14-Positionnement des matrices

Données de positionnement dans une matrice

Dans ce didacticiel, nous examinons différentes manières de prendre une partie d'une matrice et de la placer à un emplacement différent dans une autre matrice. Il existe différentes raisons pour lesquelles vous pourriez vouloir repositionner l'emplacement des données. Nous nous concentrerons particulièrement sur les effets visuels, mais les techniques que nous montrons ici sont utiles pour toute tâche impliquant le déplacement de données matricielles.

Nous montrerons comment isoler une région d'une matrice, la placer à une position particulière dans une autre matrice, la redimensionner (ce qui peut être utile pour les effets visuels tels que l'étirement, la pixellisation et le flou) et la déplacer de manière dynamique.

jit.window

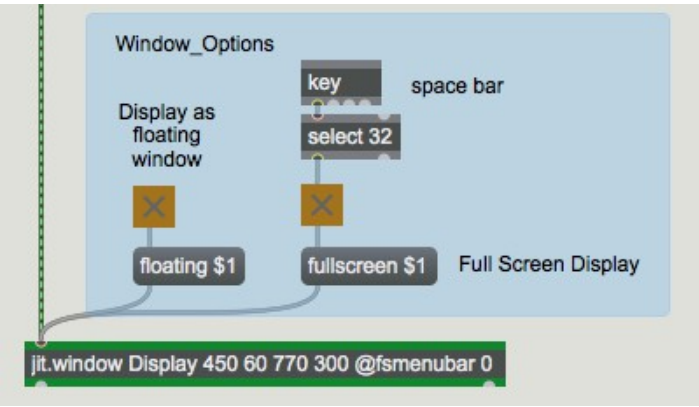
• Ouvrez le sub-patch **Window_options**.

Dans le coin inférieur gauche du patch, vous trouverez un objet *jit.windo*w. Nous avons présenté cet objet dans le ***tutoriel 1***; il crée une fenêtre séparée pour afficher le contenu d'une matrice. Dans la plupart des autres chapitres du didacticiel, nous avons plutôt utilisé l'objet *jit.pwindow*.

jit.window et *jit.pwindow* sont assez similaires - mis à part la différence évidente que l'un ouvre une fenêtre séparée tandis que l'autre utilise une région rectangulaire dans la fenêtre du Patcher – et ils partagent beaucoup des mêmes attributs et messages. Il existe cependant quelques différences, c'est pourquoi nous allons donc utiliser *jit.window* cette fois-ci afin de démontrer quelques-unes de ses caractéristiques uniques.

Vous ne pouvez probablement pas voir la ***fenêtre Display*** qui a été ouverte par l'objet *jit.window* car elle est cachée derrière la fenêtre Patcher. Cependant, si nous le souhaitons, nous pouvons faire de la fenêtre ***Display*** une fenêtre flottante, une fenêtre qui "flotte" au-dessus de toutes les autres fenêtres de Max tout en nous permettant d'interagir avec la fenêtre Patcher de premier plan. Pour ce faire, nous devons activer l'attribut **floating** de *jit.window* avec un message flottant 1. (L'attribut **floating** est **0** par défaut.)

• Ouvrez le sub-patch **Window_options** et cliquez sur le *toggle* intitulé ***Display as floating window*** pour envoyer un message flottant **1** à *jit.window*.



Faire "flotter" la fenêtre devant toutes les autres fenêtres

Notez que les coordonnées d'écran que nous avons saisies dans l'objet *jit.window* pour la zone d'affichage - **450 60 770 300** - spécifient une zone d'affichage de 320 pixels de large par 240 pixels de haut. (Pour une explication sur la façon de spécifier les coordonnées d'écran pour *jit.window*, reportez-vous au ***didacticiel 1*** et/ou à la note plus loin dans ce chapitre.)

D'une jit.matrix à l'autre

Nous allons maintenant charger une image et essayer quelques modifications.

• Cliquez sur la boîte de *message* **importmovie sunset.jpg** pour charger une image dans l'objet *jit.matrix* en haut du patch. Allumez le *metro* intitulé ***Display On / Off*** pour commencer à envoyer des messages **bang** à *jit.matrix*.

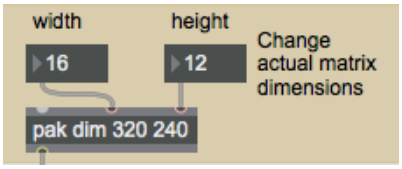
Le **bang** envoie la matrice (via *jit.hue*) à un deuxième objet *jit.matrix* avant d'afficher l'image avec *jit.window*. Dans ce deuxième objet *jit.matrix*, nous pourrons modifier les attributs pour changer la partie de la matrice que nous affichons.

La matrice passe (via *jit.hue*) à un autre *jit.matrix*, puis à *jit.window*.

Nous avons enregistré plusieurs configurations prédéfinies pour les objets de l'interface utilisateur de la fenêtre dans un objet *preset* au milieu du patch.

• Dans l'objet *preset*, cliquez sur preset 1.

Cela change les dimensions de l'objet inférieur *jit.matrix* en 16x12 en lui envoyant un message **dim 16 12**.



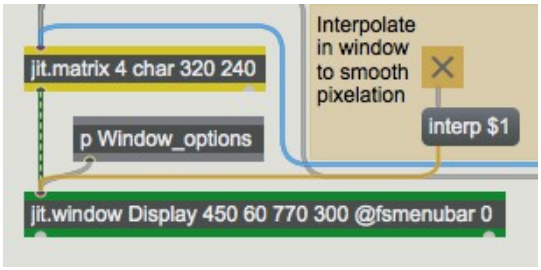
Le message **dim** change les dimensions de la matrice dans *jit.matrix*.

La matrice entrante a des dimensions de 320x240, mais la *jit.matrix* de réception n'a que que des dimensions 16 x 12, elle s'efforce donc d'afficher la totalité de la matrice qu'elle reçoit, mais elle doit nécessairement rejeter une grande partie des informations. Il en résulte une image très pixelisée. (Le terme pixelisation fait référence à l'effet de mosaïque qui résulte d'une résolution d'affichage insuffisante - un nombre insuffisant de pixels - pour représenter fidèlement une image.) Même si l'objet *jit.window* est capable d'afficher l'image en pleine résolution 320x240 (en raison des dimensions de la fenêtre saisies en argument), la matrice qu'il reçoit n'est que de 16x12. Il "étend" la matrice 16x12 en 320x240 pour l'affichage, en dupliquant les pixels si nécessaire.

• Essayez de glisser sur les deux boîtes de *nombres* intitulées ***Change actual matrix dimensions*** pour afficher différents effets de pixellisation.

Interpolation

• Désactivez maintenant les boîtes de *nombres* 16 et 12, puis cliquez sur le *toggle* intitulé ***Interpolate in window to smooth pixellisation*** pour envoyer un message **interp 1** à *jit.window*.

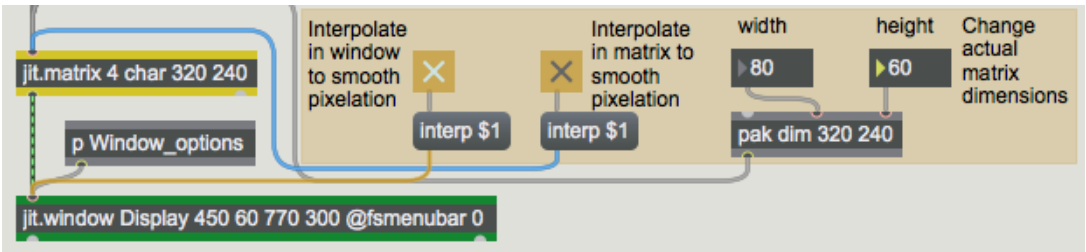


Activer l'interpolation dans *jit.window*.

Maintenant, l'objet *jit.window* - au lieu de simplement dupliquer les pixels de la matrice 16x12 pour former un groupe de blocs de 20x20 pixels - interpole entre les valeurs de la matrice entrante à mesure qu'il l'agrandit à 320x240. Autrement dit, lorsqu'il agrandit l'image, il crée une gradation douce des couleurs entre chaque valeur de cellule et ses valeurs voisines dans la matrice entrante, de sorte que toutes les transitions de cellule à cellule de la matrice 320x240 affichée sont aussi progressives que possible. L'interpolation provoque un flou extrême car la différence de taille entre la matrice entrante et l'affichage est très importante.

• Cliquez à nouveau sur le *toggle* pour désactiver l'interpolation. Cela envoie un message **interp 0** à *jit.window*, en définissant son attribut **interp** à **0** (désactivé). Entrez de nouvelles dimensions de la matrice dans les boîtes de *nombres* intitulées ***Change actual matrix dimensions*** afin que l'image ne soit pas aussi pixellisée: disons **8 0** e t **60**. (Vous devriez voir que les blocs pixellisés ne sont désormais que des 4x4.) Cliquez sur le *toggle* pour réactiver l'interpolation. Notez que dans ce cas, le flou n'est pas aussi extrême car l'interpolation ne porte que sur 4 pixels. Cliquez à nouveau sur le *toggle* pour désactiver l'interpolation.

• Cliquez maintenant sur le *toggle* intitulé ***Interpolate in matrix to smooth pixelation*** pour activer l'interpolation dans *jit.matrix* (pas *jit.window*).



L'interpolation n'en fait pas beaucoup quand vous réduisez la taille de la matrice.

Notez que cela n'a pas beaucoup d'effet. C'est parce que *jit.matrix* n'a toujours qu'une matrice 80x60 à envoyer. L'interpolation dans ce cas (lorsque nous réduisons la taille de la matrice plutôt que de l'agrandir) est plutôt inefficace.

• Cliquez à nouveau sur ce *toggle* pour désactiver l'interpolation dans *jit.matrix*.

Isoler une partie de la matrice

Nous allons maintenant examiner des moyens de se concentrer sur une partie particulière d'une matrice.

• Dans l'objet *preset*, cliquez sur le preset 2. Vous ne voyez maintenant qu'une petite partie de l'image.

Ce preset rétablit les dimensions de notre objet *jit.matrix* à 320x240. Mais nous pouvons toujours isoler une partie particulière de la matrice, sans modifier les dimensions réelles de la matrice complète, en utilisant attributs différents : **srcdimstart**, **srcdimend** et **usesrcdim**. Remarquez que nous avons envoyé trois nouveaux messages à *jit.matrix* pour définir ces trois attributs: **dimstart 40 150**, **dimend 119 209** et **usesrcdim 1**. Ces messages nous permettent de spécifier un sous-ensemble de la matrice complète entrant dans l'entrée, et d'envoyer ces valeurs comme une matrice de taille complète (dans ce cas, 320x240). Ce sous-ensemble plus petit de la matrice entrante est "étendu" (les cellules sont dupliquées si nécessaire) dans *jit.matrix* lui-même, pour remplir la taille de la matrice sortante. Les attributs **srcdimstart** et **srcdimend** sont ignorés. Dans les messages de définition des attributs **srcdimstart** et **srcdimend**, les mots **srcdimstart** et **srcdimend** sont suivis des indices de cellule décrivant les points de départ et d'arrivée dans chaque dimension. Avec nos messages **dimstart 40, 150** et **dimend 119 209**, nous avons dit à *jit.matrix* d'utiliser une région spécifique de 80x60 de la cellule 40 à 119 (inclus) dans la dimension horizontale et des cellules 150 à 209 dans la dimension verticale.

> Remarque: *Dans ce chapitre, nous avons abordé trois façons différentes de spécifier des régions rectangulaires! Il est important de préciser ce que nous spécifions dans chaque cas*.
>
> . Dans *jit.window*, nous avons saisi les coordonnées de l'écran pour la zone d'affichage de la fenêtre. Dans le système d'exploitation de l'ordinateur, les coordonnées de l'écran sont spécifiées en termes de point dans ***le coin supérieur gauche d'un pixel***. Le coin supérieur gauche de l'écran entier est 0,0; le point situé à deux pixels à droite de celle-ci (le coin supérieur gauche du troisième pixel à partir de la gauche) est 2,0; et le point situé 5 pixels plus bas (le coin supérieur gauche du sixième pixel plus bas) est 2,5. Pour décrire une zone rectangulaire de l'écran, nous saisissons des arguments pour les limites gauche, supérieure, droite et inférieure des coordonnées du rectangle..
>
> . Dans l'attribut dim de *jit.matrix*, nous avons fourni des tailles des ***dimensions*** de la matrice de l'objet: ***le nombre de cellules*** dans chaque dimension.
>
> . Dans les attributs **srcdimstart** et **srcdimend**, nous indiquons des ***index*** (inclusifs) ***des cellules*** dans la matrice. N'oubliez pas que les cellules reçoivent des numéros d'index qui vont de 0 à 1 de moins que le nombre de cellules dans cette dimension. Ainsi, pour une matrice 320x240, les indices des cellules de la première dimension vont de 0 à 319 et les indices des cellules de la deuxième dimension vont de 0 à 239. Pour définir les dimensions source de *jit.matrix*, nous devons spécifier la plage de cellules à partir de laquelle nous voulons commencer, en utilisant **srcdimstart** suivi d'un index de cellule de départ pour chacune des dimensions de la matrice, et en utilisant **srcdimend** suivi des indices de cellule pour la fin de la plage dans chaque dimension.
>
> *Ces différentes manières de décrire les régions peuvent prêter à confusion, mais si vous réfléchissez bien à ce que vous spécifiez exactement, vous serez en mesure de déduire la la manière appropriée de décrire ce que vous voulez*.

Nous utilisons seulement une plage de 80 x 60 pixels de la matrice entrante comme source, mais la matrice de destination est de 320 x 240. Encore une fois, cette expansion d'une petite matrice dans une plus grande crée un effet de pixellisation. Cette fois, cependant, l'extension se produit à l'intérieur de *jit.matrix* (c'est-à-dire entre sa région "source" et sa taille «destination»), plutôt qu'entre *jit.matrix* et *jit.window* (comme nous l'avions fait précédemment lorsque nous avons réduit les dimensions réelles de le *jit.matrix*). Par conséquent, si nous voulons lisser la pixellisation en interpolant, nous devons le faire dans *jit.matrix*. Il n'y a aucun intérêt à activer l'interpolation dans *jit.window*, puisqu'elle reçoit déjà une matrice 320x240 de *jit.matrix*.

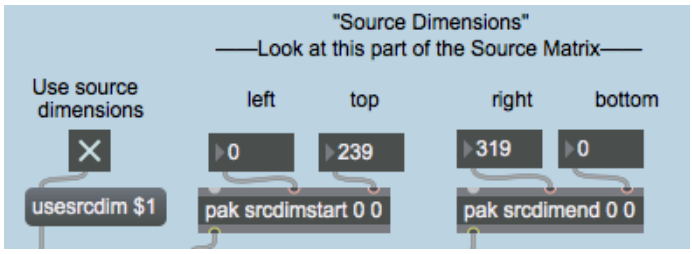
• Si vous voulez le vérifier, activez le *toggle* ***interpolate in window...*** pour envoyer un message **interp 1** dans *jit.window*. Cela n'a absolument aucun effet, car nous essayons d'interpoler une matrice 320x240 dans une zone d'affichage 320x240, de sorte qu'aucun changement ne se produit. Désactivez ce même *toggle* pour remettre l'attribut **interp** de *jit.window* à **0**. Utilisez maintenant l'autre *toggle* pour envoyer un message **interp 1** dans *jit.matrix*. Cette fois, nous obtenons l'effet de lissage que nous désirons.

• Essayez de saisir de nouvelles valeurs dans les boîtes de *nombres* pour modifier les arguments des attributs **srcdimstart** et **srcdimend**. Cela vous permet d'isoler une région particulière de l'image comme zone "source". Bien entendu, les dimensions que vous choisissez pour votre zone source détermineront la distorsion que subit l'image lorsqu'elle est agrandie pour remplir une matrice de sortie 320x240.

Retourner l'image

Vous pouvez supposer que les arguments de l'attribut **srcdimend** (les indices de cellule de fin de la région source) doivent être supérieurs aux numéros d'index de l'attribut **srcdimstart**. Mais ce n'est pas nécessairement le cas.

• Dans l'objet *preset*, cliquez sur le preset 3. L'image est maintenant retournée verticalement.



Le haut et le bas ont été retournés dans la deuxième dimension.

Cet exemple montre que si vous spécifiez un index de cellule de fin dans la dimension verticale qui est inférieur à l'index de départ, *jit.matrix* associera néanmoins ces index aux points de départ et de fin dans la dimension verticale de la matrice de destination, inversant ainsi l'orientation haut-bas des valeurs. (Cette remarque suppose que vous n'avez pas effectué le même type d'inversion de l'orientation de la matrice de destination!)

Vous pouvez faire le même type de retournement dans la (première) dimension horizontale pour retourner l'image horizontalement. Si vous inversez la région source dans les deux dimensions, vous obtenez le même effet visuel que si vous aviez fait pivoter l'image à 180 °.

• Dans l'objet *preset*, cliquez sur le préréglage 4.

Dans cet exemple, nous avons inversé la région source dans les deux dimensions, réduit la taille de la zone source à 160 x 120 et lissé la pixellisation en activant l'attribut **interp**.

Redimensionner la matrice de sortie

Tout comme nous avons spécifié la région source de la matrice, nous pouvons également spécifier une destination pour cette source. Cela ne modifie toujours pas la taille de la matrice de sortie; elle sera toujours de 320x240, comme déterminé par l'attribut **dim**. Toutefois, cela change la région dans laquelle la région source spécifiée sera placée. La région source de la matrice d'entrée sera placée dans la région de destination de la matrice de sortie (avec expansion / contraction si nécessaire). Les cellules de la matrice de sortie qui se trouvent en dehors de la région de destination resteront inchangées.

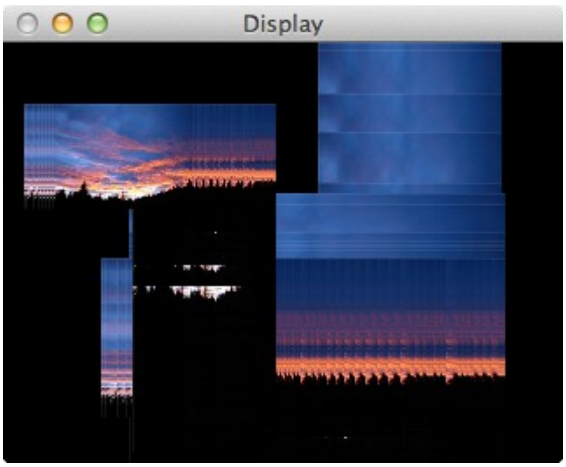
• Dans l'objet *preset*, cliquez sur le preset 5. La matrice d'entrée a été comprimée dans un rectangle de 80x60 au centre de la matrice de sortie.

La première chose à remarquer est que l'attribut **usesrcdim** a été désactivé, de sorte que nous utilisons à nouveau la matrice d'entrée entière comme source. (Les attributs **srcdimstart** et **srcdimend** sont maintenant ignorés.) L'attribut **usedstdim** étant activé, l'entrée sera placée dans la partie de la matrice de sortie spécifiée. Les attributs **dstdimstart** et **dstdimend** ont été définis pour spécifier les cellules du centre de la matrice comme destination: **dstdimstart 120 90** et **dstdimend 199 149**. Nous avons désactivé l'attribut **interp** car nous contractons l'image au lieu de l'agrandir.

Notez également que nous avons activé le *toggle* étiqueté ***Erase previous image***. Cela envoie le chiffre **1** dans l'objet *if* **$ 2** puis **clear**. La partie *if* de l'instruction est maintenant vraie, donc chaque fois que l'objet reçoit un message dans son entrée gauche, il envoie le message **clear**. Cela efface le contenu de l'objet *jit.matrix* immédiatement après l'affichage de l'image pour préparer *jit.matrix* à la prochaine matrice qu'il recevra. Cela garantit que les valeurs dans toutes les cellules en dehors de la région de destination seront **0**, de sorte que la région inutilisée de la matrice de sortie sera affichée en noir.

Modifiez certaines des valeurs dans les boîtes de *nombres* qui fournissent les dimensions de destination, pour déplacer (et redimensionner) l'image dans la fenêtre d'affichage.

Désactivez maintenant le *toggle* intitulé ***Erase previous image*** pour supprimer les messages **clear**. Modifiez à nouveau les arguments de **dstdimstart** et **dstdimend**, et remarquez ce qui est différent cette fois-ci. Les régions de destination précédentes sont toujours dessinées dans la fenêtre d'affichage parce que ces cellules de la matrice n'ont pas été effacées, et elles restent inchangées si elles se trouvent en dehors de la nouvelle région de destination. Cela donne l'effet de laisser des «traces» de l'image précédente derrière soi. Nous pouvons potentiellement utiliser ces artefacts pour leur effet visuel particulier.



Si la matrice n'est pas effacée, les anciennes zones de destination seront laissées si elles se trouvent en dehors de la nouvelle région de destination. Pour des changements continus, cela laisse une trace d'images du passé.

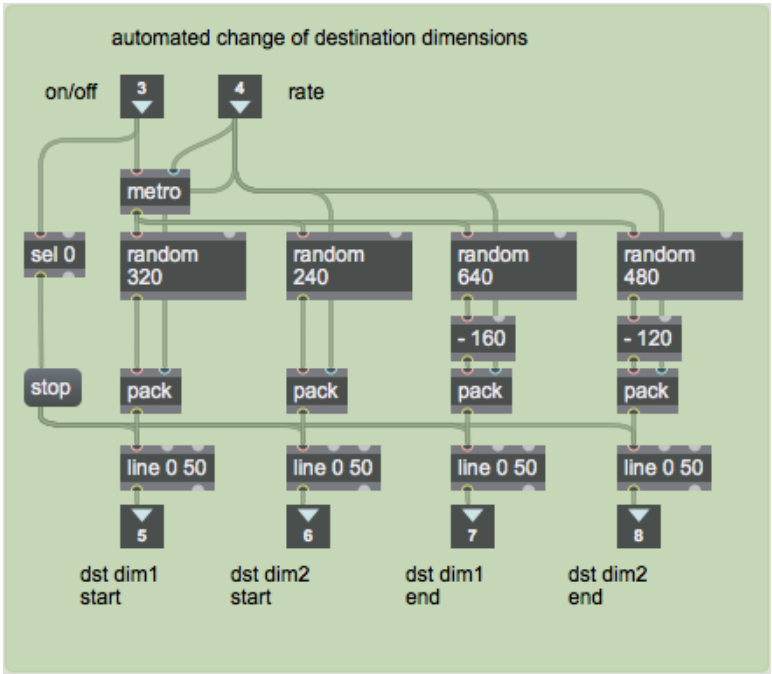
Déplacement des données de l'image dans la matrice

En configurant un processus Max automatisé modifiant les attributs **dstdimstart** et **dstdimend**, nous pouvons déplacer les données dans la matrice, de sorte que l'image semble se déplacer dans l'affichage.

• Dans l'objet *preset*, cliquez sur le preset 6.

Cela lance un processus automatisé à l'intérieur du sub-patcher **move_around** qui fournit un flux continu de nouveaux arguments pour les attributs **dstdimstart** et **dstdimend**. (il est connecté aux sections inférieures avec des cordons de connexion cachés.) Le *toggle* au-dessus du patcher active ce processus et la boîte de *nombre* donne un temps, en millisecondes, pour chaque déplacement vers une nouvelle destination.

• Double-cliquez sur l'objet *patcher* **move_around** pour voir le contenu du sub-patch. Jusqu'à présent, nous n'utilisons que la moitié droite du sub-patch.



Le processus de déplacement de destination dans le sub-patch [**move_around**]

La valeur "rate" qui arrive dans l'entrée de droite est un intervalle de temps pour l'objet *metro*. Le *metro* frappe - **bang** - périodiquement quatre objets aléatoires qui choisissent au hasard de nouveaux index de cellules à gauche, en haut, à droite et en bas. Ces points de destination sont envoyés avec les valeurs de temps aux objets *line*. Les objets *line* envoient de nouvelles valeurs toutes les 50 ms (la vitesse à laquelle nous affichons l'image) pour déplacer progressivement la région de destination vers ces nouveaux points aléatoires. En dehors du sub-patch, ces valeurs sont utilisées comme arguments pour les attributs **dstdimstart** et **dstdimend** de *jit.matrix*.

Ce sub-patch contient quelques astuces à noter. La première astuce est que nous avons fait en sorte que les arguments pour **dstdimend** puissent potentiellement dépasser la plage 320x240 de la matrice. Par exemple, nous utilisons un objet aléatoire **640** pour la dimension horizontale, puis nous

soustrayons 160 du résultat pour obtenir un indice de cellule finale compris de -160 à 479. Nous faisons cela pour augmenter la probabilité d'une zone de destination plus grande, de sorte que nous puissions avoir une vue plus grande de l'image lorsqu'elle se déplace, ce qui signifie également que l'image se déplacera plus fréquemment jusqu'au bord de la fenêtre. Il est à noter que nous pouvons spécifier des limites de destination qui sont au-delà des limites des cellules réelles dans la matrice, et *jit.matrix* placera l'image dans cette zone au mieux de ses capacités (en la coupant lorsqu'elle dépasse les limites de dimensions de la matrice.). La deuxième astuce est un détail trivial mais utile: nous utilisons un objet *sel* **0** pour détecter à quel moment le *metro* est désactivé, et nous l'utilisons pour déclencher un message **stop** à chacun des objets *line* afin qu'ils ne continuent pas à envoyer des valeurs après que l'utilisateur ait éteint le processus.

• Fermez la fenêtre de sub-patch [**move_around**].

Modifier, redimensionner et déplacer l'image source

Nous allons maintenant automatiser les modifications apportées à l'image source.

• Dans l'objet *preset*, cliquez sur le preset 7.

De la même manière que nous l'avons fait pour la zone de destination, nous modifions maintenant continuellement la zone source de l'image. En effet, nous visualisons maintenant une vue en constante évolution d'un sous-ensemble rectangulaire de la matrice source (à l'aide de **srcdimstart** et de **srcdimend**), tout en redimensionnant constamment cette vue et en la déplaçant dans la fenêtre (à l'aide de **dstdimstart** et de **dstdimend**). Comme les rectangles de source et de destination sont choisis de manière aléatoire par le sub-patch [**move_around**], l'image est parfois inversée. Nous avons activé l'attribut **interp** dans l'objet *jit.matrix* pour atténuer la pixellisation qui se produirait lorsque l'image source est étirée.

• Pour obtenir une vue un peu plus claire de ce qui se passe, essayez d'activer le *toggle* marqué **Erase previous image**.

Encore un mot sur les dimensions

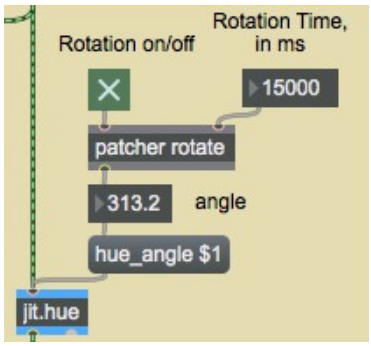
Ce tutoriel a montré comment modifier les dimensions d'un objet *jit.matrix* et comment spécifier les régions source et destination au sein de cet objet. Pour faciliter la discussion et la visualisation, nous avons utilisé une matrice à deux dimensions et spécifié des rectangles de source et de destination dans la matrice. Mais il convient de souligner que ces idées peuvent également être utilisées avec des matrices ayant un nombre quelconque de dimensions. (Le nombre d'arguments pour **srcdimstart**, **srcdimend**, **dstdimstart** et **dstdimend** doit correspondre au nombre de dimensions de l'objet *jit.matrix*.) Par exemple, si nous avons une matrice à trois dimensions, ces arguments peuvent être utilisés pour spécifier un hexaèdre dans l'espace virtuel 3D de la matrice.

> Remarque: Dans certains objets Jitter qui traitent exclusivement des matrices 2D, telles que *jit.movie*, les régions source et destination seront toujours des zones rectangulaires. Ainsi, dans ces objets, les zones source et destination sont définies dans des attributs uniques appelés **srcrect** et **dstrect**, qui prennent quatre arguments pour spécifier les cellules de délimitation (coins supérieur gauche et inférieur droit) des rectangles.

Rotation des teintes

Juste pour ajouter une petite variété de couleurs supplémentaire, nous avons placé un objet *jit.hue* entre les deux objets *jit.matrix*. (jit.hue est décrit en détail dans le ***Tutorial 7***.)

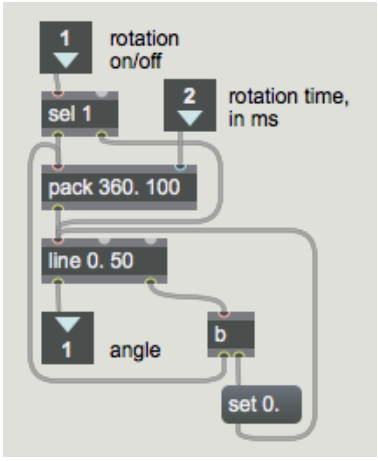
• Dans l'objet *preset*, cliquez sur le preset 8 pour voir *jit.hue* en action.



Modifier l'angle des teintes.

Ce preset désactive **usedstdim**, mais garde **usesrcdim** activé, et garde l'interpolation dans *jit.matrix* pour brouiller l'image étendue. Le processus automatisé dans le sub-patch **rotate** du patcher fait constamment pivoter l'angle de teinte de *jit.hue*.

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



Le contenu du sub-patch [**rotate**]

La valeur qui arrive dans l'entrée droite fournit un temps, en millisecondes, pour compléter une rotation de la teinte de 360 °. Lorsqu'un **1** arrive dans l'entrée gauche, le nombre **360** est combiné à cette valeur de temps pour demander à l'objet *line* d'aller de 0 à 360 dans ce laps de temps, en envoyant une nouvelle valeur d'angle toutes les 50 millisecondes. Notez que le premier argument saisi de l'objet *line* contient un point décimal. Ceci indique à *line* d'envoyer des valeurs **float** plutôt que **ints**, pour plus de précision (et parce que le message **hue_angle** de *jit.hue* attend un argument float). Lorsque *line* atteint 360, sa sortie droite envoie un **bang**. Nous utilisons ce **bang** pour remettre la valeur interne de *line* à 0, puis nous envoyons un re-**bang** à l'objet *pack* pour commencer la rotation suivante. Lorsqu'un **0** entre dans l'entrée gauche, l'objet *sel* **1** le passe directement à *line* pour arrêter *line* et remettre l'angle de teinte à 0.

• Fermez la fenêtre du sub-patch [**rotate**].

• Dans l'objet *preset*, cliquez sur le preset 9. Ceci combine pratiquement toutes les techniques d'automatisation et de manipulation d'image du patch. Les modifications des dimensions de destination de *jit.matrix* sont cette fois réglées sur 200 ms, ce qui crée un effet rythmique plus rapide.

Affichage plein écran

Lorsque votre patch Max crée juste les images souhaitées et que vous voulez afficher vos résultats d'une manière un peu plus élégante, vous pouvez demander à *jit.window* de remplir tout l'écran. *jit.window* a un attribut appelé **fullscreen**; lorsque **fullscreen** est activé, la fenêtre *jit.window* utilise la totalité de l'écran comme zone d'affichage. Si vous décochez l'attribut **fsmenubar** dans l'inspecteur *jit.window*, la barre de menus sera masquée.

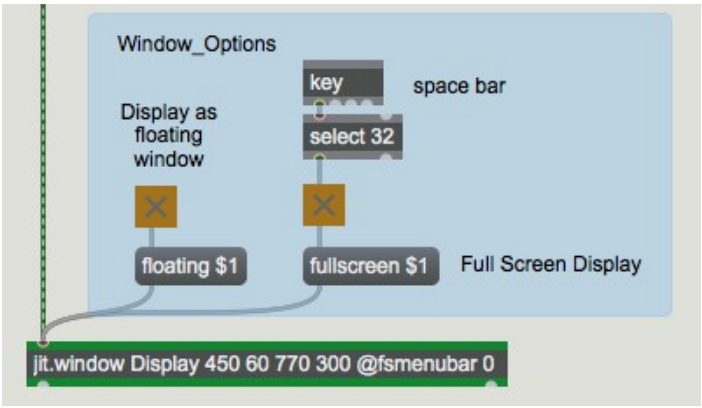
Il y a quelques points à retenir concernant propos de l'utilisation de la fonctionnalité plein écran de *jit.window*.

Tout d'abord, une fois que vous avez rempli votre écran avec une image (et surtout si vous avez également masqué la barre de menu), vous ne pourrez plus utiliser votre souris pour désactiver **fullscreen**. Vous devrez donc programmer dans votre patch Max une méthode permettant de ramener l'attribut **fullscreen** à 0.

Deuxièmement, une seule *jit.window* peut remplir un écran à un moment donné. Si vous avez plusieurs objets *jit.window* en lice pour accéder au plein écran, l'objet *jit.window* dont l'attribut **fullscreen** a été le plus récemment réglé sur **1** remplira l'écran.

En outre, même lorsque une fenêtre *jit.window* est en plein écran, sa résolution est déterminée par ses dimensions réelles (c'est-à-dire par les arguments de son attribut **rect**). Donc, si l'attribut **rect** décrit un rectangle de 320 x 240, ce sera la résolution de votre image, même si les dimensions de votre écran sont beaucoup plus grandes que cela.

Dans le sub-patch **Window_options**, nous avons inclus la possibilité d'activer et de désactiver l'attribut **fullscreen** de *jit.window* à l'aide de la barre d'espace de votre clavier. Notez que l'objet *jit.window* a un attribut nommé **fsmenubar** défini à 0. Cela supprime la barre de menus supérieure (sous Macintosh) lorsque la fenêtre est étendue.



Utilisation de la barre d'espace pour basculer la fenêtre en affichage plein écran.

• Essayez d'activer et de désactiver le mode **fullscreen** avec la barre d'espace.

• Pour obtenir un effet visuel plus abstrait, essayez d'importer l'image *colorswatch.pict* dans la matrice *jit.matrix* en haut du patch, puis essayez les différents préréglages.

Dans ce didacticiel, nous avons utilisé une image fixe comme matériau source pour que vous puissiez facilement voir les effets démontrés, mais rien ne nous empêche d'utiliser une vidéo (de *jit.movie* ou d'une autre source vidéo) comme matériel de base. (Vous pourriez vouloir copier le contenu de ce patch dans une nouvelle fenêtre du Patcher et modifier la partie supérieure gauche de

celui-ci pour essayer cela.)

Sommaire

Il existe plusieurs façons d'isoler et de repositionner certaines données dans une matrice. L'attribut **dim** de *jit.matrix* définit les dimensions et la taille réelles de la matrice. En activant les attributs **usesrcdim** et **usedstdim** de *jit.matrix*, vous pouvez lui demander d'utiliser une partie particulière de ses matrices d'entrée et de sortie, appelées régions source et destination de la matrice. Vous spécifiez les limites des cellules de ces régions avec les attributs **srcdimstart** et **srcdimend** (pour définir les cellules de début et de fin comme les angles de la région source) et les attributs **dstdimstart** et **dstdimend** (pour la région de destination). Ces attributs ne modifient pas la taille réelle de la matrice, mais ils spécifient quelle partie de la matrice d'entrée sera affichée dans quelle partie de la matrice de sortie lorsque **usesrcdim** et **usedstdim** sont activés. Si les régions source et de destination sont différentes en forme et en taille, *jit.matrix* va soit étendre ou contracter la région source pour l'adapter à la région de destination. Cela entraîne la duplication ou la perte de données, mais peut fournir des effets intéressants d'étirement ou de pixelisation. Les régions source et destination peuvent être modifiées dynamiquement avec les nombres fournis par une autre partie de votre patch Max pour une modification interactive ou automatisée de la taille, de la forme et de la position de l'image.

Lorsque l'attribut **interp** est activé, *jit.matrix* interpole (fournit des valeurs intermédiaires) entre les valeurs lorsqu'une dimension de la région de destination est supérieure à celle de la région source. Cela permet de lisser les effets de pixellisation et d'estomper les changements entre les valeurs des cellules adjacentes.

L'objet *jit.window* affiche la matrice de taille quelconque qu'il reçoit, en utilisant le rectangle d'affichage qui lui a été spécifié dans son attribut **rect**. Si la taille de la matrice reçue diffère de la taille de la zone d'affichage, l'image sera étendue, ou contractée, ou déformée par *jit.window*. Ceci peut également être utilisé pour des effets d'étirement et de pixelisation. *jit.window* possède également un attribut **interp** qui, lorsqu'il est activé, lisse la pixellisation provoquée par cette expansion et cet étirement.

Pour remplir la totalité de l'écran avec une image, vous pouvez activer l'attribut **fullscreen** de *jit.window,* et vous pouvez masquer la barre de menus avec un message **fsmenubar 0**. (N'oubliez pas de vous laisser un moyen de faire revenir la fenêtre du Patcher au premier plan.)

Nous avons présenté les techniques de redimensionnement, de repositionnement, de retournement et d'interpolation des données matricielles pour créer des effets visuels tels que l'étirement, la déformation, le flou et la pixellisation.