *router*. Les quatre sorties les plus à gauche de l'objet *router* sont connectées à un objet *jit.glue*, dont nous parlerons dans un instant.



Contrôler *router*

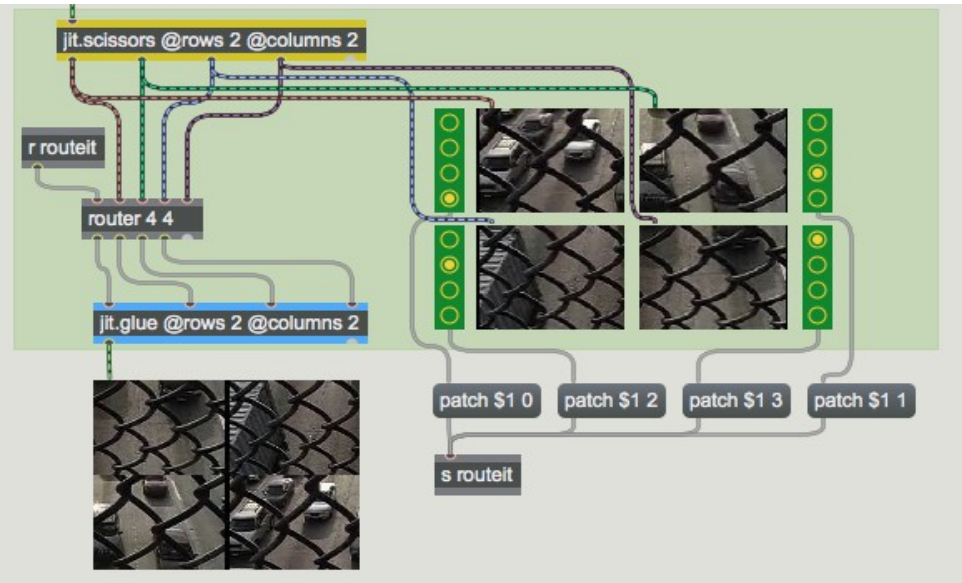
En envoyant le message **patch** suivi d'un numéro d'entrée et d'un numéro de sortie à un objet *router,* on établit une connexion virtuelle entre cette entrée et cette sortie dans l'objet. Tout message arrivant à cette entrée sera instantanément transmis à la sortie correspondante. Si une entrée a été précédemment connectée à cette sortie, un message **patch** mettra fin à cette connexion en faveur de la nouvelle.

Les objets *radiogroup* dans ce patch contrôlent les sorties du *router* auxquelles sont envoyées nos quatre petites matrices Jitter (arrivant aux entrées). Les entrées et les sorties sont numérotées à partir de **0**, ainsi le message **patch 2 1** établit une connexion entre la troisième entrée pouvant être routée et la deuxième sortie de l'objet *router*.

• Cliquez sur certains des contrôles du *radiogroup* et observez l'évolution de l'image de sortie dans la fenêtre inférieure de *jit.p change*. Remarquez comment, avec l'objet *router*, vous pouvez faire en sorte que les matrices découpées dans l'image de trafic apparaissent dans n'importe lequel des quatre quadrants de l'image composite du bas.

La colle qui maintient l'ensemble

L'objet *jit.glue* au bas du patch fait l'inverse de *jit.scissors*. Les attributs de **rows** et de **columns** spécifient les entrées, et non les sorties, et une matrice composite est produite. Elle est composée des matrices entrantes disposées en grille.



Envoi de la même matrice aux quatre entrées de *jit.glue*

> Remarque importante: Comme avec *jit.scissors*, *jit.glue* ne peut créer de nouvelles entrées et sorties que lorsque l'objet est créé, donc les attributs de **rows** et de **columns** présents dans la boîte d'objet détermineront le nombre d'entrées de l'objet. De plus, la taille (**dim**) de la matrice de sortie générée par *jit.glue* sera égale à la taille de toutes les petites matrices réunies (par exemple, nos quatre matrices 160x120 dans ce patch donneront une matrice 320x240).

Une dernière remarque à propos de *jit.glue* est que son comportement par défaut est de ne produire une matrice composite que lorsqu'une nouvelle matrice arrive à son entrée ***la plus à gauche***. Si nous devions déconnecter l'entrée la plus à gauche de notre objet *jit.glue*, nous n'obtiendrons plus de nouvelles matrices en sortie de l'objet. L'attribut **syncinlet** vous permet de faire en sorte que *jit.glue* envoie sa sortie en réponse à une entrée différente. Une valeur de **syncinlet** de **-1** fera en sorte que *jit.glue* envoie de nouvelles matrices composites lorsqu'il reçoit de nouvelles matrices à ***n'importe***

quelle entrée. Bien que cela semble une bonne idée en théorie, cela peut rapidement ralentir le taux de rafraîchissement de vos processus Jitter avec beaucoup de travail redondant.

Sommaire

L'objet *jit.scissors* découpe une matrice en matrices rectangulaires plus petites et de taille égale. L'objet *jit.glue* prend des matrices rectangulaires de taille égale et les recolle en une matrice composite. Les attributs **rows** et **columns** de ces deux objets déterminent leur nombre de sorties ou d'entrées, respectivement, lorsqu'ils sont donnés à la création de l'objet, ainsi que la manière dont la matrice est découpée ou collée. L'objet *router* vous permet de connecter arbitrairement les messages Max de plusieurs entrées à plusieurs prises de manière similaire à l'objet MSP *matrix* ~.